

ΚΩΝΣΤΑΝΤΙΝΟΣ Ν. ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ

Αναπληρωτής Καθηγητής  
Εθνικό Μετσόβιο Πολυτεχνείο

# Υπολογιστική Φυσική

Μία Πρακτική Εισαγωγή στην Υπολογιστική  
Φυσική και τον Επιστημονικό Προγραμματισμό



Ελληνικά Ακαδημαϊκά Ηλεκτρονικά  
Συγγράμματα και Βοηθήματα  
[www.kallipos.gr](http://www.kallipos.gr)

# Υπολογιστική Φυσική

## Συγγραφή

Κωνσταντίνος Ν. Αναγνωστόπουλος

## Κριτικός Αναγνώστης

Ιωάννης Ρίζος

## Συντελεστές έκδοσης

Γλωσσική Επιμέλεια: Αναστασία Τσιαδήμου

Γραφιστική Επιμέλεια: Κωνσταντίνος Ν. Αναγνωστόπουλος

Τεχνική Επεξεργασία: Κωνσταντίνος Ν. Αναγνωστόπουλος

ISBN: 978-960-603-112-0

Copyright ©ΣΕΑΒ, 2015



Το παρόν έργο αδειοδοτείται υπό τους όρους της άδειας Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 3.0. Για να δείτε ένα αντίγραφο της άδειας αυτής επισκεφτείτε τον ιστότοπο <https://creativecommons.org/licenses/by-nc-sa/3.0/gr/>

ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΩΝ

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

[www.kallipos.gr](http://www.kallipos.gr)

# ΠΕΡΙΕΧΟΜΕΝΑ

Ακρωνύμια – Αρκτικόλεξα	xī
Αντιστοίχιση Ελληνόγλωσσων-Ξενόγλωσσων Όρων	xiii
Πρόλογος	xvii
<b>1 Ο Υπολογιστής</b>	<b>1</b>
1.1 Το Λειτουργικό Σύστημα . . . . .	3
1.1.1 Filesystem . . . . .	3
1.1.2 Εντολές . . . . .	10
1.1.3 Αναζητώντας Βοήθεια . . . . .	14
1.2 Εργαλεία Επεξεργασίας Κειμένου – Φίλτρα . . . . .	16
1.3 Προγραμματίζοντας με τον Emacs . . . . .	21
1.3.1 Καλώντας τον Emacs . . . . .	22
1.3.2 Αλληλεπιδρώντας με τον Emacs . . . . .	24
1.3.3 Βασική Επεξεργασία Κειμένου . . . . .	26
1.3.4 Κόβοντας και ράβοντας . . . . .	29
1.3.5 Παράθυρα . . . . .	30
1.3.6 Αρχεία και Buffers . . . . .	31
1.3.7 Modes . . . . .	33
1.3.8 Βοήθεια στον Emacs . . . . .	34
1.3.9 Παραμετροποίηση του Emacs . . . . .	36
1.3.10 Ελληνικά στον Emacs . . . . .	37
1.4 Η Γλώσσα Προγραμματισμού: Fortran . . . . .	38
1.4.1 Τα Στοιχειώδη . . . . .	38
1.4.2 Μερικές λεπτομέρειες . . . . .	49
1.4.3 Χειρισμός των arrays . . . . .	55
1.4.4 Ιστορικές Παρατηρήσεις . . . . .	62
1.5 Κοιτάζοντας τα Αποτελέσματα . . . . .	63
1.6 Shell Scripting: Σενάρια Φλοιοού . . . . .	68

<b>2</b>	<b>Περιγραφή της Κίνησης</b>	<b>83</b>
2.1	Κίνηση στο Επίπεδο . . . . .	83
2.1.1	Απεικόνιση των Δεδομένων . . . . .	92
2.1.2	Άλλα Παραδείγματα . . . . .	96
2.2	Κίνηση στο Χώρο . . . . .	106
2.3	Κίνηση μετ' Εμποδίων . . . . .	116
2.3.1	Το Μονοδιάστατο Κουτί . . . . .	117
2.3.2	Σφάλματα . . . . .	123
2.3.3	Το Δισδιάστατο Κουτί . . . . .	127
2.4	Εφαρμογές . . . . .	131
2.5	Ασκήσεις . . . . .	151
<b>3</b>	<b>Η Λογιστική Απεικόνιση</b>	<b>157</b>
3.1	Εισαγωγή . . . . .	157
3.2	Σταθερά Σημεία και 2 <sup>η</sup> Κύκλοι . . . . .	160
3.3	Διάγραμμα διακλάδωσης . . . . .	166
3.4	Μέθοδος Newton-Raphson . . . . .	170
3.5	Υπολογισμός Σημείων Διακλάδωσης . . . . .	176
3.6	Εκθέτες Liapunov . . . . .	181
3.7	Ασκήσεις . . . . .	195
<b>4</b>	<b>Κίνηση Σωματιδίου</b>	<b>207</b>
4.1	Αριθμητική Ολοκλήρωση Εξισώσεων Νεύτωνα . . . . .	207
4.2	Πρελούδιο: Μέθοδοι Euler . . . . .	208
4.3	Μέθοδοι Runge-Kutta . . . . .	220
4.3.1	Προγραμματισμός της Runge-Kutta 4ης τάξης . . . . .	225
4.4	Σύγκριση των Μεθόδων . . . . .	229
4.5	Ταλαντώσεις με Απόσβεση και Διέγερση . . . . .	232
4.6	Εκκρεμές με Απόσβεση και Διέγερση . . . . .	240
4.7	Παράρτημα: Στη Μέθοδο Euler-Verlet . . . . .	247
4.8	Παράρτημα: Runge-Kutta 2ης τάξης . . . . .	251
4.9	Ασκήσεις . . . . .	254
<b>5</b>	<b>Κίνηση στο Επίπεδο</b>	<b>259</b>
5.1	Runge-Kutta για την Κίνηση στο Επίπεδο . . . . .	259
5.2	Βολές στο Βαρυτικό Πεδίο της Γης . . . . .	264
5.3	Κίνηση Πλανητών . . . . .	271
5.4	Σκέδαση . . . . .	275
5.4.1	Σκέδαση Rutherford . . . . .	279
5.4.2	Σκέδαση σε Άλλα Πεδία Δυνάμεων . . . . .	286
5.5	Περισσότερα Σωματίδια . . . . .	289



5.6	Ασκήσεις . . . . .	300
<b>6</b>	<b>Κίνηση στο Χώρο</b>	<b>305</b>
6.1	Runge–Kutta στις τρεις διαστάσεις. . . . .	306
6.2	Κίνηση Σωματίου σε ΗΜ πεδίο. . . . .	315
6.3	Σχετικιστική Κίνηση. . . . .	317
6.4	Ασκήσεις . . . . .	327
<b>7</b>	<b>Ηλεκτροστατική</b>	<b>331</b>
7.1	Σημειακή Κατανομή Φορτίων . . . . .	331
7.2	Το Πρόγραμμα – Ορεκτικά και ... επιδόρπιο . . . . .	334
7.3	Το Πρόγραμμα - Το Κυρίως Πιάτο . . . . .	344
7.4	Το Πρόγραμμα - Σύνοψη . . . . .	351
7.5	Ηλεκτροστατικό Πεδίο στο Κενό . . . . .	355
7.6	Αποτελέσματα . . . . .	364
7.7	Εξίσωση Poisson . . . . .	364
7.8	Ασκήσεις . . . . .	371
<b>8</b>	<b>Εξίσωση Διάχυσης</b>	<b>375</b>
8.1	Εισαγωγή . . . . .	375
8.2	Απαγωγή Θερμότητας . . . . .	377
8.3	Διακριτοποίηση . . . . .	379
8.4	Το Πρόγραμμα . . . . .	380
8.5	Αποτελέσματα . . . . .	382
8.6	Διάχυση Πάνω στον Κύκλο. . . . .	385
8.7	Ανάλυση . . . . .	389
8.8	Ασκήσεις . . . . .	393
<b>9</b>	<b>Ο Αναρμονικός Ταλαντωτής</b>	<b>395</b>
9.1	Εισαγωγή . . . . .	395
9.2	Υπολογισμός Ιδιοτιμών του $H_{nm}(\lambda)$ . . . . .	397
9.3	Αποτελέσματα . . . . .	406
9.4	Το Διπλό Πηγάδι Δυναμικού . . . . .	412
9.5	Ασκήσεις . . . . .	420
<b>10</b>	<b>Η Εξίσωση Schrödinger</b>	<b>423</b>
10.1	Εισαγωγή . . . . .	423
10.2	Το απειρόβαθο πηγάδι δυναμικού . . . . .	426
10.3	Δέσμιες Καταστάσεις . . . . .	438
10.4	Μετρήσεις . . . . .	447
10.5	Ο Αναρμονικός Ταλαντωτής - Ξανά... . . . .	453

10.6 Το Δυναμικό Lennard–Jones . . . . .	458
10.7 Ασκήσεις . . . . .	462
<b>11 Ο Τυχαίος Περιπατητής</b>	<b>467</b>
11.1 (Ψευδο)Τυχαίοι Αριθμοί . . . . .	468
11.2 Χρήση Γεννητριών Ψευδοτυχαίων Αριθμών . . . . .	480
11.3 Τυχαίες Διαδρομές . . . . .	488
11.4 Ασκήσεις . . . . .	497
<b>12 Προσομοιώσεις Μόντε Κάρλο</b>	<b>501</b>
12.1 Στατιστική Φυσική . . . . .	502
12.2 Εντροπία . . . . .	505
12.3 Διακυμάνσεις . . . . .	510
12.4 Συναρτήσεις Συσχετισμού . . . . .	512
12.5 Δειγματοληψία . . . . .	514
12.5.1 Απλή Δειγματοληψία . . . . .	514
12.5.2 Importance Sampling . . . . .	516
12.6 Διαδικασίες Markov . . . . .	516
12.7 Συνθήκη Λεπτομερούς Ισορροπίας . . . . .	518
12.8 Ασκήσεις . . . . .	520
<b>13 Το Πρότυπο Ising</b>	<b>521</b>
13.1 Εισαγωγή . . . . .	521
13.2 Ο Αλγόριθμος Metropolis . . . . .	527
13.3 Σχεδιασμός Κώδικα . . . . .	530
13.3.1 Ο Κώδικας . . . . .	536
13.3.2 Βελτίωση του Interface . . . . .	542
13.4 Θερμική Ισορροπία . . . . .	554
13.5 Αυτοσυσχετισμοί . . . . .	556
13.6 Στατιστικά Σφάλματα . . . . .	564
13.6.1 Σφάλματα Ανεξάρτητων Μετρήσεων . . . . .	566
13.6.2 Jackknife . . . . .	569
13.6.3 Bootstrap . . . . .	571
13.7 Παράρτημα: Συνάρτηση Αυτοσυσχετισμού . . . . .	572
13.8 Παράρτημα: Υπολογισμός Σφαλμάτων . . . . .	580
13.8.1 Η Μέθοδος Jackknife . . . . .	580
13.8.2 Η Μέθοδος Bootstrap . . . . .	584
13.8.3 Σύγκριση των Μεθόδων . . . . .	587
13.9 Ασκήσεις . . . . .	593

<b>14 Κρίσιμοι Εκθέτες</b>	<b>603</b>
14.1 Εισαγωγή . . . . .	603
14.2 Κρίσιμη Επιβράδυνση . . . . .	605
14.3 Ο Αλγόριθμος του Wolff . . . . .	607
14.4 Σχεδιασμός Κώδικα . . . . .	614
14.4.1 Ο Κώδικας . . . . .	616
14.5 Συλλογή Δεδομένων . . . . .	622
14.6 Ανάλυση Δεδομένων . . . . .	625
14.7 Χρόνοι Αυτοσυσχετισμού . . . . .	632
14.8 Βάθμιση Θερμοκρασίας . . . . .	638
14.9 Βάθμιση Πεπερασμένου Μεγέθους . . . . .	643
14.10 Προσδιορισμός της $\beta_c$ . . . . .	647
14.11 Μελέτη Βάθμισης με ... Κατάρρευση . . . . .	653
14.12 Binder Cumulant . . . . .	662
14.13 Παράρτημα: Βάθμιση . . . . .	667
14.13.1 Binder Cumulant . . . . .	667
14.13.2 Βάθμιση . . . . .	672
14.13.3 Βάθμιση Πεπερασμένου Μεγέθους . . . . .	674
14.14 Παράρτημα: Κρίσιμοι Εκθέτες . . . . .	678
14.14.1 Ορισμοί . . . . .	678
14.14.2 Σχέσεις . . . . .	678
14.15 Ασκήσεις . . . . .	680
<b>Bibliography</b>	<b>683</b>

Ο τόμος αυτός έχει γραφτεί με την υπόθεση ότι ο αναγνώστης εκτελεί τις εντολές και τις διαδικασίες που περιγράφονται ταυτόχρονα με την ανάγνωση του κειμένου. Αν αυτό δεν γίνεται, η βοήθεια από το κείμενο θα είναι εξαιρετικά ελλιπής.

Η ιστοσελίδα του βιβλίου είναι στη θέση <http://www.physics.ntua.gr/~konstant/ComputationalPhysics/>. Εκεί θα βρείτε συμπληρωματικό υλικό, το συνοδευτικό λογισμικό με τα προγράμματα που παρουσιάζονται στο βιβλίο, καθώς και τη μετάφραση του βιβλίου στα Αγγλικά (ελεύθερα διαθέσιμη).

**Μερικές συμβάσεις:** Κείμενο με γραμματοσειρά όπως η παρακάτω, αφορά εντολές που δίνονται στον υπολογιστή, είσοδο και έξοδο προγραμμάτων, κώδικα γραμμένο σε Fortran ή άλλη γλώσσα και ονόματα αρχείων:

```
> echo Hello world  
Hello world
```

Όταν μια γραμμή αρχίζει με τον χαρακτήρα “προτροπής” (prompt),

```
>
```

όπως παραπάνω, αυτή είναι μία εντολή που δίνουμε από τη γραμμή εντολών του φλοιού. Η δεύτερη γραμμή δείχνει αυτά που τυπώνει η εντολή στην κονσόλα.

Παρακάτω, δίνονται τα περιεχόμενα ενός αρχείου Fortran:

```
program add  
  
  z = 1.0  
  y = 2.0  
  x = z + y  
  print *, x  
  
end program add
```

Τι χρειάζεστε για να δουλέψετε στον υπολογιστή σας:

- Ένα λειτουργικό σύστημα τύπου GNU/Linux και τα βασικά εργαλεία του.
- Έναν μεταγλωττιστή (compiler) για τη γλώσσα Fortran. Ο μεταγλωττιστής gfortran διατίθεται ελεύθερα υπό άδεια ελεύθερου λογισμικού<sup>1</sup>.
- Ένα προηγμένο πρόγραμμα επεξεργασίας κειμένου κατάλληλο για προγραμματιστές, όπως ο Emacs<sup>2</sup>.
- Ένα καλό πρόγραμμα, κατάλληλο για ανάλυση δεδομένων, για να κάνετε γραφικές παραστάσεις, όπως το gnuplot<sup>3</sup>.
- Τον φλοιό tcsh<sup>4</sup>.
- Τα προγράμματα gawk<sup>5</sup>, grep, sort, cat, head, tail, less. Βεβαιωθείτε ότι είναι στη διάθεσή σας.

Αν έχετε μια διανομή GNU/Linux εγκατεστημένη στον υπολογιστή σας, η εγκατάσταση του παραπάνω λογισμικού γίνεται πολύ εύκολα από τον διαχειριστή πακέτων (software center) της διανομής. Λ.χ., σε μια διανομή τύπου Debian (Ubuntu, ...), αφού εκτελέσετε τις απλές εντολές

```
> sudo apt-get install tcsh emacs gnuplot-x11 gnuplot-doc  
> sudo apt-get install gfortran gawk gawk-doc binutils  
> sudo apt-get install manpages-dev coreutils liblapack3
```

θα βρείτε όλο το λογισμικό εγκατεστημένο στον υπολογιστή σας.

Αν δεν θέλετε να εγκαταστήσετε μια διανομή GNU/Linux στον υπολογιστή σας, έχετε τις εξής εναλλακτικές δυνατότητες:

- Εκκινήστε τον υπολογιστή σας από το DVD ή usb stick με μια live διανομή, όπως αυτή του Ubuntu<sup>6</sup>. Αυτή η επιλογή δεν θα αλλοιώσει τίποτα στον υπολογιστή σας, αν και θα τρέχει αργά.

---

<sup>1</sup><http://www.gfortran.org>

<sup>2</sup><http://www.gnu.org/software/emacs/>

<sup>3</sup><http://www.gnuplot.info>

<sup>4</sup><http://www.tcsh.org>

<sup>5</sup><http://www.gnu.org/software/gawk>

<sup>6</sup><http://www.ubuntu.com>

- Εγκαταστήστε τη διανομή Cygwin<sup>7</sup> στα Microsoft Windows. Είναι μια πολύ καλή επιλογή για όσους προτιμούν τα προϊόντα της Microsoft.
- Το λειτουργικό σύστημα Mac OS X βασίζεται στο Unix, το οποίο, σε επίπεδο φλοιού, λειτουργεί παρόμοια με το Linux. Τα πακέτα λογισμικού που αναφέρονται παραπάνω είναι διαθέσιμα και για τα Mac. Αναζητήστε πληροφορίες μέσω του Google: “gfortran for Mac”, “emacs for Mac”, “tcsh for Mac” κλπ.

Στην ιστοθέση <https://goo.gl/2pLXT2> θα βρείτε βίντεο που θα σας βοηθήσουν με τα παραπάνω.

---

<sup>7</sup><http://www.cygwin.com>

# Ακρωνύμια – Αρκτικόλεξα

blas	- Basic Linear Algebra Subprograms.
CPU	- Central Processing Unit.
FIFO	- First In First Out (queue).
FORTRAN	- FORMula TRANslator.
GNU	- GNU's Not Unix!
HM	- Ηλεκτρομαγνητικό (πεδίο).
lapack	- Linear Algebra PACKage.
LDA	- Leading Dimension of A.
LIFO	- Last In First Out (stack).
NRRW	- Non Reversal Random Walk.
pdf	- Portable Document Format.
RK45	- Runge–Kutta τάξης 4 προσαρμοζόμενου βήματος.
RW	- Random Walk (απλό).
SAW	- Self Avoiding Walk.
SOR	- Successive OverRelaxation.
tcsh	- Trusted C SHell.





# Αντιστοίχιση Ελληνόγλωσσων-Ξενόγλωσσων Όρων

allocation	- Εκχώρηση (memory allocation = εκχώρηση μνήμης υπολογιστή σε ένα πρόγραμμα, array κλπ).
anharmonic (oscillator)	- Αναρμονικός (ταλαντωτής).
animation	- Κινούμενα σχέδια. Αναφέρεται σε γραφική παράσταση δεδομένων που αλλάζουν στον χρόνο.
ansatz	- Δοκιμαστική λύση σε ένα πρόβλημα ή μία εξίσωση.
argument	- Όρισμα εντολής: λέξεις που το πρόγραμμα διαβάζει από τη γραμμή εντολών.
array	- Δομές δεδομένων στη Fortran στα οποία αναφερόμαστε με χρήση ακέραιων δεικτών (λ.χ. διανύσματα, πίνακες κλπ).
attractor	- Έλκυστής: υποσύνολο του φασικού χώρου ενός δυναμικού συστήματος στο οποίο κινείται οριακά το σύστημα για μεγάλους χρόνους.
attribute	- Χαρακτηρισμός που δίνεται στη δήλωση μίας μεταβλητής Fortran (λ.χ. parameter, allocatable κλπ).
autocorrelation	- Αυτοσυσχετισμός: στατιστικός συσχετισμός τυχαίας μεταβλητής με τον εαυτό της στον χρόνο.
bifurcation	- Διακλάδωση (εδώ των σταθερών σημείων της λογιστικής απεικόνισης).

binning	- Μέθοδος υπολογισμού σφαλμάτων με χωρισμό των δεδομένων σε ομάδες (bins - “δοχεία”).
boundary	- Σύνορο (boundary conditions = συνοριακές συνθήκες διαφορικής εξίσωσης).
buffer	- Βοηθητική περιοχή στη μνήμη. Στον Emacs, buffers είναι περιοχές στα παράθυρα που συνήθως περιέχουν δεδομένα αρχείων.
cluster	- Σύμπλεγμα από πλεγματικές θέσεις με το ίδιο spin που έχουν επιλεγεί με κάποια διαδικασία.
compiler	- Μεταγλωττιστής εντολών μιας γλώσσας προγραμματισμού σε μία άλλη γλώσσα (λ.χ. από Fortran σε assembly).
constraints	- Περιορισμοί στους οποίους υπόκειται ένα δυναμικό σύστημα.
correlation	- Στατιστικός συσχετισμός.
debugger	- Ειδικό εργαλείο αποσφαλμάτωσης προγραμμάτων.
diffusion	- Διάχυση.
directory	- Κατάλογος αρχείων.
eigenvalue	- Ιδιοτιμή πίνακα.
eigenvector	- Ιδιοδιάνυσμα πίνακα.
eigenfunction	- Ιδιοσυνάρτηση προβλήματος ιδιοτιμών διαφορικής εξίσωσης.
file	- Αρχείο δεδομένων.
filesystem	- Σύστημα οργάνωσης αρχείων ενός λειτουργικού συστήματος.
fit	- Προσαρμογή δεδομένων σε μία συνάρτηση.
format	- Μορφοποίηση δεδομένων στη Fortran (λ.χ. για την εκτύπωσή τους).
harmonic (oscillator)	- Αρμονικός (ταλαντωτής).
impurities	- Προσμείξεις (σε ένα υλικό).
integer	- Μεταβλητές της Fortran που καταχωρούν τα δεδομένα για έναν ακέραιο αριθμό.
interface	- Διεπαφή (εδώ ενός προγράμματος με τον χρήστη).

iteration	- Επανάληψη μίας επαναληπτικής (αλγοριθμικής) διεργασίας.
jackknife	- Μέθοδος υπολογισμού σφαλμάτων με χωρισμό των δεδομένων σε ομάδες.
logistic map	- Λογιστική απεικόνιση.
minibuffer	- Ειδικό buffer στον Emacs στο οποίο μπορούμε να δίνουμε εντολές.
modes	- Ειδικές καταστάσεις στις οποίες βρίσκονται buffers του Emacs ανάλογα με το περιεχόμενο των δεδομένων (Fortran mode, C mode κλπ).
observable path	- Παρατηρήσιμη φυσική ποσότητα. - Διαδρομή που οδηγεί σε μία μοναδική θέση σε ένα filesystem.
plot	- Γραφική παράσταση δεδομένων ή συνάρτησης.
prompt	- Ακολουθία χαρακτήρων που τυπώνει ένα πρόγραμμα όταν περιμένει είσοδο δεδομένων από τον χρήστη (μήνυμα προτροπής).
random	- Τυχαίο.
random number	- Τυχαίος αριθμός. Ένας μεγάλος αριθμός από αυτούς ακολουθεί μια ζητούμενη κατανομή πιθανότητας.
random walk	- Τυχαία διαδρομή.
real	- Μεταβλητές της Fortran που καταχωρούνται (προσεγγιστικά) δεδομένα για έναν πραγματικό αριθμό.
relaxation	- Μέθοδος ολοκλήρωσης προβλήματος συνολικών τιμών, που στην είσοδο δίνεται δοκιμαστική λύση και η οποία, με την επαναληπτική εφαρμογή του αλγόριθμου, προσεγγίζει τη ζητούμενη λύση.
residual	- Υπολειπόμενο μιας εξίσωσης που αντιστοιχεί στο σφάλμα προσέγγισής της από έναν αλγόριθμο.
scattering	- Σκέδαση (λ.χ. σωματιδίων).
script	- Εντολές-σενάριο οι οποίες ερμηνεύονται, χωρίς να μεταγλωττίζονται, από κάποιο πρόγραμμα (λ.χ. τον φλοιό).

seed	- Αρχική τιμή σε μια γεννήτρια ψευδοτυχαίων αριθμών. Αρχική πλεγματική θέση σε ένα Wolff cluster.
semicolon	- Ο χαρακτήρας “;” (ελληνικό ερωτηματικό).
shell	- Φλοιός, ειδικό πρόγραμμα αλληλεπίδρασης ενός χρήστη με το λειτουργικό σύστημα.
shell script	- Σενάριο φλοιού: πρόγραμμα που εκτελεί μία ακολουθία εντολών ενός φλοιού.
stdin (standard input)	- ειδικό αρχείο εισόδου δεδομένων.
stdout (standard output)	- ειδικό αρχείο εξόδου δεδομένων.
stderr (standard error)	- ειδικό αρχείο εξόδου δεδομένων για σφάλματα εκτέλεσης προγραμμάτων.
subroutine	- Υπορουτίνα: αυτόνομη διαδικασία στη Fortran, μία συνάρτηση που δεν επιστρέφει αποτέλεσμα (δηλ. τύπου void).
sweep	- Ενημέρωση ή απόπειρα ενημέρωσης όλων των βαθμών ελευθερίας ενός πλέγματος σε μια διαδικασία Μόντε Κάρλο.
thermalization	- Διαδικασία εύρεσης κατάστασης θερμικής ισορροπίας σε έναν αλγόριθμο Μόντε Κάρλο.
transient state	- Μεταβατική συμπεριφορά ενός δυναμικού συστήματος, προτού φτάσει στη σταθερή κατάσταση.

# Πρόλογος

Το βιβλίο είναι αποτέλεσμα της εντεκάχρονης εμπειρίας μου στη διδασκαλία τριών εισαγωγικών μαθημάτων με θέμα την υπολογιστική φυσική και τον επιστημονικό προγραμματισμό στο Εθνικό Μετσόβιο Πολυτεχνείο. Απευθύνεται, κυρίως, σε τριτοετείς και τεταρτοετείς φοιτητές των φυσικών επιστημών και των επιστημών του μηχανικού. Τα πρώτα του κεφάλαια μπορούν να διδαχθούν χωρίς πρόβλημα και σε δευτεροετείς φοιτητές που έχουν παρακολουθήσει τα βασικά μαθήματα φυσικής και μαθηματικής ανάλυσης, τα οποία διδάσκονται στο πρώτο έτος σε ένα οποιοδήποτε τμήμα θετικών επιστημών. Το υλικό που παρουσιάζεται στο βιβλίο μπορεί να διδαχθεί άνετα σε δύο εξαμηνιαία μαθήματα, συμπεριλαμβανομένων και των εργαστηριακών ασκήσεων.

Το βασικό κίνητρο που με οδήγησε στη συγγραφή του βιβλίου είναι, καταρχήν, η απουσία ελληνικής βιβλιογραφίας η οποία να θεραπεύει τα θέματα που παρουσιάζονται στα περιεχόμενά του σε προπτυχιακό επίπεδο, αλλά και η ανάγκη μου να δείξω στους δικούς μου φοιτητές όλες τις τεχνικές λεπτομέρειες ενός αριθμητικού υπολογισμού σε ένα επιστημονικό μοντέλο, από τον σχεδιασμό μέχρι την υλοποίηση και την ανάλυση των αποτελεσμάτων. Οι φοιτητές μου αντιμετωπίζουν περισσότερες δυσκολίες στον προγραμματισμό και στον χειρισμό των δεδομένων, παρά στην κατανόηση των φυσικών εννοιών.

Το βιβλίο δεν αποσκοπεί στο να γίνει ένα βιβλίο αναφοράς για τα θέματα που διαπραγματεύεται, αλλά να διδάξει βήμα-βήμα πώς να λυθεί ένα επιστημονικό πρόβλημα με υπολογιστικές μεθόδους. Τονίζεται πως, για να επιτευχθούν οι εκπαιδευτικοί στόχοι, είναι απαραίτητο ο αναγνώστης να εργάζεται, ταυτόχρονα με τη μελέτη του βιβλίου, πάνω στον προσωπικό του υπολογιστή και να υλοποιεί τα γραφόμενα. Το μάθημα το διδάσκω, ακολουθώντας την παραπάνω φιλοσοφία, μέσα σε ένα εργαστήριο υπολογιστών, όπου οι φοιτητές ασκούνται ταυτόχρονα με τη διδασκαλία της ύλης.

Οι απαραίτητες υπολογιστικές δεξιότητες διδάσκονται με τη μέθοδο του παραδείγματος και, για τον λόγο αυτό, καλό είναι η ύλη να διδα-

χθεί με τη σειρά που παρουσιάζεται στο βιβλίο. Στο πρώτο κεφάλαιο, παρουσιάζονται οι απολύτως απαραίτητες έννοιες και η εμβάθυνση επιτυγχάνεται στην πράξη, λύνοντας προβλήματα (όπως και στον πραγματικό κόσμο...). Η λύση των προβλημάτων παρουσιάζεται σε διαφορετικά επίπεδα υπολογιστικής δεξιότητας και, ανάλογα με το επίπεδο του αναγνώστη, μπορούν να λυθούν χρησιμοποιώντας απλά ή σύνθετα εργαλεία. Σε κάθε κεφάλαιο παρατίθεται βιβλιογραφία που έχει στόχο να βοηθήσει τον αναγνώστη να εμβαθύνει τη γνώση του πάνω σε ζητήματα που δεν υπάρχει αρκετός χώρος να αναπτυχθούν στο βιβλίο.

Ένας άλλος βασικός στόχος του βιβλίου είναι να βοηθήσει τον αναγνώστη να αποκτήσει εμπειρία και στερεό υπόβαθρο, προκειμένου, αν θέλει, να προχωρήσει σε αριθμητικούς υπολογισμούς υψηλής απόδοσης. Για τον λόγο αυτό, ως γλώσσα προγραμματισμού των βασικών προγραμμάτων που υλοποιούν έναν αλγόριθμο υψηλών αριθμητικών απαιτήσεων έχει επιλεγεί η Fortran. Η γλώσσα αυτή είναι δημοφιλής σε ομάδες που προγραμματίζουν επιστημονικά προγράμματα στους σημερινούς υπερυπολογιστές. Είναι δομημένη με σκοπό να κάνει εύκολο τον προγραμματισμό αριθμητικών αλγόριθμων υψηλών απαιτήσεων, έχει τους πιο αποδοτικούς μεταγλωττιστές στη βελτιστοποίηση εκτέλεσης προγραμμάτων και ένα μεγάλο μέρος αριθμητικών βιβλιοθηκών έχουν προγραμματιστεί στη γλώσσα αυτή. Από παιδαγωγικής άποψης είναι μια πολύ απλή γλώσσα, που επιτρέπει στον μη έμπειρο προγραμματιστή να αρχίσει να προγραμματίζει άμεσα ένα αριθμητικό πρόβλημα, χωρίς να χρειάζεται να καθορίσει δευτερεύουσας σημασίας παραμέτρους του υπολογιστικού περιβάλλοντος. Η εκτέλεση των προγραμμάτων και η ανάλυση των αποτελεσμάτων γίνεται σε ένα λειτουργικό σύστημα της οικογένειας του Unix (όπως είναι λ.χ. το Linux), εμπλουτισμένο από την πλούσια και πανίσχυρη εργαλειοθήκη GNU. Η τελευταία παρέχεται ελεύθερα από την FSF<sup>8</sup> και είναι απαραίτητη για τον πολύπλοκο χειρισμό δεδομένων που απαιτούνται σε ένα ερευνητικό πρόγραμμα. Η Fortran δεν είναι η καλύτερη επιλογή για τον προγραμματισμό λειτουργιών στις οποίες είναι απαραίτητος ο χειρισμός πολύπλοκων αντικειμένων ή η σε βάθος αλληλεπίδραση με το λειτουργικό σύστημα. Η φιλοσοφία είναι να αφήσει κάποιος τη Fortran να κάνει αυτό για το οποίο έχει φτιαχτεί να κάνει καλύτερα (αριθμητικούς υπολογισμούς) και να αναθέσει τον χειρισμό των δεδομένων και τη διαχείριση του συστήματος σε άλλα, εξωτερικά, εργαλεία. Εργαλεία όπως η awk, ο προγραμματισμός του φλοιού, το gnuplot, η Perl και άλλα, είναι πολύ ισχυρά και ευέλικτα, και συμπληρώνουν τις αδυναμίες της Fortran. Το πρόγραμμα που

---

<sup>8</sup>Free Software Foundation, [www.fsf.org](http://www.fsf.org)

χρησιμοποιείται στην απεικόνιση των δεδομένων είναι το `gnuplot`, το οποίο παρέχει πανίσχυρα εργαλεία χειρισμού των δεδομένων και δημιουργίας μεγάλου αριθμού από πολύπλοκες γραφικές παραστάσεις. Όλα τα εργαλεία που χρησιμοποιούνται στο βιβλίο δίνονται με άδεια ανοιχτού λογισμικού και είναι προσβάσιμα χωρίς χρέωση. Μπορούν να χρησιμοποιηθούν σε περιβάλλον Linux, Windows και Mac OS.

Η πιο δύσκολη έννοια που πρέπει να εμπεδωθεί σε ένα μάθημα επιστημονικού προγραμματισμού, είναι ότι ο τρόπος λύσης ενός προβλήματος με αριθμητική μέθοδο είναι τελείως διαφορετικός από τον τρόπο που λύνεται αναλυτικά. Συνήθως, οι φοιτητές προσέρχονται έχοντας ένα ισχυρό υπόβαθρο στην ανάλυση και τη θεμελιώδη πανεπιστημιακή φυσική και είναι δύσκολο να τους εξηγήσεις πώς μπορείς να λύσεις ένα πρόβλημα ανάλυσης, χρησιμοποιώντας μόνο απλές αριθμητικές πράξεις. Ακόμα πιο δύσκολο είναι να γίνει κατανοητό πως το πρόβλημα λύνεται, συνήθως, με τη διακριτοποίηση ενός μοντέλου ορισμένου στο συνεχές, κάτι το οποίο μπορεί να γίνει με πολλούς τρόπους, ανάλογα με τις ανάγκες ακρίβειας και χρήσης υπολογιστικών πόρων. Η προσεγγιστική αριθμητική λύση πρέπει να προεκταθεί στο απειροστικό όριο, έτσι ώστε ληφθεί η αναλυτική λύση με ικανοποιητική ακρίβεια. Το βιβλίο προσπαθεί να εκθέσει τον αναγνώστη σε αυτή την ιδέα προοδευτικά, αρχίζοντας από προβλήματα απλής κίνησης σωματιδίων και φτάνοντας στην παρουσίαση της μεθόδου βάθμισης πεπερασμένου μεγέθους στη στατιστική φυσική ενός μοντέλου που βρίσκεται στην περιοχή μιας συνεχούς μετάβασης φάσης.

Το βιβλίο δίνεται μαζί με συνοδευτικό υλικό που μπορεί να βρεθεί και στην ιστοθέση του <sup>9</sup>. Το συνοδευτικό λογισμικό περιέχει όλα τα προγράμματα που παρουσιάζονται στο βιβλίο, μαζί με χρήσιμα εργαλεία και λύσεις μερικών από τα προβλήματα. Κάθε κεφάλαιο συμπληρώνεται από ασκήσεις που ο αναγνώστης πρέπει να λύσει για να αποκτήσει “hands on” εμπειρία στον επιστημονικό προγραμματισμό. Ελπίζω πως έχω ήδη τονίσει αρκετά, πως είναι απαραίτητο ο αναγνώστης να εκτελεί τις εντολές που παρουσιάζονται στο βιβλίο κατά τη διάρκεια της μελέτης του.

Σας εύχομαι μια δημιουργική εμπειρία επιστημονικού προγραμματισμού!

Αθήνα 2015.

---

<sup>9</sup>[www.physics.ntua.gr/~konstant/ComputationalPhysics/](http://www.physics.ntua.gr/~konstant/ComputationalPhysics/)





# ΚΕΦΑΛΑΙΟ 1

## Ο Υπολογιστής

Σκοπός του κεφαλαίου αυτού είναι να θέσει τα θεμέλια για την ανάπτυξη δεξιοτήτων χρήσης των υπολογιστικών εργαλείων που θα χρησιμοποιήσουμε στη μελέτη των υπολογιστικών προβλημάτων που παρουσιάζονται στα επόμενα κεφάλαια. Δεν έχει σκοπό να κάνει πλήρη και εις βάθος παρουσίαση, είναι μάλλον πρακτική εκμάθηση μέσω παραδειγμάτων. Άλλωστε υπάρχουν πολλές πλήρεις και παιδαγωγικές παρουσιάσεις του υλικού που θα παρουσιάσουμε σε πολλά βιβλία ελεύθερα διαθέσιμα στο διαδίκτυο ή/και σε βιβλία τα οποία ... έχουν κάποιο τίμημα. Στη βιβλιογραφία θα σας δοθούν αρκετές προτάσεις για μελέτη σε διαφορετικά επίπεδα εμπάθυνσης.

Σε κάθε περιβάλλον εργασίας ενός υπολογιστικού προγράμματος, είναι ανάγκη να γίνουν επιλογές. Αυτές εξαρτώνται από τις συγκεκριμένες ανάγκες του προγράμματος: Απαιτήσεις αριθμητικής αποτελεσματικότητας, μικρή/μεγάλη ομάδα εργασίας, πολυπλοκότητα κώδικα, ανάγκες για αναβαθμίσεις ... αναμνήσεις από το μέλλον.

Εμείς εδώ θα διαλέξουμε να πάρουμε ένα άρωμα από τις ανάγκες ενός προγράμματος με κατεύθυνση επιστημονική/υπολογιστική. Ενόσ προγράμματος με μεγάλες ανάγκες σε εκμετάλλευση των υπολογιστικών πόρων για γρήγορους αριθμητικούς υπολογισμούς και για ευέλικτη ανάλυση (...πολλών) δεδομένων. Ένα τέτοιο περιβάλλον που προσφέρει ευελιξία, αξιοπιστία, απλότητα, δυνατά εργαλεία για ανάλυση δεδομένων και μεταγλώττιση προγραμμάτων και που προσφέρει στο χρήστη τη δυνατότητα να κάνει αποδοτικότερη χρήση των υπολογιστικών πόρων του συστήματός του είναι η ομάδα λειτουργικών συστημάτων Unix. Η σύγχρονη, δημοφιλής και ελεύθερα διαθέσιμη έκδοση τέτοιου συστήματος είναι το GNU/Linux<sup>1</sup>, μια προσπάθεια η οποία πραγματοποιήθηκε

---

<sup>1</sup>[www.gnu.org](http://www.gnu.org)

χάρη στην εθελοντική δουλειά εκατομμυρίων προγραμματιστών παγκοσμίως και που βασίστηκε στην ιδέα του Ελεύθερου Λογισμικού (όχι με την έννοια “τσάμπα”, αλλά με την έννοια της ελεύθερης διακίνησης ιδεών στο λογισμικό) που θεμελίωσε ο Richard Stallman<sup>2</sup>.

Η γλώσσα προγραμματισμού που θα διαλέξουμε είναι η Fortran. Μερικοί λόγοι για την επιλογή είναι ότι η γλώσσα αυτή είναι προσανατολισμένη σε αριθμητικές εφαρμογές και χρησιμοποιείται ευρέως από επιστήμονες και μηχανικούς. Είναι απλή και οι μεταγλωττιστές κάνουν βελτιστοποίηση, παραλληλοποίηση και διανυσματοποίηση αποτελεσματικότερα. Υπάρχουν πολλές, καλές και δοκιμασμένες βιβλιοθήκες με μαθηματικό λογισμικό από τις οποίες μερικές είναι ελεύθερα διαθέσιμες. Φυσικά, η γλώσσα αυτή υστερεί στη διεκπεραίωση πολύπλοκων διεργασιών που έχουν σχέση με το λειτουργικό σύστημα και την επεξεργασία κειμένου, αλλά το κενό καλύπτεται εύκολα με το συνδυασμό χρήσης εργαλείων του συστήματος. Επίσης, είναι απλή στη δομή της, οπότε ο αναγνώστης δεν θα δυσκολευτεί να κάνει απλούς υπολογισμούς, ακόμα και αν δεν έχει προηγούμενη εμπειρία προγραμματισμού. Τέλος, είναι μαθηματικά προσανατολισμένη: Έχει απλή, κτισμένη μέσα της, χρήση μιγαδικών αριθμών και μαθηματικών συναρτήσεων, βιβλιοθήκες διαθέσιμες για υπολογισμούς διαφορετικής ακρίβειας και αποτελεσματικότερη διαχείριση της μνήμης του υπολογιστή. Η απλότητά της και η ... ηλικία της κάνει τους αντίστοιχους μεταγλωττιστές να κάνουν την καλύτερη διαθέσιμη βελτιστοποίηση και παραλληλοποίηση του κώδικα σε σύγκριση με όλες τις άλλες γλώσσες. Οι επιστημονικές εφαρμογές συνήθως χρησιμοποιούν γλώσσα δομημένου (procedural) και όχι αντικειμενοστραφούς (object oriented) προγραμματισμού. Η Fortran έχει δυνατότητες και αντικειμενοστραφούς προγραμματισμού, αλλά συνήθως εκεί είναι πιο δημοφιλείς γλώσσες όπως οι C++/Java.

Η Fortran όπως και οι C, C++, Java είναι γλώσσες που μεταγλωττίζονται από έναν μεταγλωττιστή. Μια άλλη κατηγορία γλωσσών προγραμματισμού είναι οι ερμηνευόμενες (interpreted), όπως είναι οι perl, Basic, awk, shell programming, Macsyma, Mathematica, Matlab, Octave, Maple, .... Οι ερμηνευτές των γλωσσών αυτών ερμηνεύουν το πρόγραμμα εντολή - εντολή. Αυτό δεν επιτρέπει την ανάλυση του προγράμματος που κάνει ο μεταγλωττιστής, το οποίο είναι απαραίτητο για τη βελτιστοποίηση της απόδοσης. Οι ερμηνευόμενες γλώσσες είναι απλούστερες στη χρήση (λ.χ. με μία εντολή  $\text{Inverse}[A]$  ή  $1/A$  παίρνουμε τον αντίστροφο ενός πίνακα κάτι που χρειάζεται περισσότερη δουλειά σε μία γλώσσα όπως η Fortran, C, ...), αλλά γίνονται απαγορευτικά αργές για απαιτητικά

---

<sup>2</sup>[www.stallman.org](http://www.stallman.org)

προβλήματα. Ο χρόνος προγραμματισμού τους όμως είναι πολύ μικρότερος και ο προγραμματιστής θα πρέπει να εξετάσει αν μπορεί να λύσει το πρόβλημά του με τη βοήθειά τους, προτού αρχίσει να σχεδιάζει ένα πρόγραμμα σε μία γλώσσα όπως η Fortran.

Τέλος, αρκετές από τις εντολές του λειτουργικού συστήματος που θα συζητήσουμε παρακάτω, ερμηνεύονται έτσι μόνο από το φλοιό tcsh. Αυτή είναι μία ακόμα από τις επιλογές μας και δε θα αναλύσουμε τις διαφορές με άλλους φλοιούς έτσι ώστε η παρουσίαση να μη γίνει πολυπλοκότερη από όσο χρειάζεται.

## 1.1 Το Λειτουργικό Σύστημα

Έχετε βρεθεί στην κατάσταση να θέλετε να λύσετε ένα πρόβλημα και το πολυδιαφημισμένο και ακριβοπληρωμένο λογισμικό σας που “ψήνει και καφέ” να μην μπορεί να κάνει αυτό που αρχικά δεν προβλέψατε ότι θα ήταν αναγκαίο να γίνει; Η λύση σε αυτό το πρόβλημα είναι ένα περιβάλλον στο οποίο οι πολύπλοκες διεργασίες να καταμερίζονται σε διαφορετικά εργαλεία τα οποία επιλέγονται και συνδυάζονται με ευελιξία ανάλογα με τις ανάγκες του υπολογισμού.

Αυτή είναι η βασική φιλοσοφία των λειτουργικών συστημάτων τύπου Unix. Θεμελιώδης αρχή στο σύστημα αυτό είναι ότι όλα τα δομικά του χαρακτηριστικά είναι αρχεία, είτε πρόκειται για δεδομένα σε μορφή κειμένου, είτε εκτελέσιμα προγράμματα σε γλώσσα μηχανής, είτε σκληροί δίσκοι, εξωτερικές συσκευές, οθόνες, κάρτες ήχου ... Άρα, το πρώτο που πρέπει να κατανοήσουμε είναι η δομή του συστήματος αρχείων (filesystem).

### 1.1.1 Filesystem

Καταρχήν, σε κάθε αρχείο μας οδηγεί ένα ... μονοπάτι (path). Υπάρχουν δύο τρόποι να γράψουμε ένα path. Το σχετικό (relative) και το απόλυτο (absolute). Δύο παραδείγματα είναι:

```
bin/RungeKutta/rk.exe  
/home/george/bin/RungeKutta/rk.exe
```

Στα παραπάνω και τα δύο μπορεί να αναφέρονται στο ίδιο αρχείο, μπορεί όμως και σε διαφορετικό. Εξαρτάται “που είμαστε”. Αν “είμαστε” στον κατάλογο /home/george/, τότε αναφερόμαστε στο ίδιο αρχείο. Αν είμαστε στον κατάλογο /home/john/ ή /home/george/CompPhys/, τότε

όχι. Στις τελευταίες περιπτώσεις από το relative path γίνεται αναφορά στα αρχεία

/home/john/bin/RungeKutta/rk.exe και

/home/george/CompPhys/bin/RungeKutta/rk.exe αντίστοιχα. Πώς τα ξεχωρίζουμε; Το absolute path αρχίζει πάντα από τον χαρακτήρα /, ενώ το relative path όχι.

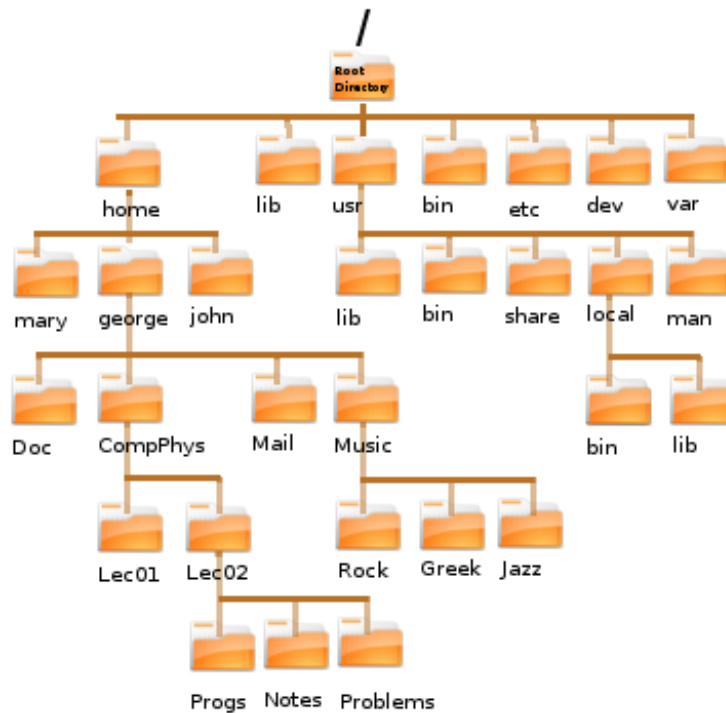
Παραπάνω, το “είμαστε” αναφέρεται σε μια θέση στο σύστημα των αρχείων που ονομάζεται “τρέχων κατάλογος” (“current directory” ή “working directory”). Σε κάθε διεργασία στο λειτουργικό σύστημα αντιστοιχεί ένας μοναδικός τρέχων κατάλογος. Το σύστημα αρχείων στο Unix είναι ενιαίο. Ακόμα και αν πρόκειται για διαφορετικούς σκληρούς δίσκους, συστήματα αρχείων που συνδέονται στον υπολογιστή μας μέσω δικτύου, το CD/DVD, ο εξωτερικός USB δίσκος, τα αρχεία-οδηγοί που αλληλεπιδρούν με το hardware (οθόνη, ποντίκι, modem, ...), όλα αναρτώνται στο ίδιο λογικά σύστημα αρχείων. Ο χρήστης/διαχειριστής έχει απόλυτη ελευθερία να τα βάλει εκεί που αυτή/ός θέλει<sup>3</sup>.

Το filesystem χτίζεται πάνω στη ρίζα του (“root”) σαν ένα ανάποδο δέντρο. Το σύμβολο του root είναι η /. Ξεκινώντας από τον κατάλογο root φτιάχνουμε καταλόγους και μέσα στους καταλόγους υποκαταλόγους κ.ο.κ. Κάθε κατάλογος χρειάζεται να γνωρίζει τον γονεϊκό του κατάλογο (“parent directory”) και τα αρχεία που περιέχει (και από αυτά μερικά μπορεί να είναι υποκατάλογοι - και αυτοί αρχεία είναι).

Όπως είπαμε στο Unix έχουμε την ελευθερία να βάλουμε τα αρχεία μας όπου θέλουμε. Ευτυχώς όμως, υπάρχουν μερικές συμβάσεις που μπορούμε να περιμένουμε ότι στα περισσότερα συστήματα θα ακολουθούνται. Έτσι, στον κατάλογο /home συνήθως βρίσκουμε τις προσωπικές περιοχές (home directories) των χρηστών, στον /etc τα αρχεία παραμετροποίησης λειτουργίας συστήματος (system configuration files), σε καταλόγους με όνομα bin τα εκτελέσιμα αρχεία των προγραμμάτων, σε καταλόγους με όνομα lib τις βιβλιοθήκες των προγραμμάτων.

Μερικές σημαντικές συμβάσεις για θέσεις στο filesystem είναι η . (τελεία = ο τρέχων κατάλογος - current directory), οι .. (δύο τελείες = ο “γονεϊκός” κατάλογος - parent directory) και η ~ (περισπωμένη = προσωπική περιοχή χρήστη - home directory). Έστω, για παράδειγμα, ότι είμαστε ο χρήστης george στον τρέχοντα κατάλογο /home/george/Music/Rock (βλ. σχήμα 1.1). Τότε τα παρακάτω paths αναφέρονται στο ίδιο αρχείο /home/george/Doc/lyrics.doc:

<sup>3</sup>Αυτό δίνει μια δυνατή αίσθηση ελευθερίας, από την άλλη είναι η αρχή του ... χάους. Ιστορικά αυτό δημιούργησε έναν πύργο της Βαβέλ για συστήματα Unix και αποτέλεσε και έναν από τους κύριους λόγους που άλλα, σαφώς κατώτερης ποιότητας, λειτουργικά συστήματα επικράτησαν στην αγορά των PC.



Σχήμα 1.1: Το filesystem στο Unix. Στην κορυφή έχουμε τη ρίζα (root directory) του συστήματος αρχείων, τον κατάλογο /. Κάθε κατάλογος περιέχει αρχεία, μεταξύ των οποίων και υποκαταλόγους. Κάθε κατάλογος έχει έναν και μοναδικό γονεϊκό κατάλογο (parent directory) που συμβολίζεται με .. (δύο τελείες). Ο / έχει για γονεϊκό κατάλογο τον εαυτό του.

```

../../../../Doc/lyrics.doc
~/Doc/lyrics.doc
~george/Doc/lyrics.doc
../../../../Doc/lyrics.doc

```

Εισάγουμε τώρα παρακάτω τις βασικές εντολές για να πλοηγούμαστε στο filesystem<sup>4</sup>. Η εντολή `cd` (change directory) αλλάζει τον τρέχοντα κατάλογο, ενώ η `pwd` (print working directory) μας αναφέρει τον τρέχοντα κατάλογο:

```
> cd /usr/bin
```

<sup>4</sup>Οι εντολές που αρχίζουν με > είναι εντολές που δίνονται από τη γραμμή εντολών και φυσικά ο αρχικός χαρακτήρας > δεν είναι μέρος της εντολής. Οι γραμμές χωρίς > είναι το κείμενο που τυπώνει η εντολή στο stdout (τερματικό).

```
> pwd
/usr/bin
> cd /usr/local/lib
> pwd
/usr/local/lib
> cd
> pwd
/home/george
> cd -
> pwd
/usr/local/lib
> cd ../../
> pwd
/usr
```

Το όρισμα της εντολής `cd` είναι ένα absolute ή relative path στο οποίο (αν είναι σωστό και έχουμε την άδεια πρόσβασης) “μεταβαίνουμε”<sup>5</sup>. Εξαιρέσεις είναι να μη δοθεί όρισμα (πάμε στο home directory) ή ο χαρακτήρας `-` (πάμε εκεί που βρισκόμασταν πριν). Η εντολή `mkdir` δημιουργεί καινούργιους καταλόγους, ενώ η `rmdir` τους σβήνει αν είναι άδειοι. Δοκιμάστε:

```
> mkdir new
> mkdir new/01
> mkdir new/01/02/03
mkdir: cannot create directory 'new/01/02/03': No such file or
    directory
> mkdir -p new/01/02/03
> rmdir new
rmdir: 'new': Directory not empty
> rmdir new/01/02/03
> rmdir new/01/02
> rmdir new/01
> rmdir new
```

Προσέξτε πως η `mkdir` δεν μπορεί να δημιουργήσει καταλόγους δύο επίπεδα πιο κάτω, ενώ η `mkdir -p` μπορεί. Ο “διακόπτης” `-p` αλλάζει τον τρόπο λειτουργίας της εντολής αυτής.

Για να δούμε τα περιεχόμενα ενός καταλόγου χρησιμοποιούμε την εντολή `ls`:

```
> ls
BE.eps  Byz.eps  Programs  srBE_xyz.eps  srB_xyz.eps
B.eps  Bzy.eps  srBd_xyz.eps  srB_xy.eps
```

---

<sup>5</sup> Δηλαδή αλλάζει τον current directory της διεργασίας.

```
> ls Programs
Backup          rk3_Byz.f90    rk3.f90
plot-commands   rk3_Bz.f90     rk3_g.f90
```

Με την πρώτη εντολή βλέπουμε τα περιεχόμενα του καταλόγου που βρισκόμαστε, ενώ στη δεύτερη (προφανώς το αρχείο Programs είναι υποκατάλογος) τα περιεχόμενα του καταλόγου που βάζουμε στην εντολή σαν όρισμα. Ένας άλλος τρόπος να δώσουμε την εντολή είναι

```
> ls -l
total 252
-rw-r--r-- 1 george users 24284 May 1 12:08 BE.eps
-rw-r--r-- 1 george users 22024 May 1 11:53 B.eps
-rw-r--r-- 1 george users 29935 May 1 13:02 Byz.eps
-rw-r--r-- 1 george users 48708 May 1 12:41 Bzy.eps
drwxr-xr-x 4 george users 4096 May 1 23:38 Programs
-rw-r--r-- 1 george users 41224 May 1 22:56 srBd_xyz.eps
-rw-r--r-- 1 george users 23187 May 1 21:13 srBE_xyz.eps
-rw-r--r-- 1 george users 24610 May 1 20:29 srB_xy.eps
-rw-r--r-- 1 george users 23763 May 1 20:29 srB_xyz.eps
```

Ο “διακόπτης” (switch) `-l` κάνει την εντολή `ls` να συμπεριφερθεί διαφορετικά. Μας δίνει τα περιεχόμενα του current directory μαζί με χρήσιμες πληροφορίες για τα αρχεία που περιέχει. Η πρώτη στήλη έχει κωδικοποιημένες τις άδειες χρήσης για κάθε αρχείο (βλ. παρακάτω). Η δεύτερη τον αριθμό των συνδέσμων (links) των αρχείων. Η τρίτη το όνομα του χρήστη (user = george) στον οποίο ανήκουν τα αρχεία. Η τέταρτη την ομάδα (group = users) του αρχείου<sup>6</sup>. Η πέμπτη το μέγεθος του αρχείου σε bytes = 8 bits. Οι επόμενες 3 τον χρόνο τελευταίας μετατροπής του αρχείου. Και τέλος, το όνομα του αρχείου.

Οι άδειες πρόσβασης `r`, `w`, `x` είναι άδειες πρόσβασης για `read`, `write`, `execute`. Όποιος έχει άδεια `r` έχει άδεια να διαβάσει και να αντιγράψει ένα αρχείο. Όποιος έχει άδεια `w` μπορεί να μεταβάλλει τα περιεχόμενα ενός αρχείου. Όποιος έχει άδεια `x` μπορεί να εκτελέσει ένα αρχείο ως πρόγραμμα<sup>7</sup>. Ειδικά για τους καταλόγους, για να μπορεί ο χρήστης/ομάδα/κόσμος να “μπει” σε έναν κατάλογο με την εντολή `cd` πρέπει να έχει άδεια `x`. Για να μπορέσει να σβήσει ένα αρχείο πρέπει να έχει άδεια `w` στον κατάλογο που ανήκει.

<sup>6</sup>Ένας χρήστης μπορεί να ανήκει σε πολλές ομάδες για να διευκολύνεται η συνεργασία με διαφορετικές ομάδες χρηστών. Φυσικά κάθε ομάδα μπορεί να έχει πολλούς χρήστες για μέλη.

<sup>7</sup>Φυσικά το αν θα μπορέσει να εκτελεστεί σωστά είναι ευθύνη του χρήστη

Οι άδειες χωρίζονται σε τρεις ομάδες: Ο ιδιοκτήτης του αρχείου (user- θέσεις 2-4), η ομάδα (group- θέσεις 5-7) και ο υπόλοιπος κόσμος (others-θέσεις 8-10). Παραδείγματος χάρη

```
-rw-r--r--
-rwxr-----
drwx--x--x
```

Στην πρώτη περίπτωση, ο ιδιοκτήτης έχει άδεια read, write, αλλά όχι execute και η ομάδα/κόσμος έχει μόνο άδεια read. Στη δεύτερη, ο χρήστης έχει άδεια read, write, execute, η ομάδα άδεια read και ο κόσμος τίποτα. Στην τρίτη, ο χρήστης έχει άδεια read, write, execute, η ομάδα/κόσμος άδεια execute. Ειδικά στην τρίτη βρίσκουμε το χαρακτηριστήρα d στην πρώτη θέση που δηλώνει ότι το αρχείο είναι κατάλογος (directory). Η πρώτη αυτή θέση, όταν είναι “κατελιημμένη”, δηλώνει αρχείο ειδικού τύπου.

Οι άδειες πρόσβασης αλλάζουν με την εντολή chmod:

```
> chmod u+x file
> chmod og-w file1 file2
> chmod a+r file
```

Με την πρώτη εντολή ο ιδιοκτήτης ( $u \equiv \text{user}$ ) παίρνει (+) άδεια x στο αρχείο file. Με τη δεύτερη, ο κόσμος ( $o \equiv \text{others}$ ) και η ομάδα ( $g \equiv \text{group}$ ) χάνουν (-) άδεια w, ενώ στην τρίτη όλοι ( $a \equiv \text{all}$ ) αποκτούν άδεια πρόσβασης r.

Τελειώνουμε την παράγραφο αυτή αναφέροντας μερικές ακόμα βασικές εντολές που αναφέρονται στη διαχείριση των αρχείων. Η εντολή cp (copy) φτιάχνει αντίγραφα αρχείων:

```
> cp file1.f90 file2.f90
> cp file1.f90 file2.f90 file3.f90 Programs
```

Η πρώτη εντολή αντιγράφει τα δεδομένα του αρχείου file1.f90 σε ένα καινούργιο αρχείο file2.f90, αν αυτό δεν υπάρχει ήδη, ή αντικαθιστά το αρχείο file2.f90 από ένα καινούργιο με τα περιεχόμενα του file1.f90. Η δεύτερη αντιγράφει τα αρχεία file1.f90 file2.f90 file3.f90 στον κατάλογο Programs (αν δεν είναι κατάλογος εισπράτουμε ... παράπονα).

Η εντολή mv (move) “μετακινεί” ή μετονομάζει αρχεία:



```
> mv file1.f90 file2.f90
> mv file1.f90 file2.f90 file3.f90 Programs
```

Η πρώτη εντολή έχει ως αποτέλεσμα να μετονομάσει το αρχείο `file1.f90` σε `file2.f90`. Η δεύτερη εντολή μετακινεί τα αρχεία `file1.f90` `file2.f90` `file3.f90` στον κατάλογο `Programs`.

Τέλος, η εντολή `rm` (remove) διαγράφει αρχεία<sup>8</sup>. Η εντολή αυτή δε “χαρίζει κάστανά”. Όταν το αρχείο διαγράφεται, το λειτουργικό σύστημα δεν μπορεί να το επαναφέρει. Προσοχή λοιπόν

```
> ls
file1.f90 file2.f90 file3.f90 file4.csh
> rm file1.f90 file2.f90 file3.f90
> ls
file4.csh
```

τα αρχεία `file1.f90` `file2.f90` `file3.f90` δεν υπάρχουν πια για το λειτουργικό σύστημα<sup>9</sup>. Για να είμαστε πιο προσεκτικοί μπορούμε να χρησιμοποιήσουμε τον διακόπτη `-i`. Τότε η εντολή ζητάει επιβεβαίωση πριν την καταστροφή:

```
> rm -i *
rm: remove regular file 'file1.f90'? y
rm: remove regular file 'file2.f90'? y
rm: remove regular file 'file3.f90'? y
rm: remove regular file 'file4.csh'? n
> ls
file4.csh
```

Στην τελευταία γραμμή απαντήσαμε αρνητικά και έτσι το αρχείο `file4.csh` δεν διαγράφηκε.

Η εντολή `rm` δε διαγράφει καταλόγους. Χρησιμοποιήστε την εντολή `rmdir` για τη διαγραφή άδειων καταλόγων. Για να διαγράψετε καταλόγους με περιεχόμενα χρησιμοποιήστε την εντολή<sup>10</sup> `rm -r`. Λ.χ. έστω ότι έχουμε στους καταλόγους `dir1` και `dir1/dir2` τα αρχεία:

<sup>8</sup>Στην πραγματικότητα αφαιρεί “links” (συνδέσεις στο filesystem μίας διαμέρισης-partition) αρχείων. Ένα αρχείο μπορεί να έχει ένα ή περισσότερα links στην ίδια διαμέριση ενός filesystem. Ένα αρχείο θεωρείται διαγραμμένο, όταν αφαιρεθούν όλα τα links του.

<sup>9</sup>Αυτό δε σημαίνει ότι τα δεδομένα τους δεν υπάρχουν πια στο δίσκο... Καταρχήν τα αρχεία μπορεί να έχουν παραπάνω από ένα link οπότε είναι προσβάσιμα από τα άλλα links. Ακόμα και αν αφαιρεθούν όλα τα links είναι δυνατόν κάποιος/α να ανακτήσει (μέρος του) αρχείου με εξειδικευμένα εργαλεία.

<sup>10</sup>Ένα `rm -r *` και τα δεδομένα σας αποτελούν ιστορία...

```
./dir1
./dir1/file2.f90
./dir1/file1.f90
./dir1/dir2
./dir1/dir2/file3.f90
```

Οι εντολές

```
> rm dir1
rm: cannot remove 'dir1': Is a directory
> rm dir1/dir2
rm: cannot remove 'dir1/dir2': Is a directory
> rmdir dir1
rmdir: dir1: Directory not empty
> rmdir dir1/dir2
rmdir: dir1/dir2: Directory not empty
> rm -r dir1
```

Με την τελευταία εντολή όλα τα παραπάνω αρχεία διαγράφονται. Εναλλακτικά, πρώτα αδειάζουμε τους καταλόγους από τα αρχεία και μετά τους διαγράφουμε με `rmdir`:

```
> cd dir1/dir2; rm file3.f90
> cd .. ; rmdir dir2
> rm file1.f90 file2.f90
> cd .. ; rmdir dir1
```

Παρατηρήστε ότι το ελληνικό ερωτηματικό (semicolon = “;”) χωρίζει εντολές οι οποίες εκτελούνται η μία μετά την άλλη.

### 1.1.2 Εντολές

Οι εντολές στο Unix είναι, όπως είπαμε, αρχεία με άδεια πρόσβασης x (execute). Όταν στη γραμμή εντολών γράψουμε μία πρόταση λ.χ.

```
> ls -l test.f90 test.dat
```

ο φλοιός (το πρόγραμμα με το οποίο ο χρήστης αλληλεπιδρά με το λειτ. σύστημα) την ερμηνεύει ως εξής: Η πρόταση χωρίζεται σε λέξεις και η πρώτη λέξη (`ls`) ερμηνεύεται ως εντολή. Οι υπόλοιπες περνάνε στην εντολή ως τα ορίσματά της. Κατά σύμβαση, λέξεις που αρχίζουν από τον χαρακτήρα - (λ.χ. `-l`, `--help`, `--version`, `-03`) έχουν συνήθως ειδική ερμηνεία και ονομάζονται “διακόπτες” (options, switches) και

κάνουνε το πρόγραμμα να εκτελείται με διαφορετικό τρόπο ανάλογα με τις τιμές τους. Είδαμε ήδη τη διαφορά με το πρόγραμμα `ls` που ανάλογα με το αν το καλούμε ως `ls` ή `ls -l`, τα αποτελέσματα τυπώνονται με διαφορετικό τρόπο.

Για να εκτελεστεί η εντολή `ls` ο φλοιός αναζητεί ένα αρχείο με το όνομα `ls` που να έχει άδεια πρόσβασης `x`. Για να καταλάβουμε πώς γίνεται η αναζήτηση αυτή πρέπει να εξηγήσουμε τι είναι οι μεταβλητές φλοιού και οι μεταβλητές περιβάλλοντος. Αυτές έχουν ένα όνομα που δίνεται από μια ακολουθία χαρακτήρων και οι τιμές τους λαμβάνονται προτάσσοντας τον χαρακτήρα `$` στο όνομά τους. Έτσι η μεταβλητή με το όνομα `PATH` έχει τιμή `$PATH`. Οι τιμές των μεταβλητών περιβάλλοντος τίθενται με την εντολή<sup>11</sup> `setenv` για τις μεταβλητές περιβάλλοντος και με την εντολή `set` για τις μεταβλητές φλοιού:

```
> setenv MYVAR test-env
> set myvar = test-shell
> echo $MYVAR $myvar
test-env test-shell
```

Δύο μεταβλητές των οποίων αναλαμβάνει ο φλοιός να τις ορίσει σωστά στο περιβάλλον του χρήστη είναι οι `PATH` και `path`:

```
>echo $path
/usr/local/bin /usr/bin /bin /usr/X11/bin
>echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11/bin
```

Βλέπουμε ότι η τιμή τους (που ο χρήστης μπορεί να αλλάξει!) αποτελείται από συνιστώσες που είναι διαδρομές στο σύστημα αρχείων. Στην πρώτη περίπτωση οι συνιστώσες χωρίζονται από κενό, ενώ στη δεύτερη από `:` (άνω-κάτω τελεία).

Έτσι, επιστρέφοντας στην ερώτηση πώς βρίσκει ο φλοιός την εντολή `ls`, θα είναι ήδη φανερό πως ψάχνει κάθε συνιστώσα της τιμής της μεταβλητής `path` μέχρι να τη βρει. Αν είστε περίεργοι, δώστε τις εντολές

```
> which ls
/bin/ls
> ls -l /bin/ls
-rwxr-xr-x 1 root root 93560 Sep 28 2006 /bin/ls
```

---

<sup>11</sup>Η εντολή `setenv` είναι ειδική για το φλοιό `tcsh`. Στο φλοιό `bash` αρκεί να καθορίσετε την τιμή με ένα `=`: π.χ. `MYVAR=test-env`.

από όπου είναι προφανές ότι το ζητούμενο αρχείο είναι το `/bin/ls`. Αν η διαδικασία αποτύχει, ο φλοιός δίνει μήνυμα σφάλματος. Αν πετύχει, το πρόγραμμα φορτώνεται από το λειτουργικό σύστημα στη μνήμη για εκτέλεση. Τα ορίσματα περνάνε στην εντολή, ώστε αυτή να τα ερμηνεύσει όπως έχει προγραμματιστεί. Στην εντολή

```
> ls -l test.f90 test.dat
```

το όρισμα `-l` είναι διακόπτης που ερμηνεύεται από την εντολή να δώσει long listing των αρχείων. Τα ορίσματα `test.f90` και `test.dat` ερμηνεύονται από την εντολή ως τα αρχεία που θα αναζητήσει για να μας δώσει πληροφορίες.

Μία σημαντική πληροφορία στην ερμηνεία των ορισμάτων είναι η χρήση “μπαλαντέρ” (wildcard):

```
> ls -l *.f90 *.dat
```

θα κάνει τον φλοιό να αναπτύξει τα αστεράκια πριν να περάσει τα ορίσματα στο πρόγραμμα σε οποιαδήποτε ακολουθία χαρακτήρων δίνει ένα υπάρχον αρχείο. Έτσι, αν ο κατάλογος που βρισκόμαστε περιέχει τα αρχεία `test.f90`, `test1.f90`, `myprog.f90`, `test.dat`, `hello.dat`, η εντολή που θα “δει” το λειτουργικό είναι

```
> ls -l myprog.f90 test1.f90 test.f90 hello.dat test.dat
```

Αυτό συμβαίνει για οποιαδήποτε άλλη εντολή.

Σε κάθε εντολή συναρτάται η καθιερωμένη είσοδος `stdin` (standard input), η καθιερωμένη έξοδος `stdout` (standard output) και η καθιερωμένη έξοδος σφαλμάτων `stderr` (standard error). Αυτές είναι συμβάσεις για αρχεία στα οποία το πρόγραμμα μπορεί να διαβάζει ή να τυπώνει δεδομένα. Όταν ο χρήστης δουλεύει σε ένα τερματικό, όλες οι παραπάνω θεωρούνται αρχικά ότι είναι το τερματικό<sup>12</sup>. Δηλ. μια εντολή που διαβάζει δεδομένα από το `stdin`, αυτά ο χρήστης θα τα εισάγει μέσω του τερματικού τυπώνοντάς τα με το πληκτρολόγιο. Αν μια εντολή τυπώνει στο `stdout` ή στο `stderr`, αυτά τυπώνονται στο τερματικό.

Η δυνατότητα που δίνει μεγάλη ευελιξία στον χρήστη να χειριστεί τις εντολές είναι η δυνατότητα επαναορισμού των παραπάνω αρχείων. Ο χρήστης μπορεί να τα ορίσει να είναι οποιοδήποτε αρχείο. Ο επαναορισμός του `stdout` γίνεται με το σύμβολο `>`.

<sup>12</sup>Σας θυμίζουμε ότι για το Unix τα πάντα είναι αρχεία.

```
> ls
file1.f90  file2.f90  file3.f90  file4.csh
> ls > results
> ls
file1.f90  file2.f90  file3.f90  file4.csh  results
```

Στην πρώτη εντολή βλέπουμε τα περιεχόμενα του καταλόγου. Στη δεύτερη επαναορίζουμε το stdout να είναι το αρχείο results. Μετά την εκτέλεση της εντολής παρατηρούμε τη δημιουργία του αρχείου results το οποίο περιέχει σαν δεδομένα τα ονόματα των αρχείων file1.f90 file2.f90 file3.f90 file4.csh. Αν το αρχείο results δεν υπάρχει, δημιουργείται, αν υπάρχει, τα περιεχόμενα του καταστρέφονται και αντικαθίστανται από το stdout της εντολής. Για να επισυνάψουμε (append) τα δεδομένα του stdout στο τέλος ενός ήδη υπάρχοντος αρχείου, χρησιμοποιούμε το σύμβολο `>>`. Έτσι, αν μετά από τις παραπάνω εντολές εκτελέσουμε

```
> ls >> results
```

τότε τα περιεχόμενα του αρχείου results θα είναι

```
file1.f90  file2.f90  file3.f90  file4.csh
file1.f90  file2.f90  file3.f90  file4.csh  results
```

Ο επαναορισμός του stdin γίνεται με το σύμβολο `<`, ενώ του stderr με το σύμβολο `>&`<sup>13</sup>. Σχετικά παραδείγματα θα δούμε στην παράγραφο 1.2.

Είναι δυνατόν το stdin/stdout μιας εντολής να οριστεί να είναι το stdout/stdin μιας άλλης εντολής. Με τον τρόπο αυτό μπορούν να συνδυαστούν οι λειτουργίες διαφορετικών εντολών, έτσι ώστε να παράγουν αποτελέσματα για τα οποία θα χρειαζόταν να γράψουμε ένα αρκετά πολύπλοκο πρόγραμμα για να τα πάρουμε. Η διαδικασία αυτή λέγεται “διασωλήνωση” (pipng) και χρησιμοποιείται κυρίως για τη δημιουργία ισχυρών φίλτρων. Για τον σκοπό αυτό χρησιμοποιείται το σύμβολο `|`

```
> cmd1 | cmd2 | cmd3 | ... | cmdN
```

Με την παραπάνω πρόταση το stdout της εντολής cmd1 γίνεται stdin της εντολής cmd2, το stdout της εντολής cmd2 γίνεται stdin της εντολής

<sup>13</sup>Το `>&` ισχύει μόνο για το φλοιό tcsh. Για άλλους φλοιούς (bash, sh, ...) διαβάστε τη σχετική βοήθεια.

cmd3 κοκ. Σχετικά παραδείγματα θα δούμε στη παράγραφο 1.2.

### 1.1.3 Αναζητώντας Βοήθεια

Το Unix απέκτησε τη φήμη λειτουργικού συστήματος μη φιλικού προς τον χρήστη. Τίποτα δεν απέχει περισσότερο από την πραγματικότητα. Παρόλο που έχει μια αρχική δυσκολία, η οποία λύνεται αν ο χρήστης μεθοδικά διαβάσει και εξασκηθεί στις βασικές εντολές του συστήματος, στη συνέχεια, όλες οι πληροφορίες για να κάνει ο χρήστης οτιδήποτε είναι διαθέσιμες online<sup>14</sup>.

Το κλειδί για άνετη πλεύση σε αυτό το ταξίδι είναι να μάθει ο χρήστης να χρησιμοποιεί το σύστημα βοήθειας που παρέχεται εντός και εκτός συστήματος. Οι περισσότερες εντολές παρέχουν βασικές πληροφορίες από μόνες τους. Από τη γραμμή εντολών, για τυχαία εντολή cmd δοκιμάστε:

```
> cmd --help  
> cmd -h  
> cmd -help  
> cmd -\?
```

Για παράδειγμα, δώστε την εντολή `ls --help`. Αν είναι εφαρμογή παραθυρική, αρχίστε από το σχεδόν πάντοτε διαθέσιμο menu “Help”. Μη φοβηθείτε να διαβάσετε...

Ας υποθέσουμε πως έχουμε ακούσει κάτι για μια εντολή που λέγεται `printf` ή κάτι τέτοιο τέλος πάντων. Το πρώτο σύστημα βοήθειας είναι τα man pages. Αυτό είναι ένα σύστημα από help files που τα αναζητούμε με την εντολή `man`:

```
> man printf
```

Η εντολή `info` δίνει περισσότερες πληροφορίες σε μορφή “βιβλίου” με βασικές δυνατότητες ξεφυλλίσματος (browsing).

```
> info printf
```

οι εντολές

---

<sup>14</sup>Σε αντίθεση με άλλα δημοφιλή λειτουργικά συστήματα τα οποία είναι “μαύρα κουτιά”.

```
> man -k printf
> whatis printf
```

μας πληροφορούν ότι υπάρχουν και άλλες, πιθανώς σχετιζόμενες εντολές fprintf, fwprintf, wprintf, sprintf... Ειδικά το αποτέλεσμα της δεύτερης το παραθέτουμε γιατί είναι διδακτικό:

```
> whatis printf
printf          (1)  - format and print data
printf          (1p) - write formatted output
printf          (3)  - formatted output conversion
printf          (3p) - print formatted output
printf [builtins] (1) - bash built-in commands, see bash↵
(1)
```

Η δεύτερη στήλη είναι το “τμήμα” (section) των man pages στο οποίο αναφέρεται η εντολή. Η πρόσβαση στα τμήματα γίνεται δίνοντας το σαν όρισμα στην εντολή:

```
> man 1 printf
> man 1p printf
> man 3 printf
> man 3p printf
> man bash
```

δίνει πρόσβαση στις αντίστοιχες πληροφορίες. Στο τμήμα ένα βρίσκουμε το printf ως “κοινή εντολή”, στο τμήμα 3 ως συνάρτηση της γλώσσας C. Άλλα τμήματα είναι το 2 (εντολές διαχείρισης συστήματος), 4, 5, 8 κλπ. Περιηγηθείτε στον κατάλογο /usr/share/man/ για να δείτε με τα μάτια σας περισσότερα.

Δίνοντας την εντολή

```
> printf --help
```

παίρνουμε πάλι αρκετή πληροφορία. Η εντολή

```
> locate printf
```

μας δείχνει πολλά σχετικά αρχεία στο σύστημα. Οι εντολές

```
> which printf
> where printf
```

μας δίνουν πληροφορία για το πού βρίσκονται τα αρχεία-προγράμματα που εκτελούνται, όταν δίνεται η εντολή `printf`.

Μια άλλη σημαντική ευκολία που μας προσφέρει ο φλοιός είναι η “συμπλήρωση εντολών”. Μπορούμε να γράψουμε μέρος του ονόματος μιας εντολής και να πατήσουμε τον συνδυασμό πλήκτρων `[Ctrl-d]`<sup>15</sup> (δηλ. ταυτόχρονα το πλήκτρο `Ctrl` και το πλήκτρο `d`). Τότε ο φλοιός θα μας συμπληρώσει όλες τις εντολές των οποίων το όνομα αρχίζει με τα γράμματα που έχουμε ήδη γράψει<sup>16</sup>:

```
> pri[Ctrl-d]
printafm      printf      printenv      printnodetest
```

Δοκιμάστε λ.χ. την εντολή `x[Ctrl-d]` και θα μάθετε (σχεδόν) τα πάντα για τις εντολές διαθέσιμες στο παραθυρικό σύστημα X: `xterm`, `xeyes`, `xclock`, `xcalc`, ....

Τέλος, μεγάλη πηγή πληροφοριών είναι το διαδίκτυο. Google your blues... και θα εκπλαγείτε πόσοι άλλοι έχουν ασχοληθεί με το πρόβλημά σας.

## 1.2 Εργαλεία Επεξεργασίας Κειμένου – Φίλτρα

Για την ανάλυση των δεδομένων που θα παράγουμε χρειαζόμαστε εργαλεία τα οποία να επεξεργάζονται ευέλικτα τα αρχεία κειμένου<sup>17</sup>. Μερικά εργαλεία που μπορούν να φτιάξουν περίπλοκα και ισχυρά φίλτρα είναι τα προγράμματα `cat`, `less`, `head`, `tail`, `grep`, `sort` και `awk`. Ας αναφέρουμε και τα προγράμματα `perl` και `sed` για τον αναγνώστη που ενδιαφέρεται να πλουτίσει το οπλοστάσιό του, παρόλο που δε θα τα περιγράψουμε εδώ λόγω χώρου.

Ας υποθέσουμε ότι έχουμε το αρχείο με δεδομένα με όνομα `data`<sup>18</sup> με τα περιεχόμενα μιας αποθήκης τροφίμων και το κοστολόγιό τους:

```
bananas 100 pieces 1.45
```

<sup>15</sup>Στο φλοιό `bash` πατήστε ένα ή δύο `[Tab]`.

<sup>16</sup>Με τον ίδιο τρόπο γίνεται και συμπλήρωση ονομάτων αρχείων. Γράψτε μερικώς το όνομα ενός αρχείου στο όρισμα μιας εντολής και πατήστε `[Tab]` ή `[Ctrl-d]`.

<sup>17</sup>Εδώ εννοούμε αρχεία που αποτελούνται από εκτυπώσιμους χαρακτήρες ASCII ή UTF. Δεν εννοούμε τα μορφοποιημένα αρχεία κειμένου - έγγραφα - τύπου `.odt`, `.doc` κλπ.

<sup>18</sup>Μπορείτε να βρείτε το αρχείο στο συνοδευτικό λογισμικό.



apples	325	boxes	1.18
pears	34	kilos	2.46
bread	62	kilos	0.60
ham	85	kilos	3.56

Η εντολή

```
> cat data
```

απλά τυπώνει τα περιεχόμενα στο stdout. Η εντολή παίρνει τα αρχεία από το όρισμα της εντολής ή αν δε δοθούν, το stdin και τυπώνει τα περιεχόμενά τους στο stdout. Αφού αυτά μπορεί να επαναοριστούν, η εντολή

```
> cat < data > data1
```

παίρνει τα περιεχόμενα του αρχείου data από το stdin και τα τυπώνει στο stdout που εδώ έχει επαναοριστεί να είναι το αρχείο data1. Η εντολή έχει ισοδύναμο αποτέλεσμα με την

```
> cp data data1
```

Η εντολή

```
> cat data data1 > data2
```

τυπώνει πρώτα τα περιεχόμενα του data και μετά του data1 μέσα στο αρχείο data2.

Η εντολή

```
> less gfortran.txt
```

τυπώνει στο stdout τα περιεχόμενα του data σελίδα-σελίδα. Πατήστε [space] για να προχωρήσετε μια σελίδα, [b] για να γυρίσετε πίσω μια σελίδα και τα πάνω/κάτω βελάκια για να προχωρήσετε μια γραμμή. Με [g] πάτε στην αρχή του αρχείου και με [G] στο τέλος. Με [h] παίρνετε βοήθεια και με [q]... αναχωρείτε (quit).

Με τις εντολές

```
> head -n 1 data  
bananas 100 pieces 1.45
```

```
> tail -n 2 data
bread      62 kilos   0.60
ham        85 kilos   3.56
> tail -n 2 data | head -n 1
bread      62 kilos   0.60
```

παίρνουμε την πρώτη γραμμή του αρχείου data, τις δύο τελευταίες και την δεύτερη από το τέλος αντίστοιχα. Προσέξτε πώς με piping των δύο εντολών τις συνδυάσαμε για να φτιάξουμε το φίλτρο “τύπωσε τη δεύτερη γραμμή από το τέλος”.

Η εντολή sort τυπώνει τα περιεχόμενα του αρχείου κατά αύξουσα διάταξη των γραμμών, όπου η σύγκριση γίνεται χαρακτήρα-χαρακτήρα (όχι αριθμητικά):

```
> sort data
apples    325 boxes   1.18
bananas   100 pieces 1.45
bread      62 kilos   0.60
ham        85 kilos   3.56
pears      34 kilos   2.46
```

Για αντίστροφη διάταξη δοκιμάστε την εντολή sort -r data. Για να διατάξουμε τα περιεχόμενα συγκρίνοντας τους αριθμούς στη δεύτερη στήλη χρησιμοποιούμε τον διακόπτη -k 2 (=δεύτερη στήλη) και -n (=αριθμητική – numerical – διάταξη):

```
> sort -k 2 -n data
pears      34 kilos   2.46
bread      62 kilos   0.60
ham        85 kilos   3.56
bananas   100 pieces 1.45
apples    325 boxes   1.18
```

Αν αμελήσω τον διακόπτη -n οι γραμμές συγκρίνονται με βάση τους χαρακτήρες της λέξης στη δεύτερη στήλη:

```
> sort -k 2 data
bananas   100 pieces 1.45
apples    325 boxes   1.18
pears      34 kilos   2.46
bread      62 kilos   0.60
ham        85 kilos   3.56
```

Η τελευταία στήλη έχει αριθμούς με υποδιαστολή (όχι ακραίους). Για

να κάνουμε τη διάταξη με βάση την αξία τέτοιων αριθμών βάζουμε το διακόπτη `-g`:

```
> sort -k 4 -g data
bread      62 kilos  0.60
apples     325 boxes 1.18
bananas    100 pieces 1.45
pears      34 kilos  2.46
ham        85 kilos  3.56
```

Η εντολή `grep` αναλύει ένα αρχείο κειμένου γραμμή-γραμμή αναζητώντας μια ακολουθία χαρακτήρων που έχουμε ζητήσει. Κάθε τέτοια γραμμή που βρίσκει την τυπώνει στο `stdout`:

```
> grep kilos data
pears      34 kilos  2.46
bread      62 kilos  0.60
ham        85 kilos  3.56
```

τυπώνει κάθε γραμμή που έχει το “kilos”. Αν θέλουμε να τυπώνει κάθε γραμμή που δεν περιέχει το `kilos`, προσθέτουμε τον διακόπτη `-v`:

```
> grep -v kilos data
bananas    100 pieces 1.45
apples     325 boxes  1.18
```

Η ακολουθία χαρακτήρων που αναζητούμε μπορεί να είναι ένα `regular expression`. Για να περιγράψουμε πλήρως τα θηρία αυτά, θέλουμε μισό βιβλίο... Μερικά παραδείγματα:

```
> grep ^b data
bananas    100 pieces 1.45
bread      62 kilos  0.60
> grep '0$' data
bread      62 kilos  0.60
> grep '3[24]' data
apples     325 boxes  1.18
pears      34 kilos  2.46
```

Η πρώτη τυπώνει τις γραμμές που αρχίζουν από `b` (αγνοεί την 2η γραμμή), η δεύτερη αυτές που τελειώνουν σε `0` (αγνοεί την πρώτη γραμμή) ενώ η τρίτη γραμμές που περιέχουν τις ακολουθίες χαρακτήρων `32` ή `34` (αγνοεί την τελευταία γραμμή).

Το πιο δυνατό όμως εργαλείο για ανάλυση είναι το πρόγραμμα `awk`. Στην πιο απλή του χρήση, αναλύει το κείμενο του αρχείου γραμμή-γραμμή και ορίζει μεταβλητές `$1`, `$2`, ... στις οποίες αποθηκεύει την τιμή της πρώτης, δεύτερης, ... λέξης της γραμμής. Στη μεταβλητή `$0` αποθηκεύει όλη τη γραμμή, ενώ η μεταβλητή `NF` μετράει τον αριθμό των λέξεων στη γραμμή. Η μεταβλητή `NR` μετράει τις γραμμές που έχει επεξεργαστεί μέχρι στιγμής.

Ένα πρόγραμμα `awk` μπορεί να γραφτεί στη γραμμή εντολών. Είναι εντολές που περικλείονται ανάμεσα σε αγκύλες `{ ... }` και εκτελούνται για κάθε γραμμή του αρχείου. Ειδική περίπτωση αποτελούν οι εντολές που γράφονται μέσα στο κατασκευάσμα `BEGIN{ ... }` και `END{ ... }` που είναι εντολές που εκτελούνται μια φορά πριν την επεξεργασία και μετά την επεξεργασία των γραμμών του αρχείου. Για παράδειγμα η εντολή:

```
> awk '{ print $1,"total value= ",$2*$4}' data
bananas total value= 145
apples total value= 383.5
pears total value= 83.64
bread total value= 37.2
ham total value= 302.6
```

τυπώνει το είδος (1η στήλη= `$1`) και την συνολική αξία του: ποσότητα (2η στήλη= `$2`)  $\times$  αξία μονάδας (4η στήλη= `$4`). Άλλα παραδείγματα είναι

```
> awk '{ value += $2*$4}END{ print "Total= ",value}' data
Total= 951.94
> awk '{ av += $4}END{ print "Average Price= ",av/NR}' data
Average Price= 1.85
> awk '{ print $2^2 * sin($4) + exp(-$4)}' data
```

Στην πρώτη εντολή υπολογίζουμε τη συνολική αξία των προϊόντων: Σε κάθε γραμμή προσθέτουμε (`+=`) στη μεταβλητή `value` την συνολική αξία του προϊόντος. Στο τέλος (`END{ ... }`) τυπώνουμε το άθροισμα που συσσωρεύσαμε στο τέλος του αρχείου. Η δεύτερη εντολή τυπώνει τη μέση τιμή των τιμών. Με τον ίδιο τρόπο προσθέτουμε στη μεταβλητή `av` την τιμή κάθε προϊόντος (2η στήλη= `$2`) και στο τέλος, τυπώνουμε το σύνολο δια τον αριθμό των προϊόντων (`=αρ. γραμμών = NR`). Η τελευταία εντολή κάνει μια αυθαίρετη αριθμητική πράξη: Τυπώνει το τετράγωνο της δεύτερης στήλης επί το ημίτονο της τέταρτης και προσθέτει το εκθετικό της -4ης στήλης.

Οι δυνατότητες των παραπάνω εργαλείων δεν εξαντλούνται σε ένα μικρό κεφάλαιο. Διαβάστε τις man και info pages και θα μάθετε να τις κάνετε να ψήνουν και ... καφέ!

## 1.3 Προγραμματίζοντας με τον Emacs

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα θεμελιώδη για τη χρήση ενός editor<sup>19</sup> για προγραμματισμό. Για έναν προγραμματιστή που προγραμματίζει αρκετές ώρες κάθε μέρα, το περιβάλλον και τα εργαλεία επεξεργασίας του κειμένου των εντολών προγραμματισμού καθορίζουν κατά ένα σημαντικό ποσοστό τη συνολική ...ποιότητα της ζωής του/της. Και όπως βλέπετε είμαστε αρκετά προσεκτικοί στη διατύπωση: Δε μιλάμε για προγράμματα επεξεργασίας κειμένου *μορφοποιημένων* εγγράφων (λ.χ. Open Office) που δίνουν έμφαση στη φόρμα του κειμένου, αλλά για επεξεργαστές απλού κειμένου που αποτελείται από σκέτους (χωρίς φόρμα) χαρακτήρες που “διαβάζονται” (“printable characters” σε αντίθεση με τους “non printable characters”). Παραδείγματα απλών τέτοιων επεξεργαστών στο Linux είναι οι επεξεργαστές gedit, vi, pico, nano κλπ που θα μπορούσε κανείς να χρησιμοποιήσει εναλλακτικά για την επεξεργασία του κώδικα στα προγράμματα που παρουσιάζουμε στο μάθημα. Με αυτούς μπορεί κάποιος εύκολα να επεξεργαστεί απλά προγράμματα έχοντας βασικές λειτουργίες επεξεργασίας κειμένου (editing). Υπάρχουν λειτουργίες σε έναν επεξεργαστή κειμένου που κάνει τον προγραμματισμό ανετότερο και βοηθά στην ... υγιεινή κρατώντας μακριά τα ενοχλητικά ... έντομα! Λ.χ. η αναγνώριση από τον επεξεργαστή των εντολών της γλώσσας προγραμματισμού, των μεταβλητών και των δομικών στοιχείων επιτρέπει την “όμορφη” παρουσίασή τους με κατάλληλο χρωματισμό ή/και font, επισημαίνει σφάλματα όταν δεν κλείνουν παρενθέσεις ή οι εντολές δεν μπαίνουν στο σωστό σημείο στο αρχείο του προγράμματος κλπ. Ένας “πολύγλωσσος” και “πολυμορφικός” επεξεργαστής κειμένου με πολλές δυνατότητες και ευκολίες για τον προγραμματιστή είναι ο GNU Emacs editor<sup>20</sup>. Ο Emacs είναι ανοιχτό λογισμικό, διατίθεται ελεύθερα και μπορεί να εγκατασταθεί σε λειτουργικό σύστημα Linux, Mac και MS Windows. Ο χρήστης

<sup>19</sup>editor= πρόγραμμα επεξεργασίας κειμένου

<sup>20</sup><http://www.gnu.org/software/emacs/> (main site), <http://www.emacswiki.org/> (expert tips), <http://en.wikipedia.org/wiki/Emacs> (general info)

μπορεί να τον προγραμματίσει<sup>21</sup> να εκτελεί απλές, αλλά και σύνθετες λειτουργίες της αρεσκείας του/της, καθώς και να έχει μια σχεδόν ολοκληρωμένη αλληλεπίδραση με το λειτουργικό σύστημα και πολλές από τις εφαρμογές που βρίσκονται σε αυτό. Ο πιο προχωρημένος χρήστης μπορεί λ.χ. να επεξεργαστεί ένα αρχείο σε γλώσσα Fortran, να το μεταγλωττίσει και να το διορθώσει με τη βοήθεια του debugger δίνοντας εντολές μέσα από τον Emacs.

Για τον προγραμματισμό πολύπλοκων προγραμμάτων με πολλές χιλιάδες γραμμές κώδικα και πολύπλοκο συσχετισμό διεργασιών είναι συνηθισμένο να χρησιμοποιούνται εξειδικευμένα περιβάλλοντα προγραμματισμού. Αυτά προσφέρουν στον προγραμματιστή ολοκληρωμένες λύσεις για τον προγραμματισμό σε μια γλώσσα (λ.χ. C++, Java κλπ) ενσωματώνοντας σε ένα απλό interface και τις λειτουργίες μεταγλωττισμού, debugging, βοήθειας κλπ. Το μειονέκτημα σε αυτά είναι η εξειδίκευση που περιορίζει την ελευθερία του προγραμματιστή ως προς την επιλογή γλώσσας, βιβλιοθηκών, λειτουργικού συστήματος και συνήθως έχουν ακριβές άδειες χρήσης. Είναι, επίσης, δύσκολη η μεταφορά των εργασιών ενός προγραμματιστή από έναν υπολογιστή σε έναν άλλο και φυσικά η επεξεργασία του προγράμματος από διαφορετικά περιβάλλοντα προγραμματισμού. Η πολύπλοκη και εξειδικευμένη παραμετροποίησή τους συνήθως “δένει” τον προγραμματιστή και το πρόγραμμα με το συγκεκριμένο πακέτο περιβάλλοντος προγραμματισμού.

### 1.3.1 Καλώντας τον Emacs

Στη γραμμή εντολών πληκτρολογήστε:

```
> emacs &
```

Προσέξτε τον χαρακτήρα & στο τέλος της εντολής. Χωρίς αυτόν το prompt του φλοιού δεν επιστρέφει και δεν μπορούμε να δώσουμε άλλη εντολή από τον φλοιό. Με αυτόν η εντολή (όπως και κάθε εντολή την οποία τελειώνουμε με το &) πάει στο “υπόβαθρο” (background) δηλ. ξεκινάει μία διεργασία ανεξάρτητη από τον φλοιό η οποία λειτουργεί ακόμα και αν η διεργασία του φλοιού τερματιστεί.

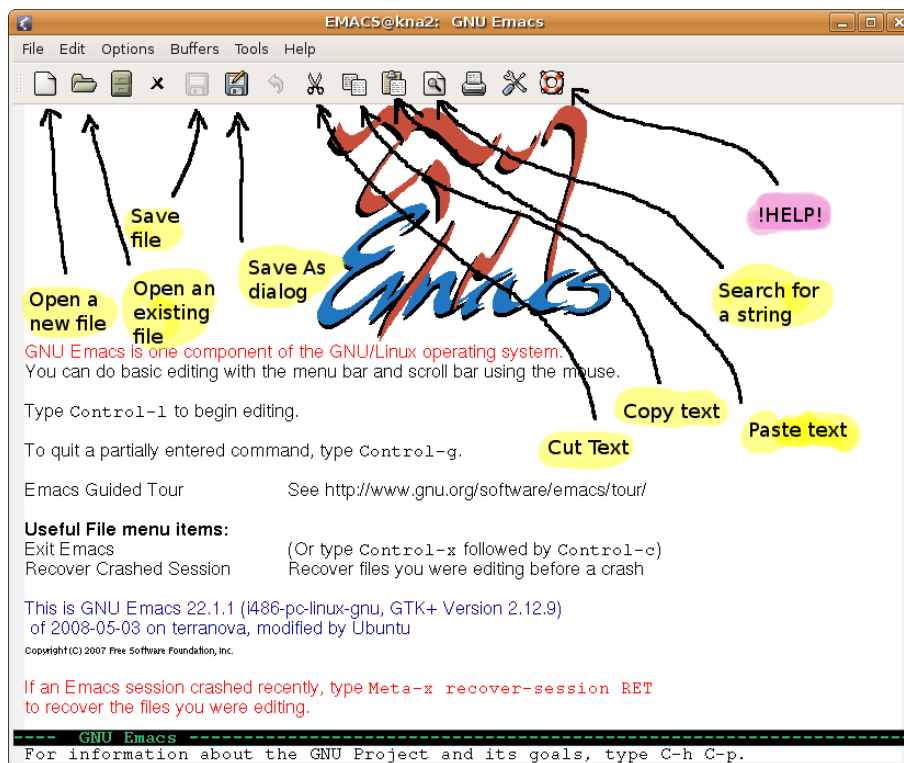
Τα παραπάνω ισχύουν όταν έχουμε παραθυρικό περιβάλλον και τότε ο Emacs ξεκινάει στο δικό του ανεξάρτητο παράθυρο. Μπορούμε όμως

<sup>21</sup>Ο Emacs είναι γραμμένος σε μια διάλεκτο της γλώσσας προγραμματισμού Lisp που λέγεται Elisp. Για προγραμματισμό απλών λειτουργιών δεν απαιτείται λεπτομερής γνώση της γλώσσας αυτής.

να τρέχουμε τον Emacs και σε ένα απλό τερματικό, είτε για γρήγορη επεξεργασία κειμένου είτε γιατί δε διαθέτουμε παραθυρικό περιβάλλον<sup>22</sup> αλλά μόνο κονσόλα. Στην τελευταία περίπτωση απλά παραλείπουμε το & στο τέλος της εντολής, ενώ αν έχουμε παραθυρικό περιβάλλον και θέλουμε ο Emacs να τρέξει στην κονσόλα, δίνουμε την εντολή

```
> emacs -nw
```

και ο Emacs θα ξεκινήσει μέσα στην κονσόλα.

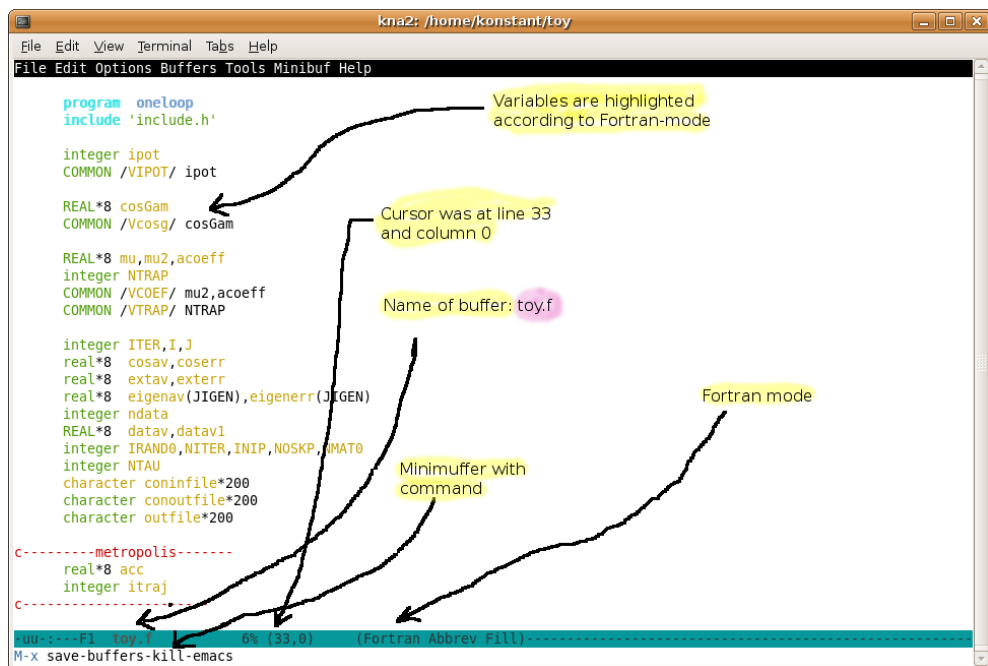


Σχήμα 1.2: Το παράθυρο του Emacs σε ένα παραθυρικό περιβάλλον. Φαίνονται και επεξηγούνται τα βασικά κουμπιά λειτουργίας του.

<sup>22</sup>Αυτό μπορούμε να το καταλάβουμε δίνοντας την εντολή `echo $DISPLAY` και αν πάρουμε το μήνυμα σφάλματος `DISPLAY: Undefined variable.`, τότε δεν έχουμε σύνδεση με παραθυρικό περιβάλλον (X server). Αλλιώς, θα πάρουμε την τιμή `:0.0`, `localhost:10.0` κλπ.

### 1.3.2 Αλληλεπιδρώντας με τον Emacs

Με τον Emacs αλληλεπιδρούμε με διάφορους τρόπους. Οι “νεοφώτιστοι αρχάριοι” θα προτιμήσουν τα κουμπιά και τα μενού που προσφέρει που συνήθως έχουν διαισθητική μορφή και ονόματα που συναντά στους περισσότερους επεξεργαστές κειμένου. Αλλά για να χρησιμοποιήσει κανείς τις προχωρημένες δυνατότητες του Emacs είναι καλό να συνηθίσει τις άλλες μορφές αλληλεπίδρασης που είναι οι συντομεύσεις πλήκτρων και η εκτέλεση εντολών με το όνομά τους από τη γραμμή εντολών του Emacs, το minibuffer<sup>23</sup>.



Σχήμα 1.3: Ο Emacs στην κονσόλα. Στην εικόνα έχουμε μεταβεί στο minibuffer πληκτρολογώντας M-x και έχουμε γράψει την εντολή save-buffers-kill-emacs η οποία τερματίζει τον Emacs αφού σώσει τα μεταβληθέντα από την επεξεργασία buffers. Η ίδια εντολή δίνεται ισοδύναμα πληκτρολογώντας C-x C-c. Φαίνεται η mode line στην οποία, ανάμεσα σε άλλα, είναι γραμμένο το όνομα του αρχείου/buffer (toy.f), το ποσοστό του buffer που είναι ορατό στο παράθυρο (6%), η γραμμή και η στήλη που βρίσκεται το σημείο που επεξεργαζόμαστε (33,0) και το editing mode που βρίσκεται ο buffer [Fortran mode (Fortran), Abbreviation mode (Abbrev), Auto Fill mode (Fill)].

Οι εντολές που δίνονται με συντομεύσεις από το πληκτρολόγιο είναι

<sup>23</sup>Είναι και ο πιο απλός τρόπος αλληλεπίδρασης, όταν καλούμε τον Emacs στην κονσόλα.



συνδυασμός πλήκτρων που πατά κάποιος σε συνδυασμό με τα πλήκτρα Ctrl (Control key) και Alt. Θα ακολουθήσουμε την εξής σύμβαση: Όταν γράφουμε έναν συνδυασμό πλήκτρων αρχίζοντας με C-, θα εννοούμε ότι τα πλήκτρα που ακολουθούν πατιούνται ταυτόχρονα με το Control key, ενώ αν γράφουμε M-, θα εννοούμε ότι τα πλήκτρα που ακολουθούν πατιούνται ταυτόχρονα με το Alt key<sup>24</sup>. Μερικές εντολές συντομεύονται από μια ακολουθία από δύο ή περισσότερους χαρακτήρες. Λ.χ. πατώντας C-x C-c (δηλ. κρατάμε πατημένο το Ctrl key και ταυτόχρονα πατάμε το x και μετά κρατώντας πατημένο το Ctrl key πατάμε το c) δίνουμε την εντολή να βγούμε από τον Emacs, ενώ πατώντας C-x 2 (δηλ. κρατάμε πατημένο το Ctrl key και ταυτόχρονα πατάμε το x και μετά αφήνουμε το Ctrl key και πατάμε το 2) δίνουμε την εντολή να χωριστεί το παράθυρο του buffer που βρισκόμαστε σε δύο ίσα μέρη.

Οι πιο χρήσιμες συντομεύσεις είναι οι M-x (πατάμε το Alt και κρατώντας το πατημένο πατάμε το x) και η C-g. Η πρώτη μας οδηγεί στο minibuffer από όπου μπορούμε να δώσουμε μία εντολή με το όνομά της. Για παράδειγμα, δώστε την εντολή save-buffers-kill-emacs που απλά θα τερματίσει τη συνεδρία του Emacs. Η δεύτερη είναι το “κουμπί SOS” που διακόπτει οτιδήποτε κάνει ο Emacs (λ.χ. αν κάποια εντολή κολλήσει, δώσουμε λάθος εντολή κλπ): Πατώντας C-g ο Emacs σταματάει οποιαδήποτε διεργασία κάνει και επιστρέφει στο buffer που εργαζόμαστε. Λ.χ. αν πατήστε κατά λάθος M-x και βρεθείτε στο minibuffer χωρίς να το θέλετε, πατήστε C-g για να ακυρώσετε τη διαδικασία και να επιστρέψετε στο buffer που επεξεργαζόσαστε.

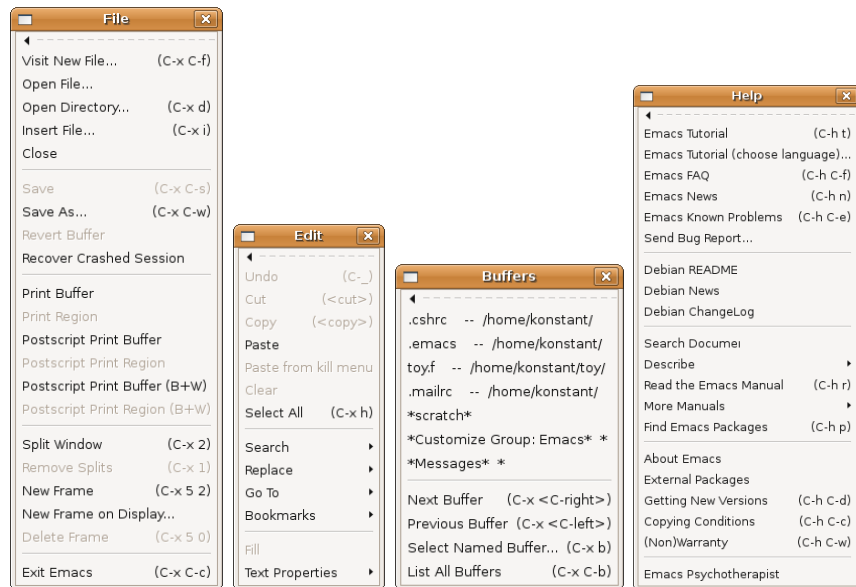
Είναι, επίσης, χρήσιμο να ορίσουμε συμβάσεις που να υποδηλώνουν τι κάνουμε με το ποντίκι. Με Mouse-1, Mouse-2, Mouse-3 υποδηλώνουμε ένα απλό κλικ με το αριστερό, μεσαίο<sup>25</sup> και δεξί κουμπί αντίστοιχα. Με Drag-Mouse-1 υποδηλώνουμε ότι κρατάμε το αριστερό κουμπί διαρκώς κρατημένο και ταυτόχρονα σέρνουμε το ποντίκι.

Ανακεφαλαιώνουμε συνοψίζοντας τους δυνατούς τρόπους για να δίνουμε μία εντολή στον Emacs. Θεωρούμε λ.χ. την εντολή που ανοίγει ένα καινούργιο αρχείο σε ένα buffer:

- Από το εικονίδιο που μοιάζει με λευκό χαρτί επάνω αριστερά στη γραμμή των κουμπιών του σχήματος 1.2.

<sup>24</sup>Στη γλώσσα του Emacs, το M- είναι το “Meta” key το οποίο βρίσκεται εκτός από το Alt και στο Esc (Escape key). Στην περίπτωση αυτή, σε αντίθεση με το Alt, το Esc το πατάμε πρώτα και το αφήνουμε και μετά πατάμε τα επόμενα πλήκτρα. Αυτό μπορεί να είναι και η πιο απλή μας επιλογή σε ορισμένα “χαζά” τερματικά. Αν και αυτό δε δουλεύει – μάλλον σπάνιο στην εποχή μας – δοκιμάστε το C-[.

<sup>25</sup>Αν το ποντίκι δεν έχει μεσαίο κουμπί πατάμε τη ροδέλα. Αν δεν έχει ροδέλα, τότε το αριστερό και το δεξί κουμπί ταυτόχρονα.



Σχήμα 1.4: Τα βασικά μενού που συναντά κανείς στον Emacs σε παραθυρικό περιβάλλον. Βλέπουμε τις βασικές εντολές και σε παρένθεση μας υπενθυμίζεται η αντίστοιχη συντόμευση πληκτρολογίου. Λ.χ. η εντολή File → Visit New File μπορεί να δοθεί από το πληκτρολόγιο πληκτρολογώντας C-x C-f. Σημειώστε τις εντολές File → Visit New File (άνοιγμα αρχείου), File→Save (εγγραφή αλλαγών του buffer στο αντίστοιχο αρχείο), File→Exit Emacs (κλείσιμο Emacs), File → Split Window (χωρισμός παραθύρου στα δύο), File→New Frame (άνοιγμα νέου παραθύρου) και φυσικά τις γνωστές εντολές Cut, Copy, Paste, Undo από το Edit menu. Από το μενού Buffers μπορούμε να επιλέξουμε διαφορετικά buffers με τα περιεχόμενα των άλλων αρχείων που επεξεργαζόμαστε. Στους καινούργιους χρήστες συστήνουμε να δουν το Emacs Tutorial και Read Emacs Manual στο Help menu.

- Από την εντολή μενού File→Visit New File.
- Από τη συντόμευση πληκτρολογίου C-x C-f.
- Από εντολή στο minibuffer: M-x find-file.

Ο πρώτος τρόπος είναι διαθέσιμος για τις πολύ βασικές εντολές, ο δεύτερος για περισσότερες, ο τρίτος για τις περισσότερες (αλλά όχι όλες) και ο τέταρτος για όλες τις εντολές που είναι διαθέσιμες για διαδραστική χρήση.

### 1.3.3 Βασική Επεξεργασία Κειμένου

Για να επεξεργαστούμε ένα αρχείο, ο Emacs τοποθετεί τα περιεχόμενά του σε ένα buffer. Το buffer είναι ένα κομμάτι της μνήμης όπου αντι-

γράφονται τα περιεχόμενα ενός αρχείου και όχι το ίδιο το αρχείο. Για να καταγραφούν οι αλλαγές στα περιεχόμενα ενός buffer πρέπει να τις “σώσουμε”, δηλ. ο Emacs να γράφει το buffer πίσω στο αρχείο. Μέχρι να γίνει αυτό το αρχικό αρχείο μένει ανέπαφο<sup>26</sup>. Ο Emacs μπορεί να έχει ανοιχτά πολλά buffers τα οποία, όταν συνδέονται με ένα αρχείο, έχουν από προεπιλογή το ίδιο όνομα του αρχείου<sup>27</sup>. Το όνομα ενός buffer φαίνεται στη mode line του Emacs όπως φαίνεται στο σχήμα 1.3. Ο κύκλος επεξεργασίας ενός αρχείου συνοψίζεται στα εξής σημεία:

- Διάβασμα των περιεχομένων του αρχείου σε ένα buffer.
- Αλλαγή από τον χρήστη των περιεχομένων του buffer.
- Εγγραφή των δεδομένων του buffer πίσω στο αρχείο.

Φυσικά αν το αρχείο δεν υπάρχει και δημιουργείται εξ’ αρχής, το πρώτο βήμα παραλείπεται.

Το σημείο στο οποίο βρισκόμαστε νοητά και εισάγουμε κείμενο λέγεται “το σημείο” (point). Αυτό καταδεικνύεται από τον δρομέα (cursor) που τυπικά είναι ένα κόκκινο τετραγωνάκι που αναβοσβήνει<sup>28</sup>. Κάθε buffer έχει μία θέση που ονομάζεται “το σημάδι” (the mark) το οποίο μαζί με το σημείο ορίζει σε κάθε παράθυρο την “περιοχή” (the region). Αυτή είναι μια νοητή περιοχή κειμένου σε κάθε παράθυρο όπου μπορούν να δράσουν οι συναρτήσεις του Emacs (λ.χ. αποκοπή, αντιγραφή, αλλαγή κεφαλαίων σε μικρά γράμματα, έλεγχος ορθογραφίας κλπ). Την περιοχή τη θέτουμε ορίζοντας το σημάδι (mark) επιλέγοντας ένα σημείο και πληκτρολογώντας C-SPC<sup>29</sup> (ή στο minibuffer M-x set-mark-command). Μετακινώντας τον δρομέα στο σημείο που θέλουμε, ορίζουμε την επιθυμητή περιοχή. Εναλλακτικά με το Drag-Mouse-1 (κρατάμε αριστερό

<sup>26</sup> Αν χάσουμε μία συνεδρία του Emacs είναι δυνατόν να ανακτήσουμε μέρος των αλλαγών που κάναμε. Μπορούμε να χρησιμοποιήσουμε την εντολή M-x recover-file ή να αναζητήσουμε ένα αρχείο στον δίσκο με όνομα ίδιο με αυτό του αρχείου που επεξεργαζόμαστε ανάμεσα σε δύο #. Λ.χ. το buffer του αρχείου file.f90 σώζεται αυτόματα και περιοδικά στο αρχείο #file.f90#

<sup>27</sup> Αυτό δεν είναι αναγκαστικό. Μπορεί ο χρήστης να αλλάξει το όνομα ενός buffer χωρίς να αλλάξει το αρχείο με το οποίο συνδέεται. Επίσης, αν ανοίξουμε αρχεία που έχουν το ίδιο όνομα (λ.χ. index.html) που βρίσκονται σε διαφορετικούς καταλόγους, τότε αυτά ονομάζονται από προεπιλογή index.html, index.html<2>, index.html<3>, ...

<sup>28</sup> Το σημείο είναι πάντα μεταξύ χαρακτήρων όχι πάνω σε αυτούς. Ο δρομέας είναι πάνω στον χαρακτήρα αμέσως δεξιά από το σημείο. Κάθε παράθυρο έχει ένα σημείο οπότε κάθε buffer μπορεί να έχει περισσότερα από ένα σημεία, αν εμφανίζεται σε διαφορετικά παράθυρα.

<sup>29</sup> Πατάμε ταυτόχρονα το Control key και το Space bar.

κουμπί ποντικιού πατημένο και σέρνουμε το ποντίκι) μαρκάρουμε μία περιοχή. Το σημάδι μπορεί να τεθεί και με Mouse-3 δηλ. με απλό κλικ του δεξιού πλήκτρου του ποντικιού (άρα, Mouse-1 Mouse-3 ορίζουν μία περιοχή θέτοντας πρώτα το σημείο και μετά το σημάδι).

Ανοίγουμε ένα αρχείο με την εντολή C-x C-f και πληκτρολογώντας το όνομά του. Αν το αρχείο υπάρχει, βλέπουμε τα περιεχόμενά του στο buffer που δημιουργείται, αλλιώς παίρνουμε ένα άδειο buffer. Τότε:

- Πλοηγούμαστε στο buffer με τα βελάκια του πληκτρολογίου Εναλλακτικά με τις εντολές C-n, C-p, C-f και C-b.
- Αν έχουμε πολλές “σελίδες ” προχωράμε μία-μία σελίδα με Page Up, Page Dn από το πληκτρολόγιο. Εναλλακτικά, με τις εντολές C-v, M-v
- Εισάγουμε κείμενο απλά πληκτρολογώντας το.
- Σβήνουμε τους χαρακτήρες που βρίσκονται πίσω από το σημείο με το Backspace και αυτούς που είναι μπροστά με το Delete. Με την εντολή C-d σβήνουμε τον μπροστινό χαρακτήρα.
- Σβήνουμε ολόκληρη τη γραμμή που είναι μπροστά από το σημείο με C-k
- Ανοίγουμε μια καινούργια γραμμή με Enter ή C-o.
- Πάμε στην αρχή της γραμμής με το πλήκτρο Home και στο τέλος της με το πλήκτρο End. Εναλλακτικά, με C-a και C-e αντίστοιχα.
- Πάμε στην αρχή του buffer με το πλήκτρο C-Home και στο τέλος με C-End. Εναλλακτικά, με τις εντολές στο minibuffer: M-x beginning-of-buffer και M-x end-of-buffer.
- Πάμε σε μια γραμμή που θέλουμε με M-x goto-line. Στην προτροπή της εντολής δίνουμε τον αριθμό της γραμμής που θέλουμε να πάμε.
- Αναζητούμε κείμενο μπροστά από το σημείο με την εντολή C-s. Πληκτρολογούμε το κείμενο μέχρι να το βρούμε. Για να βρούμε το ίδιο κείμενο ξανά (και ξανά) πληκτρολογούμε C-s όσες φορές χρειαστεί. Το ίδιο κάνουμε με το C-r για να αναζητήσουμε κείμενο πίσω από το σημείο.
- Αν θέλουμε να γράψουμε Ελληνικά, διαβάζουμε την Παράγραφο 1.3.10, σελ. 37

Μόλις τελειώσουμε σώζουμε τις αλλαγές που κάναμε με την εντολή C-s ή από το εικονίδιο “δισκέτα” ή από το μενού File→Save. Επίσης, η εντολή στο minibuffer είναι M-x save-buffer.

### 1.3.4 Κόβοντας και ράβοντας

Για πιο προχωρημένη επεξεργασία ακολουθούμε τις παρακάτω οδηγίες:

- Undo!. Πολλές από τις παρακάτω αλλαγές μπορεί να είναι καταστροφικές. Ο Emacs έχει επαναλαμβανόμενο undo με μεγάλη μνήμη. Πατώντας C-/ επανειλημμένα προσεγγίζουμε την προηγούμενη κατάσταση που βρισκόμαστε. Εναλλακτικά, με C-x u και από το μενού Edit→Undo. Θυμίζουμε ότι το C-g σταματάει οποιαδήποτε λειτουργία του Emacs και έτσι μπορεί να μας γλυτώσει από μέρος μίας καταστροφής, αν διακόψουμε το έγκλημα κατά τη διάρκεια που εκτελείται.
- Αποκοπή κειμένου με το ποντίκι: Κάνουμε κλικ Mouse-1 στην αρχή του κειμένου που θέλουμε να αποκόψουμε, μετά κλικ Mouse-3 στο τέλος του κειμένου. Αμέσως μετά, ένα δεύτερο κλικ Mouse-3 και ... πάει (στην πραγματικότητα το κείμενο αντιγράφεται στο Kill ring<sup>30</sup> και είναι διαθέσιμο για επικόλληση).
- Αποκοπή κειμένου από το πληκτρολόγιο: Επιλέγουμε το αρχικό σημείο και θέτουμε το σημάδι (mark) με C-SPC. Μετακινούμε τον δρομέα αμέσως μετά το σημείο που θέλουμε να θέσουμε ως τέλος της περιοχής. Πληκτρολογούμε C-w.
- Αντιγραφή κειμένου με το ποντίκι: Σέρνουμε το ποντίκι Drag-Mouse-1 από την αρχή ως το τέλος. Η περιοχή “φωτίζεται”. Εναλλακτικά όπως στην αποκοπή χωρίς το δεύτερο κλικ στο τέλος: Mouse-1 στην αρχή και Mouse-3 στο τέλος.
- Αντιγραφή κειμένου με το πληκτρολόγιο: C-SPC στην αρχή, μετακινούμαστε το τέλος της περιοχής και μετά M-w.
- Επικόλληση κειμένου με το ποντίκι: Στο σημείο που θέλουμε να κάνουμε την εισαγωγή, πατάμε το μεσαίο κουμπί<sup>31</sup>.
- Επικόλληση κειμένου με το πληκτρολόγιο: Στο σημείο που θέλουμε να κάνουμε εισαγωγή πληκτρολογούμε C-y

---

<sup>30</sup> Ας πούμε το clipboard του Emacs.

<sup>31</sup> Αν δεν έχει, τη ροδέλα ή αριστερό και δεξί ταυτόχρονα.

- Επικόλληση κειμένου που είχαμε αντιγράψει παλιότερα: Μία εύκολη επιλογή είναι από το μενού `Edit→Paste from kill menu` και επιλέγουμε το κείμενο που θέλουμε να επικολλήσουμε. Από το πληκτρολόγιο όπως πριν `C-y` και αμέσως μετά `M-y` επανειλημμένα μέχρι να εμφανιστεί το κείμενο που θέλουμε.
- Εισαγωγή ολόκληρου αρχείου: Αν θέλουμε να εισάγουμε τα περιεχόμενα ενός ολόκληρου αρχείου, πληκτρολογούμε `C-x i` στο σημείο που θέλουμε να γίνει η εισαγωγή. Εναλλακτικά, με την εντολή `M-x insert-file`.
- Εισαγωγή ολόκληρου buffer: Αν θέλουμε να εισάγουμε τα περιεχόμενα ενός ολόκληρου buffer, το κάνουμε με την εντολή `M-x insert-buffer`.
- Αντικατάσταση κειμένου: Με την εντολή `M-x query-replace` αντικαθιστούμε διαδραστικά μία ακολουθία χαρακτήρων με μία άλλη. Στην ερώτηση αν θέλουμε να γίνει η αντικατάσταση, απαντούμε `y` (ναι), `n` (όχι), `q` (στοπ). Με `,` (κόμμα) γίνεται μία αντικατάσταση και σταματάει. Αν είμαστε σίγουροι ότι θέλουμε να γίνουν όλες οι αντικαταστάσεις, εκτελούμε την εντολή `M-x replace-string`.
- Αντικατάσταση κεφαλαίων-μικρών γραμμάτων: Επιλέγουμε μία περιοχή που θέλουμε να γίνει η αλλαγή και εκτελούμε μία από τις εντολές `M-x upcase-region`, `M-x capitalize-region`, `M-x downcase-region`.

Το `Edit` μενού έχει πολλές από τις παραπάνω λειτουργίες για τους νεοσύλλεκτους.

Τίποτα, επίσης, δε μας εμποδίζει οι αποκοπές, αντιγραφές, επικολλήσεις να γίνονται από το ένα παράθυρο στο άλλο ακόμα και αν πρόκειται για buffer συνδεδεμένα με διαφορετικά αρχεία.

### 1.3.5 Παράθυρα

Πολλές φορές είναι βολικό να επεξεργαζόμαστε το ίδιο ή διαφορετικά αρχεία σε διαφορετικά παράθυρα. Το “παράθυρο” (window) στον Emacs αναφέρεται σε διαφορετικές περιοχές του ίδιου παραθύρου με την έννοια που δίνουμε σε ένα παραθυρικό περιβάλλον. Ο Emacs μπορεί να χωρίσει ένα παράθυρο σε ένα ή περισσότερα παράθυρα οριζόντια ή κάθετα. Μελετήστε το σχήμα 1.5 στη σελίδα 75 στο οποίο επεξηγούνται

οι βασικές έννοιες. Επίσης, μπορεί να ανοίξει ένα διαφορετικό παράθυρο με την έννοια του παραθυρικού περιβάλλοντος. Τέτοια παράθυρα λέγονται “πλαίσια” (frames)<sup>32</sup>. Θα κρατήσουμε αυτή την ορολογία όταν αναφερόμαστε στον Emacs.

- Τοποθέτηση δρομέα στο κέντρο παραθύρου και καθαρισμός παραθύρου από σκουπίδια: C-1 (προσοχή: 1 όχι 1)
- Χωρισμός παράθυρου στα δύο, οριζόντια: C-x 2
- Χωρισμός παράθυρου στα δύο, κάθετα: C-x 3
- Κατάργηση των άλλων παράθυρων: C-x 1
- Κατάργηση του τρέχοντος παράθυρου: C-x 0
- Μετακίνηση δρομέα σε άλλο παράθυρο: Με Mouse-1 ή C-x o
- Αλλαγή μεγέθους παράθυρων: Με το ποντίκι Drag-Mouse-1 στις διαχωριστικές γραμμές τους. Με το πληκτρολόγιο, C-^ αλλάζει την οριζόντια διάσταση και C-} την κάθετη.
- Καινούργιο πλαίσιο: C-x 5 2
- Κατάργηση πλαισίου: C-x 5 0
- Μετακίνηση δρομέα σε άλλο πλαίσιο: Με το ποντίκι Mouse-1 ή με C-x 5 o.

Διαφορετικά παράθυρα μπορείτε να έχετε και όταν ο Emacs τρέχει στην κονσόλα, κάτι που μπορεί να είναι η μεγαλύτερη ευλογία για προχωρημένη επεξεργασία κειμένου, όταν δε βρίσκεστε σε παραθυρικό περιβάλλον. Φυσικά, τότε δεν μπορείτε να έχετε διαφορετικά πλαίσια.

### 1.3.6 Αρχεία και Buffers

- Άνοιγμα αρχείου: C-x C-f ή M-x find-file.

---

<sup>32</sup>Να σημειώσουμε πως όταν θέλετε να επεξεργαστείτε ένα ή περισσότερα αρχεία είναι καλό να ανοίγετε καινούργια πλαίσια στην ίδια συνεδρία του Emacs και όχι να ξεκινάτε κάθε φορά τον Emacs από την αρχή. Μια καινούργια διαδικασία Emacs τραβάει πόρους από το σύστημά σας και δεν επικοινωνεί με μία άλλη.

- Σώσιμο αλλαγών του buffer σε αρχείο: C-x C-s ή M-x save-buffer. Αν θέλουμε ταυτόχρονα να βγούμε από τον Emacs C-x C-c ή M-x save-buffers-kill-emacs (στο μενού File→Save ή εικονίδιο με δισκέτα).
- Σώσιμο αλλαγών σε buffer σε άλλο αρχείο: C-x C-w ή M-x write-file (στο μενού File→Save As ή στο εικονίδιο με δισκέτα και μολύβι).
- Σώσιμο όλων των buffers στα αρχεία τους: C-x s ή M-x save-some-buffers.
- Σύνδεση ενός buffer με ένα (άλλο) αρχείο: M-x set-visited-file-name.
- Κατάργηση buffer: C-x k
- Αλλαγή buffer στο παράθυρο που βρισκόμαστε: C-x b
- Προβολή λίστας buffer: Από το μενού Buffers ή με την εντολή C-x C-b. Στη δεύτερη περίπτωση πατώντας Enter δίπλα σε ένα buffer το εμφανίζουμε στο παράθυρο. Υπάρχουν εντολές διαχείρισης των buffers που μπορείτε να βρείτε από τη βοήθεια του Emacs (αν βάλετε τον δρομέα στο παράθυρο αυτό πληκτρολογήστε C-h m)
- Ανάκτηση δεδομένων από επεξεργασμένο buffer: Αν χάσατε τη συνεδρία ενός Emacs, μην απελπίζεστε. Στην καινούργια συνεδρία πληκτρολογήστε M-x recover-file και ακολουθήστε τις οδηγίες. Η εντολή M-x recover-session επαναφέρει όλα τα αρχεία που επεξεργαζόσαστε μαζί.
- Αρχεία ασφαλείας: Όταν σώζετε ένα buffer σε ένα αρχείο, το παλιό αρχείο γίνεται αντίγραφο ασφαλείας. Αν το αρχείο έχει όνομα myfile, το αντίγραφο ασφαλείας έχει όνομα myfile~. Είναι δυνατόν να παραμετροποιήσετε τον Emacs να κρατάει πολλές “εκδόσεις” (versions) των αλλαγών που κάνετε. Σας παραπέμπουμε στην τεκμηρίωση.
- Πλοήγηση και δράσεις σε καταλόγους: C-x d ή M-x dired. Μπορείτε να δράσετε στα αρχεία του καταλόγου (άνοιγμα, διαγραφή, αλλαγή ονομασίας, αντιγραφή κλπ) με τις ανάλογες εντολές (μέσα από το παράθυρο του dired δώστε την εντολή C-h m και διαβάστε τη σχετική τεκμηρίωση).



### 1.3.7 Modes

Σε κάθε buffer που επισκεπτόμαστε ο Emacs μπορεί να βρίσκεται σε διαφορετικά modes (όχι ένα, αλλά πολλά). Σε διαφορετικά modes οι συντομεύσεις των εντολών από το πληκτρολόγιο μπορεί να είναι διαφορετικές, ο χρωματισμός των δομικών στοιχείων του buffer διαφορετικός κλπ. Υπάρχουν major modes που είναι μοναδικές για κάθε buffer, αλλά και minor modes που μπορεί να συνυπάρχουν αρμονικά μαζί με άλλες major και minor modes. Ο Emacs μπορεί να ξεκινά αυτόματα μία major και μία ή περισσότερες minor modes ανάλογα με το όνομα ή/και το περιεχόμενο του αρχείου που επισκεπτόμαστε. Μπορούμε όμως και εμείς ρητά να επιλέξουμε και να επιβάλλουμε τις modes που επιθυμούμε με τις κατάλληλες εντολές.

Οι modes οι οποίες είναι ενεργές σε ένα buffer σημειώνονται μέσα σε παρένθεση στη mode line (βλ. σχήμα 1.3 και 1.5).

- M-x f90-mode: Μας ενδιαφέρει ιδιαίτερα στο μάθημα αυτό. Αφορά αρχεία με εντολές στη γλώσσα Fortran και τα πιο χρήσιμα χαρακτηριστικά τις είναι η τοποθέτηση του δρομέα στο κατάλληλο σημείο πατώντας το πλήκτρο TAB, ο χρωματισμός των εντολών και των μεταβλητών, η αναγνώριση των δομικών στοιχείων του προγράμματος (subroutines, if, do loops, comments, statement labels κλπ). Μια άλλη ενδιαφέρουσα λειτουργία είναι να πάρει τις εντολές μιας ολόκληρης περιοχής και να τις κάνει σχόλια (comments) δηλ. ανενεργές.
- M-x c-mode: Για κώδικα γραμμένο στη γλώσσα C. Ανάλογες modes για γλώσσες προγραμματισμού είναι οι java-mode, perl-mode, awk-mode, python-mode, makefile-mode, octave-mode, mathematica-mode και άλλες.
- M-x latex-mode: Για την επεξεργασία αρχείων που έχουν κείμενο σε L<sup>A</sup>T<sub>E</sub>X.
- M-x text-mode: Για την επεξεργασία απλών (.txt) αρχείων κειμένου.
- M-x fundamental-mode: Η βασική mode, όταν δεν υπάρχει καλύτερη...

Minor modes που παρουσιάζουν ενδιαφέρον είναι οι:

- M-x auto-fill-mode: Όταν μία γραμμή γίνεται πολύ μακριά, την κόβει αυτόματα. Σχετική είναι η εντολή M-x fill-paragraph.

- `M-x overwrite-mode`: Αντί να εισάγονται οι χαρακτήρες ανάμεσα στους υπάρχοντες, γράφονται από πάνω τους. Δίνοντας την εντολή ξανά, αλλάζουμε στην προηγούμενη κατάσταση.
- `M-x read-only mode`: Όταν θέλουμε να επισκεφτούμε ένα πολύτιμο αρχείο που δε θέλουμε να αλλάξουμε κατά λάθος τα περιεχόμενά του, μπαίνοντας σε αυτή τη mode ο Emacs απαγορεύει τις αλλαγές. Αυτό μπορεί να γίνει ανοίγοντας ένα αρχείο με την εντολή `C-x C-r` (`M-x find-file-read-only`) ή/και να εναλλάσσουμε τη mode με την εντολή `C-x C-q` (`M-x toggle-read-only`). Βλέπε σχήμα 1.5, το buffer `jack.c` που σημειώνεται στο mode line με `%%`. Κάνοντας κλικ πάνω στα `%%`, μπορούμε να καταργούμε/επαναφέρουμε την `read-only mode`.
- `M-x flyspell-mode`: Άμεσος έλεγχος ορθογραφίας.
- `M-x font-lock-mode`: Χρωματισμός δομικών στοιχείων του buffer. Συνήθως είναι η προεπιλογή, αν όχι, την ενεργοποιούμε (ή την απενεργοποιούμε) με αυτήν την εντολή.

Σε παραθυρικό περιβάλλον έχουμε τη δυνατότητα να επιλέξουμε modes από την mode line. Με το Mouse-3 πάνω στο όνομα μιας mode μας δίνονται επιλογές για την (απ)ενεργοποίηση minor modes. Με το Mouse-1 μπορούμε να (απ)ενεργοποιήσουμε την `read-only mode` κάνοντας κλικ αριστερά στο `:%%` ή `--` αντίστοιχα. Βλέπε σχήμα 1.5.

### 1.3.8 Βοήθεια στον Emacs

Ο Emacs έχει πολύ λεπτομερή online τεκμηρίωση. Στους νέους χρήστες συστήνεται να ακολουθήσουν τις οδηγίες στο `emacs tutorial` το οποίο εκπαιδεύει τον χρήστη στις βασικές εντολές χρήσης και επεξεργασίας κειμένου. Αυτό γίνεται με την εντολή `C-h t` ή από το μενού `Help→Emacs Tutorial`. Αφεθείτε στις οδηγίες και είναι ... διασκεδαστικό. Η `man page` του (εντολή `man emacs`) έχει συνοπτικές πληροφορίες, κυρίως για τον τρόπο που καλείται ο Emacs από τη γραμμή εντολών.

Πολύ εκτενής τεκμηρίωση βρίσκεται στις `info pages`<sup>33</sup>. Η χρήση του `info` είναι κεφάλαιο βιβλίου από μόνη της, αλλά στο παραθυρικό περιβάλλον του Emacs είναι σχετικά απλή. Με την εντολή `C-h r` (μενού

<sup>33</sup> Αν προτιμάτε έγγραφα-βιβλία σε μορφή PDF επισκεφτείτε την ιστοσελίδα του Emacs [www.gnu.org/software/emacs/](http://www.gnu.org/software/emacs/) και επιλέξτε `Documentation`. Αυτός ο σύνδεσμος τώρα είναι ο [www.gnu.org/software/emacs/manual/emacs.html](http://www.gnu.org/software/emacs/manual/emacs.html) από όπου μπορείτε να κατεβάσετε το εγχειρίδιο χρήσης των 600 περίπου σελίδων!

Help→Read the Emacs Manual) ανοίγουμε απευθείας τη σελίδα του Emacs. Πατώντας τα πλήκτρα SPC και Backspace διαβάζουμε την τεκμηρίωση σελίδα-σελίδα. Αλλά στο info έχουμε υπερσυνδέσμους όπως στην πλοήγηση στο διαδίκτυο. Με Mouse-1 επιλέγετε έναν σύνδεσμο και με τα βελάκια στα εικονίδια μπορείτε να πάτε στον προηγούμενο/επόμενο σύνδεσμο όπως και στην προηγούμενη θέση που είσατε. Πατώντας πλήκτρα, δίνετε εντολές στο info, λ.χ. πατώντας d βρίσκεστε στον κεντρικό κατάλογο του info και μπορείτε να δείτε όλες τις εφαρμογές που έχουν info τεκμηρίωση. Με την εντολή g (info) (πληκτρολογήστε τους χαρακτήρες όπως τους βλέπετε με τις παρενθέσεις) βρίσκεστε στην τεκμηρίωση του info και εκεί μπορείτε να μάθετε την προχωρημένη χρήση της για να διαβάζετε αποτελεσματικά την τεκμηρίωση των εφαρμογών.

Ο Emacs είναι δομημένος διαισθητικά και φιλικά προς τον χρήστη. Τα περισσότερα θα τα μάθετε όχι από προσεκτική μελέτη του εγχειρίδιου χρήσης, αλλά από τα ίδια τα ονόματα των εντολών και τη συνοπτική τεκμηρίωσή τους. Όλες οι εντολές του Emacs αποτελούνται από ολόκληρες λέξεις που χωρίζονται με ένα - που σχεδόν σχηματίζουν πλήρεις προτάσεις.

- Auto completion εντολών: Οι εντολές αυτοσυμπληρώνονται στο minibuffer πατώντας το πλήκτρο TAB. Για παράδειγμα, μεταβείτε στο minibuffer πληκτρολογώντας M-x. Πληκτρολογήστε λ.χ. capi [TAB] και η εντολή συμπληρώνεται σε capitalize-. Πατώντας το [TAB] άλλη μια φορά ανοίγει ένα buffer με τις δυνατές επιλογές: capitalize-region και capitalize-word. Πληκτρολογήστε r [TAB] και η εντολή συμπληρώνεται στη μοναδική capitalize-region. Για να δείτε όλες της εντολές που αρχίζουν από s (...πολλές), πληκτρολογήστε M-x s [TAB] [TAB]. Επιλέξτε με το ποντίκι το buffer \*Completions\* και πλοηγηθείτε στις δυνατότητες. Από τα ονόματα των εντολών θα πάρετε ιδέες. Αν έχετε χρόνο, πληκτρολογήστε M-x [TAB] [TAB] και όλες οι διαθέσιμες εντολές θα είναι στο ... buffer σας!
- Τεκμηρίωση συντομεύσεων πληκτρολογίου: Δεν ξέρετε τι κάνει η εντολή C-s; Κανένα πρόβλημα... Πληκτρολογήστε C-h k και μετά C-s. Αντίστροφα, ξεχάσατε με ποια πλήκτρα συντομεύεται η εντολή save-buffer; Πληκτρολογήστε C-h w και μετά την εντολή.
- Τεκμηρίωση συναρτήσεων: Ψάχνετε μια εντολή λ.χ. save-κάτι-που-ξέχασα; Πληκτρολογήστε C-h f και μετά save- [TAB] να δείτε τις επιλογές. Με Mouse-2 επιλέξτε την εντολή που σας ενδιαφέρει

ή πληκτρολογήστε/συμπληρώστε με [TAB] το υπόλοιπο όνομα (λ.χ. save-buffers-kill-emacs) και στο buffer \*Help\* θα διαβάσετε την τεκμηρίωση της εντολής.

- Τεκμηρίωση μεταβλητών: Με C-h ν μπορείτε να μάθετε την τιμή και τη λειτουργία των μεταβλητών στον Emacs.
- Τεκμηρίωση apropos: Δεν θυμάστε ακριβώς το όνομα της εντολής; Κανένα πρόβλημα... Πληκτρολογήστε C-h a και μια λέξη (λ.χ. save) και θα δείτε όλες τις εντολές που περιέχουν τη λέξη αυτή. Περισσότερες πληροφορίες θα πάρετε με C-h d
- Τεκμηρίωση modes: Πληκτρολογώντας C-h m μέσα από ένα buffer παίρνετε πληροφορίες για τις modes που είναι ενεργοποιημένες για το buffer. Θα δείτε εκεί τις ειδικές εντολές που δέρονται με τις συντομεύσεις πληκτρολογίου.
- Τεκμηρίωση info: C-h i
- Ξεχάσατε τα παραπάνω ή θέλετε να μάθετε και άλλα; Πληκτρολογήστε C-h ? και πλοηγηθείτε στις επιλογές που σας δίνονται.

### 1.3.9 Παραμετροποίηση του Emacs

Ο Emacs έχει δυνατότητα παραμετροποίησης σε οποιοδήποτε βάθος: Από την απλή σύνδεση πλήκτρων με εντολές που θέλουμε να συντομεύσουμε μέχρι τον προγραμματισμό πολύπλοκων λειτουργιών στη γλώσσα Elisp. Ο πιο διαδεδομένος τρόπος για τον μέσο χρήστη είναι να εισάγει τις κατάλληλες εντολές στο αρχείο ~/.emacs στην προσωπική του περιοχή. Ο Emacs διαβάζει και εκτελεί τις εντολές αυτές πριν ξεκινήσει. Παράδειγμα ενός τέτοιου αρχείου με ενδεικτικές λειτουργίες είναι το παρακάτω:

```
; Define F1 key to save the buffer
(global-set-key [f1] 'save-buffer)
; Define Control-c s to save the buffer
(global-set-key "\C-cs" 'save-buffer)
; Define Meta-s (Alt-s) to interactively search forward
(global-set-key "\M-s" 'isearch-forward)
; Define M-x is to interactively search forward
(defalias 'is 'isearch-forward)
; Define M-x fm to set fortran-mode for the buffer
(defun fm() (interactive) (f90-mode))
; Define M-x sign to sign my name
```

```
(defun sign() (interactive) (insert "K. N. Anagnostopoulos"))
```

Στα περιεχόμενα του παραπάνω αρχείου τα ελληνικά ερωτηματικά ; ορίζουν το υπόλοιπο της γραμμής να είναι σχόλια, τα οποία είναι για μας και αγνοούνται (δεν ερμηνεύονται). Οι πρώτες τρεις εντολές δεσμεύουν τα πλήκτρα F1, C-c s και M-s σε συγκεκριμένες συναρτήσεις-εντολές. Η επόμενη δείχνει πώς να ορίσουμε ψευδώνυμο (alias) μιας εντολής που χρησιμοποιούμε συχνά. Οι τελευταίες δύο ορίζουν δύο πολύ απλές συναρτήσεις (fm) και (sign) που μπορούμε να τις καλέσουμε από το minibuffer όπως αναφέρεται στα σχετικά σχόλια.

Για περισσότερα παραδείγματα αναζητήστε στο Google: “emacs .emacs file” για να δείτε τα αρχεία που χρησιμοποιούν άλλοι χρήστες.

Επίσης, είναι δυνατόν να παραμετροποιήσετε τον Emacs από το μενού Options → Customize Emacs.

Για τη σε βάθος εκμάθηση της γλώσσας Elisp σας παραπέμπουμε στο Emacs Lisp Reference Manual στη διεύθυνση [www.gnu.org/software/emacs/manual/elisp.html](http://www.gnu.org/software/emacs/manual/elisp.html)

### 1.3.10 Ελληνικά στον Emacs

Με πολλή συντομία περιγράφουμε πώς γίνεται να επεξεργαστούμε αρχεία με ελληνικούς χαρακτήρες. Εδώ ο χρήστης πρέπει να προσδιορίσει αν οι ελληνικοί χαρακτήρες θα αναπαρίστανται από τους 8-bit χαρακτήρες του συστήματος iso8859-7<sup>34</sup> ή από τους πιο διαδεδομένους 16-bit Unicode χαρακτήρες.

Για να μπορέσουμε να διαβάσουμε αρχεία με χαρακτήρες Unicode πρέπει ο Emacs σε ένα παραθυρικό περιβάλλον να ξεκινήσει με μία κατάλληλη γραμματοσειρά Unicode (UTF)<sup>35</sup>. Αν αυτό δεν είναι η προεπιλογή, επιλέγουμε εμείς μία γραμματοσειρά<sup>36</sup>. Μια επιλογή δίνεται από την παρακάτω εντολή (ο χαρακτήρας \ συνεχίζει την εντολή στην επόμενη γραμμή, εσείς μπορείτε να τη γράψετε σε μία γραμμή):

```
> emacs -fn \
-misc-fixed-medium-r-normal--18-120-100-100-c-90-iso10646-1 &
```

<sup>34</sup>Χρήσιμοι σε προγράμματα όπως το L<sup>A</sup>T<sub>E</sub>X με τη χρήση του Babel.

<sup>35</sup>Αν καλούμε τον Emacs στην κονσόλα, θα πρέπει η κονσόλα να εμφανίζει χαρακτήρες Unicode

<sup>36</sup>Με την εντολή `xlsfonts | grep iso10646 | less` βλέπουμε τις διαθέσιμες γραμματοσειρές Unicode, ενώ με την εντολή `xlsfonts | grep iso8859-7 | less` τις διαθέσιμες ελληνικές 8-bit γραμματοσειρές ISO8859-7.

Στη συνέχεια, μπορούμε να εισάγουμε ελληνικούς χαρακτήρες χρησιμοποιώντας την αλλαγή της μεθόδου πληκτρολογίου του παραθυρικού περιβάλλοντος (λ.χ. πληκτρολογώντας Alt-Shift) όπως και σε οποιαδήποτε άλλη εφαρμογή. Εναλλακτικά (αν λ.χ. δεν είμαστε σε UTF περιβάλλον) με την εντολή C-\ (M-x toggle-input-method) και εισάγοντας –μόνο την πρώτη φορά – “greek” στο minibuffer εναλλάσσουμε από αγγλικά σε ελληνικά.

Για τους 8-bit χαρακτήρες τύπου ISO8859-7 καλούμε τον Emacs με την ανάλογη γραμματοσειρά. Μία επιλογή είναι

```
> emacs -fn \
  -misc-fixed-medium-r-normal--18-120-100-100-c-90-iso8859-7 &
```

Αφού ανοίξουμε το αρχείο που επιθυμούμε σε ένα buffer, μπορούμε από το μενού να διαλέξουμε περιβάλλον γλώσσας Options -> Mule (Multilingual environment) -> Set Language Environment -> Greek (ή στο minibuffer M-x set-language-environment) και επιλέγουμε μέθοδο εισαγωγής χαρακτήρων Options -> Mule (Multilingual environment) -> Select Input Method-> ``greek'' (ή στο minibuffer M-x toggle-input-method). Στη συνέχεια, με τη συντόμευση εντολής C-\ εναλλάσσουμε από αγγλικά σε ελληνικά.

## 1.4 Η Γλώσσα Προγραμματισμού: Fortran

Στην παράγραφο αυτή θα αναφέρουμε τα απολύτως απαραίτητα που χρειάζεται να ξέρετε προκειμένου να αρχίσετε να γράφετε και να τρέχετε προγράμματα σε γλώσσα Fortran. Δεν πρόκειται για συστηματική εκμάθηση της γλώσσας, αλλά για μία πρακτική προσέγγιση μέσω παραδειγμάτων. Ο αναγνώστης ενθαρρύνεται να ανατρέξει στη βιβλιογραφία [9,10,11] για περισσότερες λεπτομέρειες.

Για να ωφεληθεί ο/η αναγνώστης/τρια από το κεφάλαιο αυτό πρέπει απαραίτητα να γράφει τα προγράμματα και να τα εκτελεί στον υπολογιστή του/της.

### 1.4.1 Τα Στοιχειώδη

Το πρώτο πρόγραμμα που γράφει κανείς σε μια καινούργια γλώσσα ή/και υπολογιστικό περιβάλλον, είναι ένα “Hello World” πρόγραμμα, το οποίο απλά τυπώνει στο stdout αυτή τη φράση. Καταφέροντας να δει τη φράση αυτή τυπωμένη, έχει κάνει τη μισή δουλειά που χρειάζεται

για να προγραμματίσει στο περιβάλλον αυτό. Το εν λόγω πρόγραμμα σε Fortran το γράφουμε σε ένα αρχείο `hello.f90` ως εξής:

```
program hello

!print a message to the world:
print *, 'Hello World!' !this is a comment

end program hello
```

Οι εντολές στη Fortran είναι ακολουθίες χαρακτήρων που γράφουμε από την 1η μέχρι και την 132η στήλη. Κάθε γραμμή αρχίζει μία καινούργια εντολή<sup>37</sup> και μπορούμε σε μία γραμμή να γράψουμε περισσότερες από μία εντολές χωρίζοντάς τις με ένα ελληνικό ερωτηματικό (; - semicolon). Αν μια σειρά αρχίζει με ! (θαυμαστικό), αυτή η σειρά αγνοείται και μπορεί να χρησιμοποιηθεί για σχόλια που επεξηγούν και τεκμηριώνουν το πρόγραμμα. Αν μια σειρά έχει θαυμαστικό σε κάποια στήλη, οτιδήποτε μετά το θαυμαστικό αγνοείται και μπορεί να χρησιμοποιηθεί για σχολιασμό (εξαιρείται το θαυμαστικό μέσα σε μονά η διπλά εισαγωγικά, όπου εκεί θεωρείται ότι είναι μέρος του κειμένου μιας ακολουθίας χαρακτήρων όπως λ.χ. στο 'Hello World!'). Η σωστή και επαρκής τεκμηρίωση ενός προγράμματος είναι απαραίτητη για προγράμματα που σχεδιάζετε, ώστε να μπορούν να χρησιμοποιηθούν επανειλημμένα, να συντηρηθούν και να επεκταθούν από ομάδα πολλών προγραμματιστών και να είναι μεγαλύτερα από μερικές γραμμές κώδικα. Η ιδέα που είχατε σήμερα θα σας φαντάζει άγνωστη μετά από λίγες εβδομάδες και ο χρόνος που θα καταναλώσετε για να διορθώσετε, να βελτιώσετε και να εξηγήσετε στους άλλους αυτό που κάνατε, θα είναι σίγουρα περισσότερος από το χρόνο που θα ξοδέψετε για να συγγράψετε μια καλή και σαφή τεκμηρίωση του προγράμματός σας.

Η κύρια είσοδος σε ένα πρόγραμμα καθορίζεται από την εντολή `program name` όπου `name` είναι ό,τι θέλουμε, αρκεί να αρχίζει από γράμμα A-Z ή a-z, να αποτελείται από το πολύ 31 αλφαριθμητικούς χαρακτήρες και το `_` (underscore). Το τέλος του προγράμματος, όπως και κάθε αυτοδύναμης ενότητας του προγράμματος (υπορουτίνες, συναρτήσεις), καθορίζεται από την εντολή `end` στην οποία προσθέτουμε και το όνομα της ενότητας που κλείνει (εδώ `end program hello`).

Στην τέταρτη γραμμή είναι το “ζουμί”: Η εντολή `print` είναι ο απλούστερος τρόπος να τυπώσουμε κάτι στο `stdout`. Προσέξτε το “\*,” που

<sup>37</sup>Μία μεγάλη εντολή μπορεί να συνεχιστεί σε επόμενη γραμμή, όπως θα πούμε παρακάτω, βάζοντας στο τέλος της γραμμής ένα `&` και συνεχίζοντας στην επόμενη.

είναι μέρος του συντακτικού και φυσικά δεν τυπώνεται... Για τη Fortran τα κεφαλαία/μικρά γράμματα είναι ισοδύναμα και θα μπορούσαμε να γράψουμε PRINT, Print, .... Η φράση που θέλουμε να τυπώσουμε είναι μια ακολουθία χαρακτήρων που περικλείεται από μονά ή διπλά εισαγωγικά ('Hello World!' ή "Hello World!").

Για να τρέξει το πρόγραμμα, πρέπει να μεταφραστεί σε γλώσσα μηχανής. Τη δουλειά αυτή την αναλαμβάνει ο μεταγλωττιστής (compiler). Σε κάθε σύστημα το πρόγραμμα αυτό μπορεί να έχει διαφορετικό όνομα ή ακόμα και ο προγραμματιστής να έχει περισσότερες από μία επιλογές. Πρέπει να ενημερωθείτε από τον διαχειριστή του συστήματος ή τα σχετικά εγχειρίδια. Τυπικά ονόματα τέτοιων προγραμμάτων είναι f90, ifort, gfortran, .... Η πρώτη μας δουλειά είναι να μελετήσουμε με προσοχή τα εγχειρίδια χρήσης. Εκεί μαθαίνουμε πώς να χρησιμοποιήσουμε τις δυνατότητές του με τον καλύτερο τρόπο για το δικό μας πρόγραμμα (λ.χ. βελτιστοποίηση - optimization)

Στο δικό μας σύστημα θα χρησιμοποιήσουμε τον gfortran<sup>38</sup>. Η εντολή που θα δώσουμε για τη μεταγλώττιση είναι<sup>39</sup>

```
> gfortran hello.f90 -o hello
```

Ο διακόπτης -o ορίζει ο μεταγλωττισμένος κώδικας να γραφτεί στο αρχείο hello. Αν η μεταγλώττιση είναι επιτυχής, τότε το πρόγραμμα τρέχει με την εντολή:

```
> ./hello  
Hello world!
```

όπου το ./ το βάλαμε για να προσδιορίσουμε ρητά πώς εκτελούμε το πρόγραμμα που περιέχεται στο αρχείο hello και πού βρίσκεται στον τρέχοντα κατάλογο (.).

Ας δοκιμάσουμε τώρα να κάνουμε έναν απλό υπολογισμό, την περιμέτρο και το εμβαδόν ενός κύκλου ακτίνας R. Για τον λόγο αυτό θα χρειαστούμε να χρησιμοποιήσουμε μεταβλητές τύπου REAL. Στο αρχείο area\_01.f90 πληκτρολογούμε

```
program circle_area
```

<sup>38</sup>Ο μεταγλωττιστής αυτός είναι ελεύθερο λογισμικό και είναι διαθέσιμος για όλα τα δημοφιλή λειτουργικά συστήματα. Δείτε στο σύνδεσμο <http://gcc.gnu.org/fortran/>

<sup>39</sup>Θυμίζουμε στον αναγνώστη ότι γραμμές που αρχίζουν με > είναι εντολές που δίνουμε στη γραμμή εντολών. Οτιδήποτε άλλο είναι output του προγράμματος.



```

PI = 3.14159265358979
R  = 4.0
print *, 'Perimeter= ', 2.0*PI*R
print *, 'Area=      ', PI*R**2

end program circle_area

```

Στο παραπάνω πρόγραμμα ορίσαμε τις τιμές των δύο μεταβλητών R, PI στη 3η και 4η γραμμή. Το ότι οι μεταβλητές είναι τύπου REAL καθορίζεται από το όνομα της μεταβλητής. Η Fortran έχει implicit rules για να το καθορίζει. Σύμφωνα με αυτούς, μεταβλητές που το όνομά τους αρχίζει από i, j, k, l, m, n είναι τύπου INTEGER (ακέραιοι), ενώ κάθε άλλη είναι τύπου REAL. Αλλαγή γίνεται μόνο αν δηλώσουμε ρητά τον τύπο μιας μεταβλητής όπως θα δείξουμε αργότερα<sup>40</sup>. Στην 5η και 6η γραμμή κάνουμε τον υπολογισμό  $2\pi R$  και  $\pi R^2$  κατευθείαν στο όρισμα της εντολής print. Οι τελεστές πολλαπλασιασμού και δύναμης είναι \* και \*\* αντίστοιχα. Προσέξτε ότι στις σταθερές 2.0 και 4.0 βάλαμε ρητά την υποδιαστολή. Αν τις παραλείψουμε, οι σταθερές είναι τύπου INTEGER και, αν δεν είναι αυτό που πραγματικά θέλουμε, το αποτέλεσμα μπορεί να μας ... καταπλήξει<sup>41</sup>. Αν υποθέσουμε ότι το πρόγραμμα είναι αποθηκευμένο στο αρχείο area\_01.f90, οι εντολές μεταγλωττισμού και εκτέλεσης του προγράμματος είναι

```

> gfortran area_01.f90 -o area
> ./area
Perimeter=      25.13274
Area=          50.26548

```

Ας δοκιμάσουμε τώρα μια επαναλαμβανόμενη διεργασία. Ας κάνουμε τον παραπάνω υπολογισμό για 10 διαφορετικούς κύκλους ακτίνας  $R_i = 1.28 + i, i = 1, \dots, 10$ . Τις ακτίνες θα τις αποθηκεύσουμε σε ένα array R(10) τύπου REAL. Το αρχείο area\_02.f90:

```

program circle_area

dimension R(10)

PI = 3.14159265358979
R(1) = 2.28
do i=2,10
  R(i) = R(i-1) + 1.0

```

<sup>40</sup>Μπορούμε να αλλάξουμε τους κανόνες αυτούς με την εντολή implicit.

<sup>41</sup>Δοκιμάστε την εντολή print \*, 2.0/4.0, 2/4 στο παραπάνω πρόγραμμα.

```

enddo

do i = 1,10
  perimeter = 2*PI*R(i)
  area      = PI*R(i)**2
  print *,i,' ) R= ',R(i),' perimeter= ',perimeter
  print *,i,' ) R= ',R(i),' area      = ',area
enddo

end program circle_area

```

Η εντολή dimension R(10) ορίζει ένα μονοδιάστατο array με 10 στοιχεία. Με τον τρόπο αυτό στη Fortran, τα στοιχεία των arrays αναφέρονται με έναν δείκτη που παίρνει τιμές από 1 μέχρι το μήκος του array (εδώ 10). Άρα R(4) είναι το τέταρτο στοιχείο του R.

Μεταξύ των εντολών

```

do i = 2, 10
  ...
enddo

```

περιέχονται εντολές που εκτελούνται επαναληπτικά με την ακέραια μεταβλητή i να παίρνει τιμή από 2 έως 10 με βήμα 1<sup>42</sup>. Η εντολή

```
R(i) = R(i-1) + 1.0
```

ορίζει την ακτίνα με δείκτη i να είναι κατά 1 μεγαλύτερη από την προηγούμενη. Για να είναι σωστή η επαγωγή θα πρέπει να ορίσουμε την τιμή της R(1), πριν αρχίσει το do loop. Μετά από αυτή την εξήγηση, νομίζω πως μπορεί εύκολα να γίνει κατανοητό τι γίνεται στο δεύτερο do loop του προγράμματος. Ο αναγνώστης θα πρέπει να δοκιμάσει το παραπάνω πρόγραμμα και να πειραματιστεί κάνοντας μικροαλλαγές.

Ας μετατρέψουμε τώρα το παραπάνω πρόγραμμα, έτσι ώστε ο χρήστης να δίνει διαδραστικά τις ακτίνες του κύκλου, το πρόγραμμα να υπολογίζει τις ακτίνες και τα εμβαδά και στη συνέχεια, να γράφει τα αποτελέσματα σε ένα αρχείο. Άρα, το πρόγραμμα πρέπει να πάρει ως input από τον χρήστη τα μέτρα των ακτίνων  $R_i, i = 1, \dots, 10$ . Γράφουμε στο αρχείο area\_03.f90:

```
program circle_area
```

<sup>42</sup>Το βήμα μπορεί να αλλάξει λ.χ. do i=0,12,4 τρέχει για i=0,4,8,12 και η do i=10,6,-2 για i=10,8,6 αντίστοιχα.

```

implicit none

integer ,parameter      :: N=10
real    ,parameter      :: PI=3.141593
real    ,dimension(N)   :: R
real                                          :: area ,perimeter
integer                                     :: i

do i=1,N
  print*, 'Enter radius of circle: '
  read *, R(i)
  print*, 'i= ',i, ' R(i)= ',R(i)
enddo

open(UNIT=13,FILE='AREA.DAT')

do i = 1,N
  perimeter = 2*PI*R(i)
  area      = PI*R(i)**2
  write(13,*)i, ' R= ',R(i), ' area= ',area,&
    ' perimeter= ',perimeter
enddo

close(13)

end program circle_area

```

Παρατηρήστε τώρα ότι η πρώτη εντολή που δίνουμε είναι η `implicit none`. Αυτό δηλώνει ότι δε θέλουμε να χρησιμοποιήσουμε τους `implicit` κανόνες της Fortran, αλλά θέλουμε να υποχρεώσουμε τον εαυτό μας να δηλώσει ρητά κάθε μεταβλητή του προγράμματος. Αυτό σημαίνει πως θα μας πάρει λίγο παραπάνω χρόνο να πληκτρολογήσουμε τους ορισμούς, αλλά σας υπόσχομαι ότι αυτός ο κόπος δεν συγκρίνεται με τίποτα με τον πόνο να βρει κάποιος δύσκολα σφάλματα στο πρόγραμμα που οφείλονται σε μικρά ορθογραφικά λάθη στα ονόματα των μεταβλητών<sup>43</sup>. Θα ακολουθήσουμε αυτή την πρακτική σε ολόκληρο το βιβλίο.

Μετά από αυτή την εντολή ακολουθούν οι δηλώσεις (declarations) των μεταβλητών. Οι μεταβλητές `N`, `i` δηλώνονται ως `integer`, ενώ οι `PI`, `area`, `perimeter`, `R(N)` ως `real`. Οι `N`, `PI` δηλώνονται να είναι παράμετροι (parameter) των οποίων η τιμή δεν μπορεί να αλλάξει στη ροή του προγράμματος.

Μετά τις δηλώσεις των μεταβλητών ακολουθούν οι εκτελέσιμες εντολές. Το πρώτο `do loop` δεν έχει τίποτα καινούργιο εκτός από την εντολή

<sup>43</sup>Ποια η διαφορά στο όνομα της μεταβλητής `p11` από την `p11`;

```
read *, R(i)
```

Με την εντολή αυτή διαβάζουμε από το stdin την τιμή της μεταβλητής R(i). Ο χρήστης πρέπει να την πληκτρολογήσει στο τερματικό και να πατήσει το πλήκτρο [Enter]. Μπορούμε με την ίδια εντολή read να διαβάσουμε περισσότερες από μία μεταβλητές.

Για να τυπώσουμε δεδομένα σε ένα αρχείο, πρέπει να συνδέσουμε το όνομα του αρχείου με ένα UNIT που αντιστοιχεί σε έναν ακέραιο με τιμή μέσα σε κάποια όρια που καθορίζονται από το σύστημα<sup>44</sup>. Η σύνδεση αυτή γίνεται με την εντολή open και μετά μπορούμε να γράφουμε με την εντολή write(unit\_number,\*)...<sup>45</sup>. Όταν τελειώσουμε, κλείνουμε το αρχείο με την εντολή close και μπορούμε να συνδέσουμε UNIT με ίδιο αριθμό σε άλλο αρχείο. Η λογική ροή είναι δηλαδή

```
open(UNIT=13,FILE='AREA.DAT')
...
write(13,*) ....
...
close(13)
```

Το όνομα του αρχείου καθορίζεται από το όρισμα FILE='AREA.DAT' της εντολής open και εδώ η διαφοροποίηση στα κεφαλαία ή τα μικρά γράμματα δίνει διαφορετικά αρχεία. Το όρισμα FILE='path' μπορεί να πάρει οποιοδήποτε path από το σύστημα των αρχείων.

Το τελευταίο που παρατηρούμε είναι η γραμμή

```
write(13,*)i,') R= ',R(i), ' area= ',area,&
      ' perimeter= ',perimeter
```

που μας δείχνει πώς να συνεχίζουμε μια μακριά εντολή στην επόμενη γραμμή. Αρκεί να βάλουμε το χαρακτήρα & στο τέλος της γραμμής και η επόμενη γραμμή θεωρείται συνέχεια της προηγούμενης. Αυτό μπορεί να γίνει μέχρι 39 φορές.

Το επόμενο βήμα είναι να μάθουμε πώς να χωρίζουμε το πρόγραμμά μας σε λογικά διαφορετικές διαδικασίες οι οποίες μπορεί να επαναλαμβάνονται πολλές φορές στο πρόγραμμά μας. Θα δείξουμε τη διαδικασία της υπορουτίνας (subroutine) ορίζοντας τον υπολογισμό του εμβαδού και της περιφέρειας του κύκλου να γίνεται από την subroutine

<sup>44</sup>Μπορείτε με ασφάλεια να χρησιμοποιήσετε από 10-99. Ειδικά, το 5 είναι το stdin, το 6 το stdout και το 0 το stderr.

<sup>45</sup>Δοκιμάστε τι γίνεται, αν γράψετε σε ένα UNIT χωρίς να έχετε ανοίξει ένα αρχείο...

area\_of\_circle. Ορίστε τι γράφουμε μέσα στο αρχείο area\_04.f90:

```

program circle_area

  implicit none

  integer ,parameter  :: N=10
  real    ,parameter  :: P=3.141593
  real    ,dimension(N):: R
  real    :: area,perimeter
  integer :: i

  do i=1,N
    print*, 'Enter radius of circle: '
    read *, R(i)
    print*, 'i= ',i, ' R(i)= ',R(i)
  enddo

  open(UNIT=13,FILE='AREA.DAT')

  do i = 1,N
    call area_of_circle(R(i),perimeter,area)
    write(13,*)i,') R= ',R(i), ' area= ',area,&
      ' perimeter= ',perimeter
  enddo

  close(13)

end program circle_area

subroutine area_of_circle(R,L,A)
  implicit none
  real :: R,L,A
  real ,parameter :: PI = 3.141593 , PI2 = 2.0*PI

  L= PI2*R
  A= PI*R*R

  return
end subroutine area_of_circle

```

Οι αλλαγές που κάναμε αφορούν καταρχήν το κυρίως πρόγραμμα. Οι υπολογισμοί της περιμέτρου και του εμβαδού αντικαταστάθηκαν από τη γραμμή

```
call area_of_circle(R(i),perimeter,area)
```

Η εντολή `call` κάνει αυτό που λέει: καλεί τη διαδικασία που ορίζεται στην υπορουτίνα `area_of_circle`. Τα  $(R(i), \text{perimeter}, \text{area})$  είναι τα ορίσματα της υπορουτίνας. Το  $R(i)$  είναι μεταβλητή εισόδου η οποία παρέχει δεδομένα για να κάνει τον υπολογισμό η υπορουτίνα. Οι `perimeter, area` είναι οι μεταβλητές εξόδου στις οποίες κατά την έξοδό της η υπορουτίνα αποθηκεύει τα αποτελέσματα. Ο προγραμματιστής της υπορουτίνας πρέπει να μας δώσει σαφείς οδηγίες για τις μεταβλητές εισόδου/εξόδου έτσι ώστε να χρησιμοποιήσουμε σωστά την υπορουτίνα.

Η υπορουτίνα προγραμματίζεται ανάμεσα στις δηλώσεις

```
subroutine area_of_circle(R,L,A)
...
end subroutine area_of_circle
```

Τα ορίσματα  $R, L, A$  ορίζονται στην υπορουτίνα και τα ονόματά τους δεν είναι αναγκαστικό να είναι τα ίδια με αυτά που χρησιμοποιούμε για να καλέσουμε την υπορουτίνα. Δηλώνονται ρητά με τις δηλώσεις `real :: R, L, A`. Οι μεταβλητές περνούν *by reference* το οποίο σε απλά ελληνικά σημαίνει πως οποιαδήποτε αλλαγή στις τιμές τους μέσα στην υπορουτίνα, αλλάζει και τις αντίστοιχες τιμές στο πρόγραμμα που την κάλεσε. Άρα με τις εντολές `L= PI2*R`, `A= PI*R*R` πετυχαίνουμε αυτό που θέλουμε, δηλ. να επιστρέψουμε στο χρήστη της υπορουτίνας την περίμετρο και το εμβαδόν κύκλου ακτίνας  $R$ . Τέλος με την εντολή `return` επιστρέφουμε τον έλεγχο στο πρόγραμμα που κάλεσε την υπορουτίνα. Οι μεταβλητές - παράμετροι `PI`, `PI2` είναι “ιδιωτικές” της `area_of_circle` και δεν “φαίνονται” από το κυρίως πρόγραμμα. Το ίδιο και οι μεταβλητές του κυρίως προγράμματος (`i`, `N`, ...) δεν είναι γνωστές στην υπορουτίνα.

Τέλος ας δώσουμε χωρίς πολλά λόγια και ένα πρόγραμμα `trionymo.f90` που υπολογίζει τις ρίζες ενός τριωνύμου:

```
! =====
! Program to compute roots of a 2nd order polynomial
! Tasks: Input from user, logical statements,
!        use of functions, stop
!        Accuracy in floating point arithmetic
!        e.g. IF(x.eq.0.0)
!
! Tests: a,b,c= 1 2 3 D= -8
!        a,b,c= 1 -8 16 D= 0 x1= 4
```

```

!      a,b,c= 1 -1 -2 D=      9. x1=      2. x2=     -1.
!      a,b,c= 2.3 -2.99 -16.422 x1=      3.4 x2=     -2.1
! But:  6.8(x-4.3)**2 = 6.8 x**2 -58.48*x+125.732
!      a,b,c= 6.8 -58.48 125.73199
!      D= 0.000204147349 x1=      4.30105066 x2=      4.29894924
!      a,b,c= 6.8 -58.48 125.732, D=      -0.000210891725 < 0!!
! =====
program trionymo
  implicit none
  real :: a,b,c,D
  real :: x1,x2
  real :: Discriminant

  print*, 'Enter a,b,c:'
  read *,a,b,c

  ! Test if we have a well defined polynomial of 2nd degree:
  if( a .eq. 0.0) stop 'trionymo: a=0'

  ! Compute the discriminant (= diakrinousa)
  D = Discriminant(a,b,c)
  print *, 'Discriminant: D= ',D

  ! Compute the roots in each case: D>0, D=0, D<0 (no roots)
  if(D .gt. 0.0 )then
    call roots(a,b,c,x1,x2)
    print *, 'Roots:      x1= ',x1, ' x2= ',x2
  else if (D .eq. 0.0) then
    call roots(a,b,c,x1,x2)
    print *, 'Double Root: x1= ',x1
  else
    print *, 'No real roots'
  endif

end program trionymo

! =====
! This is the function that computes the discriminant
! A function returns a value. This value is assigned with the
! statement:
! Discriminant = <value>
! i.e. we simply assign anywhere in the program a variable with
! the name of the function.
! =====
real function Discriminant(a,b,c)
  implicit none
  real :: a,b,c

  Discriminant = b**2 - 4.0 * a * c

```

```

end function Discriminant
! =====
! The subroutine that computes the roots.
! =====
subroutine roots(a,b,c,x1,x2)
  implicit none
  real :: a,b,c
  real :: x1,x2
  real :: D, Discriminant

  if(a .eq. 0.0) stop 'roots: a=0'

  D = Discriminant(a,b,c)
  if(D.ge.0.0)then
    D = sqrt(D)
  else
    print *, 'roots: Sorry , cannot compute roots , D<0=',D
    stop
  endif

  x1 = (-b + D)/(2.0*a)
  x2 = (-b - D)/(2.0*a)

end subroutine roots

```

Το πρόγραμμα ζητάει τους συντελεστές του τριωνύμου  $ax^2+bx+c$ . Ελέγχει αν είναι καλά ορισμένο  $a \neq 0$  και αν όχι, σταματάει το πρόγραμμα με την εντολή stop. Στη συνέχεια, υπολογίζει τη διακρίνουσα (discriminant)  $D = b^2 - 4ac$  καλώντας τη συνάρτηση Discriminant(a,b,c). Η συνάρτηση (function) διαφέρει από τη subroutine στο ότι καλείται απευθείας (χωρίς την εντολή call) και επιστρέφει μια τιμή της οποίας ο τύπος πρέπει να δηλωθεί όπως οποιαδήποτε άλλη μεταβλητή (real :: Discriminant). Στη συνέχεια, ξεχωρίζουμε τις γνωστές περιπτώσεις με τη δομή

```

if(D .gt. 0.0 )then
  ...
else if (D .eq. 0.0) then
  ...
else
  ...
endif

```



όπου παρατηρούμε και τους τελεστές σύγκρισης `.gt.` (greater than-αυστηρά μεγαλύτερο) και `.eq.` (equal-ίσο)<sup>46</sup>.

Για τη συνάρτηση `Discriminant` πρέπει να δηλωθεί τι τύπου τιμή επιστρέφει (εδώ `real`), καθώς και ο τύπος των ορισμάτων της όπως και για τη subroutine. Η τιμή που επιστρέφει καθορίζεται τοποθετώντας τη σε μια μεταβλητή με όνομα ίδιο με αυτό της συνάρτησης:

```
real function Discriminant(a,b,c)
...
Discriminant = b**2 - 4.0 * a * c
...
end function Discriminant
```

### 1.4.2 Μερικές λεπτομέρειες

Την παράγραφο αυτή μπορείτε να την αγνοήσετε την πρώτη φορά που διαβάζετε αυτό το κεφάλαιο. Σκοπός είναι περισσότερο να χρησιμεύσει σαν αναφορά, όταν θα έχετε απορίες στα επόμενα κεφάλαια.

Ξεκινάμε αναφέροντας και άλλους ενδιαφέροντες τύπους μεταβλητών. Στο παρακάτω πρόγραμμα δείχνουμε πώς να χρησιμοποιήσετε μεταβλητές τύπου `CHARACTER`, πραγματικούς διπλής ακρίβειας `REAL(8)` και μιγαδικούς αριθμούς μονής `COMPLEX` και διπλής ακρίβειας `COMPLEX(8)`:

```
program f90_vars
  implicit none

  character(100) :: string

  real(4)      :: x !single precision, same as real :: x
  real(8)      :: x8 !equivalent to: double precision x8
  !real(16)    :: x16 !may not be supported by all compilers
  !Complex Numbers:
  complex(4)   :: z  !single precision, same as complex :: z
  complex(8)   :: z8 !double precision

  !A string: a character array:
  string = 'Hello World!' !string smaller size, leaves blanks
                          !TRIM: trim blanks
  print *, 'A string :: ', string, ' :: ', TRIM(string), ' :: '
  print *, 'join them :: ', string // string, ' :: '
```

<sup>46</sup>Παρόμοιοι τελεστές είναι `.lt.`, `.ge.`, `.le.` (μικρότερο, μεγαλύτερο ή ίσο, μικρότερο ή ίσο) και `.ne.`, `.and.`, `.or.` (μη ίσο, λογικό και, λογικό ή)

```

print *, 'join them:: ', TRIM(string) // TRIM(string), ':: '
!Reals with increasing accuracy: Determine PI=3.14159...
x  = 4.0 *atan(1.0 )
!Use D for double precision exponent
x8  = 4.0D0*atan(1.0D0)
!Use Q for quadruple precision exponent
!x16 = 4.0Q0*atan(1.0Q0)
print *, 'x4= ', x, ' x8= ', x8, '!', ' x16= ', x16
print *, 'x4: ', range(x ), precision(x ), EPSILON(x ), &
        TINY(x ), HUGE(x )
print *, 'x8: ', range(x8 ), precision(x8 ), EPSILON(x8 ), &
        TINY(x8 ), HUGE(x8 )

!Complex numbers: single precision
z = (2.0,1.0)*cexp((3.0,-1.0))
print *, 'z= ', z, ' Re(z)= ', REAL(z), ' Im(z)= ', IMAG(z), &
        ' |z|= ', ABS(z), ' z*= ', CONJG(z)

!Complex numbers: double precision
z8 = (2.0D0,1.0D0)*cdexp((3.0D0,-1.0D0))
print *, 'z= ', z8, ' Re(z)= ', DBLE(z8), ' Im(z)= ', DIMAG(z8), &
        ' |z|= ', CDABS(z8), ' z*= ', DCONJG(z8)
print *, 'z4: ', range(z ), precision(z )
print *, 'z8: ', range(z8 ), precision(z8 )

end program f90_vars

```

Τα σημεία που πρέπει να προσέξουμε στο παραπάνω πρόγραμμα είναι:

- Ο αριθμός  $K$  στη δήλωση `REAL(K)` ::  $x$  υποδηλώνει τον αριθμό των bytes που χρησιμοποιούνται για την αποθήκευση της μεταβλητής  $x$ . Για  $K=4$  έχουμε μεταβλητές μονής ακρίβειας, για  $K=8$  διπλής και για  $K=16$  τετραπλής ακρίβειας. Η τελευταία δυνατότητα δεν προσφέρεται σε όλες τις πλατφόρμες. Για τις δηλώσεις `COMPLEX(K)` ::  $z$ , το  $K$  αναφέρεται στην ακρίβεια του πραγματικού και φανταστικού μέρους του  $z$ <sup>47</sup>.
- Όταν χρησιμοποιούμε σταθερές στις μεταβλητές διπλής ακρίβειας, βάζουμε πάντα τον εκθέτη, έστω και αν είναι 0. Ο εκθέτης υποδηλώνεται με το γράμμα `D` αντί του `E` που χρησιμοποιείται για τις μεταβλητές `REAL`. Αλλιώς, η σταθερά χάνει την επιθυμητή ακρίβεια.

<sup>47</sup>Μονή ακρίβεια σημαίνει χοντρικά 7 σημαντικά ψηφία και τάξεις μεγέθους από  $10^{-38}$  –  $10^{38}$ . Διπλή ακρίβεια σημαίνει χοντρικά 16-17 σημαντικά ψηφία και τάξεις μεγέθους από  $10^{-308}$  –  $10^{308}$ . Παρατηρήστε πώς παίρνουμε αυτές τις πληροφορίες από τις συναρτήσεις `range`, `precision`, `tiny`, `huge`.

- Οι συναρτήσεις για μεταβλητές διπλής ακρίβειας, συνήθως, παίρνουν ένα έξτρα D στο όνομά τους ( $\text{exp} \rightarrow \text{dexp}$ ,  $\text{ABS} \rightarrow \text{DABS}$ ), ενώ οι αντίστοιχες για μιγαδικούς ένα έξτρα C ( $\text{DABS} \rightarrow \text{CABS}$ ,  $\text{exp} \rightarrow \text{cexp}$  κλπ). Τρέξτε το πρόγραμμα και παρατηρήστε την αυξημένη ακρίβεια υπολογισμού του  $\pi$  και του  $z = (2 + i)e^{3-i}$  χρησιμοποιώντας μεταβλητές διπλής ακρίβειας.
- Οι μεταβλητές τύπου CHARACTER δηλώνονται με το μέγεθός τους, εδώ 100 χαρακτήρες βάζοντας CHARACTER(100). Αν περάσουμε το όριο αυτό, οι παραπάνω χαρακτήρες ... κόβονται.
- Όταν θέτουμε μεταβλητές τύπου CHARACTER με το =, αυτές γεμίζουν από τα αριστερά προς τα δεξιά. Οι υπόλοιποι χαρακτήρες μέχρι το τέλος της μεταβλητής θέτονται να είναι ο κενός χαρακτήρας (blank).
- Όταν τυπώνεται μία μεταβλητή τύπου CHARACTER, τυπώνονται όλοι οι χαρακτήρες της, συμπεριλαμβανομένων και των καταληκτικών κενών (blanks). Με τη συνάρτηση TRIM, τα καταληκτικά κενά κόβονται. Παρατηρήστε πώς τυπώνεται το 'Hello World!' στο παραπάνω πρόγραμμα...
- Ο τελεστής // ενώνει δύο μεταβλητές ή/και σταθερές τύπου CHARACTER. Παρατηρήστε πώς επιδρά η συνάρτηση TRIM στα αποτελέσματα του παραπάνω προγράμματος.

Ένα άλλο σημαντικό στοιχείο της γλώσσας που παραλείψαμε στην προηγούμενη παράγραφο είναι η κοινή χρήση μεταβλητών από διαφορετικά μέρη του προγράμματος. Μία μεταβλητή που ορίζεται σε ένα υποπρόγραμμα (main program, subroutine, function) είναι τοπική και διαφορετικά υποπρογράμματα δεν μπορούν να έχουν πρόσβαση σε αυτή. Για να αποκτήσουμε πρόσβαση σε κοινό σημείο της μνήμης όπου αποθηκεύουμε τις τιμές των μεταβλητών, χρησιμοποιούμε την εντολή COMMON. Δείτε το παρακάτω παράδειγμα:

```
! -----
program f90_common
  implicit none
  real :: k1=1.0,k2=1.0,k3=1.0
  common /CONSTANTS/k1,k2

  print *, 'main: k1= ',k1, ' k2= ',k2, ' k3= ',k3
  call s1 !prints k1 and k2 but not k3
```

```

call s2 !changes the value of k2 but not k3
print *, 'main: k1= ', k1, ' k2= ', k2, ' k3= ', k3

end program f90_common
! -----
subroutine s1()
  implicit none
  real k1, k2, k3
  common /CONSTANTS/ k1, k2

  print *, 's1: k1= ', k1, ' k2= ', k2, ' k3= ', k3
end subroutine s1
! -----
subroutine s2()
  implicit none
  real k1, k2, k3
  common /CONSTANTS/ k1, k2

  k2 = 2.0
  k3 = 2.0
end subroutine s2

```

Το COMMON block εδώ έχει το όνομα CONSTANTS και μπορούμε να αναφερόμαστε σε αυτό από οποιαδήποτε υπορουτίνα ή συνάρτηση του προγράμματος. Στην πραγματικότητα, δείχνει σε ένα συγκεκριμένο σημείο της μνήμης και εδώ δεσμεύουμε τον χώρο για δύο μεταβλητές τύπου REAL τις k1, k2. Οι μεταβλητές αυτές διαβάζονται και αλλάζουν τιμές από τις υπορουτίνες s1 και s2, ενώ η k3 παρόλο που έχει κοινό όνομα και στο κύριο πρόγραμμα και στις υπορουτίνες, αναφέρεται σε διαφορετικές μεταβλητές κάθε φορά. Το πρόγραμμα τυπώνει:

main: k1=	1.000000	k2=	1.000000	k3=	1.000000
s1: k1=	1.000000	k2=	1.000000	k3=	-2.8117745E-05
main: k1=	1.000000	k2=	2.000000	k3=	1.000000

Ένα από τα αδύναμα σημεία της Fortran είναι η περιορισμένη δυνατότητα να χειριστούμε ευέλικτα το Input/Output (I/O). Για τον λόγο αυτό θα χρησιμοποιήσουμε άλλα προγράμματα όπως awk, perl ή προγράμματα στη γλώσσα C. Ακόμα όμως και η Fortran έχει χειρισμό του I/O, αλλά όντας επιστημονικά προσανατολισμένη αυτός αφορά κυρίως την ακρίβεια παρουσίασης των αριθμών. Αν έχετε να χειρισθείτε κείμενο με πολύπλοκο τρόπο, καλύτερα να διαλέξετε μια άλλη γλώσσα προγραμματισμού... Μέχρι στιγμής οι μόνες εντολές φορμά που χρησιμοποιήσαμε για I/O είναι οι προκαθορισμένες χρησιμοποιώντας το \*, λ.χ. print \*, read \*, write( ,\*). Αλλά το \* μπορεί να αντικατασταθεί

με εντολές φορμά σύμφωνα με το παρακάτω παράδειγμα:

```
program f90_format1
  implicit none
  integer          :: i
  real             :: x
  real, dimension(10) :: a
  real(8)          :: x8

  i = 123456
  x = 2.0 * atan2(1.0,0.0)
  print '(A5,I6,F12.7)', 'x,i= ', i,x
  x8 = 2.0D0*atan2(1.0D0,0.0D0)
  write(6, '(F18.16,E24.17,G24.17,G24.17)') x8,x8,&
    1.0D15*x8,1.0D18*x8
  write(6, '(3F20.16)') x8,x8/2.0,cos(x8)
  write(6, '(200F12.6)')(a(i), i=1,10)
end program f90_format1
```

Προσέξτε τις παρενθέσεις μέσα στα εισαγωγικά: (A5,I6,F12.7) είναι εντολή φορμά για την εντολή print και δίνει οδηγίες για την εκτύπωση τριών μεταβλητών: A είναι για CHARACTER, I για INTEGER και F για REAL. Οι αριθμοί αμέσως μετά το γράμμα υποδηλώνουν τον αριθμό των χαρακτήρων που θα χρησιμοποιηθούν για την εκτύπωση. Προσοχή! Αν δεν είναι αρκετές οι θέσεις εκτύπωσης, η Fortran θα αρνηθεί να κάνει την εκτύπωση και θα τυπώσει μια σειρά από \*, τα αστεράκια του τρόμου<sup>48</sup>. Και στις θέσεις αυτές πρέπει να συνυπολογίσετε τον αριθμό των δεκαδικών ψηφίων, την υποδιαστολή, το πρόσημο, τα ψηφία και το πρόσημο του εκθέτη... Μην είστε τσιγκούνηδες λοιπόν, δώστε άπλετο χώρο και μπορεί να σας χρειαστεί... Εδώ A5 υποδηλώνει CHARACTER που θα τυπωθεί σε 5 θέσεις χαρακτήρων, I6 INTEGER 6 θέσεων και F12 REAL 12 χαρακτήρων. Μετά την υποδιαστολή στο F12.7 υποδηλώνουμε πόσα δεκαδικά ψηφία θέλουμε να τυπωθούν.

Στην εντολή φορμά (F18.16,E24.17,G24.17,G24.17) δίνουμε οδηγίες για την εκτύπωση μιας μεταβλητής διπλής ακρίβειας. Στην καλύτερη περίπτωση έχουμε περίπου 16 δεκαδικά ψηφία ακρίβειας οπότε δεν έχει νόημα να κρατάμε παραπάνω (σε έναν υπολογισμό, συνήθως χάνουμε ακρίβεια). Με την εντολή F θέλει προσοχή: Αν χρειαστεί εκθέτης για την αναπαράσταση, ο αριθμός δε θα τυπωθεί και συνήθως το αποφεύγουμε, εκτός αν είμαστε σίγουροι ότι δε χρειάζεται εκθέτης. Η επιλογή E γράφει τον αριθμό πάντα σε επιστημονική μορφή με εκ-

<sup>48</sup>Σκεφτείτε μετά από έναν επίπονο υπολογισμό να πάτε να δείτε τα πολυπόθητα αποτελέσματα μόνο για να ανακαλύψετε ότι κάνατε λάθος στην εντολή του φορμά...

θέτη. Η επιλογή G γράφει τον αριθμό χωρίς εκθέτη, αν δε χρειάζεται, και με εκθέτη αν χρειάζεται. Οι αριθμοί έχουν την ίδια έννοια όπως και πριν. Στην εντολή φορμά (3F20.16) δείχνουμε πώς δίνουμε ένα πολλαπλασιαστικό παράγοντα 3 στην εκτύπωση των REAL\*8. Και στην τελευταία, δείχνουμε πώς να τυπώνουμε ένα μεγάλο διάνυσμα σε μία γραμμή: write(6, '(200F12.6)')(a(i), i=1,10). Ο πολλαπλασιαστικός παράγοντας μπορεί να είναι μεγαλύτερος από αυτόν που θα χρησιμοποιήσουμε. Το πρόγραμμα τυπώνει (τη δεύτερη γραμμή τη διπλώσαμε για να φαίνεται):

```
x,i= 123456      3.1415927
3.1415926535897931 0.31415926535897931E+01 3141592653589793.0
                                0.31415926535897933E+19
3.1415926535897931 1.5707963267948966 -1.0000000000000000
0.000000      0.000000      0.000000      ....
```

Οι εντολές φορμά μπορούν να μοιράζονται με ... παραπομπές. Αν μια εντολή αρχίζει με έναν αριθμό 1-99999, ο αριθμός αυτός είναι μία “ετικέτα” στην εντολή που ακολουθεί (labeled statement). Αν η εντολή αυτή είναι εντολή FORMAT, τότε μπορούμε να αναφερθούμε σε αυτή με τον αριθμό της από τις εντολές PRINT, WRITE και READ. Έτσι, πολλές εντολές I/O μπορούν να χρησιμοποιούν την ίδια εντολή FORMAT, αν τυπώνουν με τον ίδιο τρόπο. Το παρακάτω πρόγραμμα κάνει ακριβώς ό,τι και το παραπάνω με τη μόνη διαφορά ότι χρησιμοποιούμε labeled statements και εντολές FORMAT:

```
program f90_format2
  implicit none
  integer i
  real x, a(10)
  real*8 x8

  i = 123456
  x = 2.0 *atan2(1.0,0.0)
  print 100,'x,i= ',i,x
  x8 = 2.0D0*atan2(1.0D0,0.0D0)
  write(6,123) x8,x8,&
    1.0D15*x8,1.0D18*x8
  write(6,4444) x8,x8/2.0,cos(x8)
  write(6,9999)(a(i), i=1,10)
100  FORMAT(A5,I6,F12.7)
123  FORMAT(F18.16,E24.17,G24.17,G24.17)
4444 FORMAT(3F20.16)
9999 FORMAT(200F12.6)
```

```
end program f90_format2
```

Τέλος, ο/η αναγνώστης/τρια θα πρέπει να μελετήσει τις διαθέσιμες συναρτήσεις της Fortran (intrinsic functions) που δίνονται στον Πίνακα 1.2 της σελίδας 77.

### 1.4.3 Χειρισμός των arrays

Την παράγραφο αυτή μπορείτε να την αγνοήσετε την πρώτη φορά που διαβάζετε αυτό το κεφάλαιο. Θα σας φανεί χρήσιμη αργότερα.

Τα arrays είναι ένας τρόπος να αναφερόμαστε σε μια συστοιχία τιμών συγκεκριμένου τύπου που είναι αποθηκευμένες στη μνήμη. Οι τιμές ή στοιχεία των arrays αναφέρονται με τους δείκτες οι οποίοι παίρνουν ακέραιες τιμές μέσα σε κάποια όρια. Για παράδειγμα

```
A(1), A(2), ..., A(10)
```

αναφέρεται στις δέκα επιτρεπόμενες real τιμές ενός array που έχει δηλωθεί ως `real, dimension(10) :: A`. Οι δείκτες μπορεί να είναι ακέραιες εκφράσεις, όπως

```
A(i), B(2*i+3), C(INT(x+y(j)))
```

όπου στην τελευταία χρησιμοποιούμε την ακέραια τιμή της συνάρτησης `INT(x)`. Παρατηρήστε ότι τα στοιχεία των arrays και οι τιμές συναρτήσεων χρησιμοποιούν ίδιου τύπου παρενθέσεις και για να καταλάβει ο μεταγλωττιστής τη διαφορά θα πρέπει να κοιτάξει τις δηλώσεις των ονομάτων τους. Οι δηλώσεις των arrays γίνονται με τον παρακάτω τρόπο:

```
real, dimension(10) :: a,b
real, dimension(20) :: c,d
```

ορίζει τα arrays `a`, `b`, `c`, `d` με στοιχεία `a(1) ... a(10)`, `b(1) ... b(10)`, `c(1) ... c(20)` και `d(1) ... d(20)` που είναι τύπου `real`. Ισοδύναμα θα μπορούσαν να οριστούν με τη δήλωση

```
real :: a(10), b(10), c(20), d(20)
```

ή

```
integer, parameter :: n1 = 10, n2 = 20
```



```
real,    dimension(n1) :: a, c(n2)
real                                :: b(n1), d(n2)
```

Στην τελευταία μορφή, χρησιμοποιήσαμε τις σταθερές  $n1$ ,  $n2$  για τις δηλώσεις και δείχνουμε πώς να κάνουμε δηλώσεις arrays με διαφορετική έκταση. Στην παραπάνω μορφή, το κατώτερο όριο (lower bound) των arrays είναι το 1 και το ανώτερο όριο (upper bound) 10 για τα  $a$ ,  $b$  και 20 για τα  $c$ ,  $d$ . Τα όρια αυτά μπορούμε να τα προσδιορίσουμε ρητά. Έτσι, οι δηλώσεις

```
integer, parameter      :: n1 = 10, n2 = 20
real,    dimension(0:n1) :: a
real,    dimension(-n1:n2) :: c
```

ορίζουν το array  $a$  με 11 τιμές  $a(0) \dots a(10)$  και το array  $c$  με 31 τιμές  $c(-10) \dots c(-1)$   $c(0)$   $c(1) \dots c(20)$ .

Τα παραπάνω arrays έχουν μόνο μία διάσταση (“διανύσματα”). Μπορούμε όμως να ορίσουμε και arrays με περισσότερες από μία διαστάσεις προσθέτοντας περισσότερους δείκτες στο όνομα του array<sup>49</sup>. Έτσι, η δήλωση

```
integer, dimension(2,2) :: a
```

δηλώνει ένα array με integer τιμές  $a(1,1)$ ,  $a(1,2)$ ,  $a(2,1)$ ,  $a(2,2)$ . Παρακάτω δηλώνουμε δύο arrays με τρεις διαστάσεις:

```
integer, parameter :: n1 = 10, n2 = 20, n3 = 2*n1+n2
real,    dimension(n1,n2,n3) :: a
real,    dimension(-n1:n1,0:n2,13:n3) :: b
```

Μερικοί σημαντικοί ορισμοί που θα συναντήσετε στη βιβλιογραφία (οι αγγλικοί όροι δίνονται με έντονη γραφή):

- **array**: διαδοχικές τιμές στη μνήμη ορισμένου τύπου στις οποίες αναφερόμαστε χρησιμοποιώντας έναν ή περισσότερους δείκτες. Οι μεταβλητές με μόνο μία τιμή λέγονται βαθμωτές (**scalar**).
- Κάθε διάσταση (**dimension**) έχει ένα άνω και ένα κάτω όριο (**upper bound**, **lower bound**) που καθορίζουν τα όρια των επιτρεπτών τιμών των δεικτών (**index**) του array. Όταν το lower bound παραλείπεται σε μία δήλωση, τότε αυτό τίθεται ίσο με 1.

<sup>49</sup>Η Fortran επιτρέπει μέχρι επτά δείκτες σε ένα array.



- Ο αριθμός των διαστάσεων ή ισοδύναμα ο αριθμός των δεικτών ενός array λέγεται **rank** του array.
- Η έκταση (**extent**) μιας διάστασης είναι ο αριθμός των στοιχείων στη διάσταση αυτή. Ισούται με (upper bound)-(lower bound)+1.
- Το μέγεθος (**size**) ενός array είναι ο συνολικός αριθμός των στοιχείων του. Για ένα μονοδιάστατο array είναι το ίδιο με το extent του array, ενώ για ένα πολυδιάστατο το γινόμενο των extents κάθε διάστασης.
- Το σχήμα (**shape**) ενός array είναι το rank και το extent της κάθε διάστασής του.

Για να καθορίσουμε τις τιμές των στοιχείων των arrays μπορούμε να τις χειριστούμε όπως τα scalars:

```
integer :: i
real    :: a(4), b(2,2)

b(1,1) = 2.0 ; b(1,2) = 4.0
b(2,1) = 3.4 ; b(2,2) = 7.8
do i=1,4
  a(i) = 1.0
enddo
```

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε το όνομα του array ως αντικείμενο. Για παράδειγμα

```
a = (/ 1.0, 2.0, 3.0, 4.0 /)
b = 0.0
```

στην πρώτη γραμμή θέτουμε τις τιμές στο array a χρησιμοποιώντας έναν array constructor. Στη δεύτερη γραμμή, όλα στοιχεία του b θέτονται ίσα με 0. Αυτό δείχνει μία εξαιρετική ευκολία που παρέχει η Fortran στη χρήση των arrays. Μπορούμε να χρησιμοποιήσουμε όλες τις intrinsic operations της Fortran με ολόκληρα arrays, αρκεί αυτά να είναι **conformable**. Δύο arrays είναι conformable, αν έχουν το ίδιο shape ή το ένα είναι scalar. Έτσι, το παρακάτω πρόγραμμα

```
integer :: i,j
real    :: x,y,a(10),b(10),c(4,4),d(4,4)
```

```
do i=1,10
  a(i) = b(i)
enddo

do j=1,4
  do i=1,4
    c(i,j) = x*d(i,j)+y
  enddo
enddo
```

είναι ισοδύναμο με

```
integer :: i,j
real    :: x,y,a(10),b(10),c(4,4),d(4,4)

a = b
c = x*d+y
```

Πολλές συναρτήσεις της Fortran έχουν την ιδιότητα να είναι **elemental**, δηλ. στα ορίσματά τους παίρνουν τα ονόματα των arrays και δρουν σε κάθε στοιχείο αυτού χωριστά. Για παράδειγμα οι εντολές

```
integer :: i,j
real    :: x,y,a(10),b(10),c(4,4),d(4,4)

c = sin(d) + x*exp(-2.0*d)
call random_number(a)
```

θέτουν  $c(i,j) = \sin(d(i,j)) + x \cdot \exp(-2.0 \cdot d(i,j))$  για κάθε επιτρεπτή τιμή των  $i,j$ , και τα στοιχεία  $a(i)$  σε έναν διαφορετικό τυχαίο αριθμό μεταξύ του 0 και 1. Για να είναι τα arrays conformable δεν είναι αναγκαστικό να έχουν τα ίδια lower/upper bounds. Για παράδειγμα, η πρώτη εντολή παρακάτω  $b=c*d$  έχει το ίδιο αποτέλεσμα με το do loop

```
integer :: i
real    :: b(0:19), c(10:29), d(-9:10)

b = c*d

do i=1,20
  b(i-1) = c(i+9) * d(i-10)
enddo
```

Παρακάτω δίνουμε μερικές χρήσιμες συναρτήσεις που δρουν σε arrays. Υποθέστε ότι

```
real :: a(-10:10), b(-10:10), c(10,10), d(10,10), e(10,10)
```

τότε

- LBOUND(a) και UBOUND(a) δίνουν τα lower bound και upper bound του array a. Εδώ LBOUND(a) = -10 και UBOUND(a) = 10.
- c = TRANSPOSE(d) θέτει  $c(i, j) = d(j, i)$ .
- e = MATMUL(c, d) θέτει το array e ίσο με το γινόμενο των πινάκων c, d. Δηλ.  $e(i, j) = \sum_{k=1}^{10} c(i, k) * d(k, j)$ . Προσοχή, η εντολή e=c\*d θέτει  $e(i, j) = c(i, j) * d(i, j)$ .
- SUM(a) δίνει το άθροισμα όλων των στοιχείων του a.  
Δηλαδή  $\sum_{i=-10}^{10} a(i)$
- PRODUCT(a) δίνει το γινόμενο όλων των στοιχείων του a.  
Δηλαδή  $\prod_{i=-10}^{10} a(i)$
- DOT\_PRODUCT(a, b) δίνει το εσωτερικό γινόμενο των a, b.  
Δηλαδή  $\sum_{i=-10}^{10} a(i) * b(i)$
- MAXVAL(a), MINVAL(a) δίνουν τη μέγιστη και ελάχιστη τιμή του array a αντίστοιχα.

Για περισσότερες συναρτήσεις και τις πλήρεις οδηγίες για τη χρήση τους δείτε τη βιβλιογραφία [11, 10, 9].

Παρακάτω δίνονται μερικές πληροφορίες σχετικά με την είσοδο / έξοδο (input / output ή I/O) για τα arrays. Η είσοδος (“διάβασμα”) και έξοδος (“εκτύπωση”) των arrays μπορεί να γίνεται απλά διαβάζοντας και εκτυπώνοντας τα στοιχεία τους με τη σειρά που θέλουμε. Για παράδειγμα στο παρακάτω πρόγραμμα διαβάζουμε το array a και εκτυπώνουμε το array b με δύο διαφορετικούς τρόπους:

```
integer :: i, j
real    :: a(4), b(2,2)

do i=1,4
  read *, a(i)
enddo
read *, (a(i), i=1,4)

do j=1,2
  do i=1,2
```

```

    print *,b(i,j)
  enddo
enddo
print *,( (b(i,j), i=1,2), j=1,2)

```

Μέσα στα do loops η είσοδος και έξοδος γίνεται ένα στοιχείο ανά γραμμή. Οι εντολές (a(i), i=1,4) και ( (b(i,j) i=1,2), j=1,2) λέγονται implied do loops και διαβάζουν/εκτυπώνουν από/στην ίδια γραμμή. Αν η γραμμή εξαντληθεί από στοιχεία, τότε το πρόγραμμα συνεχίζει να διαβάζει από την επόμενη. Δοκιμάστε το...

Η είσοδος/έξοδος των arrays μπορεί να γίνει, αντί για implied do loops, χρησιμοποιώντας απλά το όνομα των arrays. Στην περίπτωση αυτή, τα στοιχεία των arrays διαβάζονται/εκτυπώνονται με συγκεκριμένη σειρά. Για παράδειγμα

```

real    :: a(4), b(2,2)

read    *, a
read    *, b

print   *, a,b

```

θα διαβάσει από το stdin πρώτα τις τιμές a(1) a(2) a(3) a(4), θα προχωρήσει στην επόμενη γραμμή (record) και θα διαβάσει τα b(1,1), b(2,1), b(1,2), b(2,2). Προσέξτε, ότι ο πίνακας b θα διαβαστεί κατά στήλες! Η εκτύπωση θα δώσει σε ένα record τα a(1) a(2) a(3) a(4) b(1,1), b(2,1), b(1,2), b(2,2) (πάλι δηλ. το b τυπώνεται κατά στήλες).

Τέλος, χωρίς πολλά λόγια, αλλά παραπέμποντας τον αναγνώστη στη βιβλιογραφία, παραθέτουμε ορισμένες δυνατότητες που δίνει η γλώσσα Fortran στον χειρισμό των arrays. Διαβάστε τα σχόλια στον παρακάτω κώδικα για επεξηγήσεις:

```

program arrays
  implicit none
  integer :: i,j,n,m
  real    :: a(3), b(3,3), c(3,3)=-99.0, d(3,3)=-99.0, s
  integer :: semester(1000), grade(1000)
  logical :: pass(1000)
  !construct the matrix: use the RESHAPE function
  !|1.1  -1.2  -1.3|
  !|2.1   2.2  -2.3|
  !|3.1   3.2   3.3|
  b = RESHAPE((/ 1.1, 2.1, 3.1, & !(notice rows<->columns)

```

```

-1.2, 2.2, 3.2, &
-1.3, -2.3, 3.3 /) ,(/3,3/))
!same matrix, now exchange rows and columns: ORDER=(/2,1/)
b = RESHAPE((/ 1.2, -1.2, -1.3, &
2.1, 2.2, -2.3, &
3.1, 3.2, 3.3 /) ,(/3,3/),ORDER=(/2,1/))
a = b(:,2) !a assigned the second column of b: a(i)=b(i,2)
a = b(1,:) !a assigned the first row of b: a(i)=b(1,i)
a = 2.0*b(:,3)+sin(b(2,:))!a(i)= 2*b(i,3)+sin(b(2,i))
a = 1.0+2.0*exp(-a)+b(:,3)!a(i)= 1+2*exp(-a(i))+b(i,3)
s = SUM(b) !returns sum of all elements of b
s = SUM(b,MASK=(b.gt.0))!returns sum of positive elements of b
a = SUM(b,DIM=1) !each a(i) is the sum of the columns of b
a = SUM(b,DIM=2) !each a(i) is the sum of the rows of b
!repeat all the above using PRODUCT!
!all instructions may be executed in parallel at any order!
FORALL(i=1:3) c(i,i) = a(i) !set the diagonal of c
!compute upper bounds of indices in b:
n=UBOUND(b,DIM=1);m=UBOUND(b,DIM=2)
!log needs positive argument, add a restriction ("mask")
FORALL(i=1:n,j=1:m, b(i,j).gt.0.0 ) c(i,j) = log(b(i,j))
!upper triangular part of matrix:
!careful, j=i+1:m NOT permitted
FORALL(i=1:n,j=1:m, i .lt. j ) c(i,j) = b(i,j)
!each statement executed BEFORE the next one!
FORALL(i=2:n-1,j=2:n-1)
!all right hand side evaluated BEFORE the assignment
!i.e., the OLD values of b averaged and then assigned to b
b(i,j)=(b(i+1,j)+b(i-1,j)+b(i,j+1)+b(i,j-1))/4.0
c(i,j)=1.0/b(i+1,j+1) !the NEW values of b are assigned
END FORALL
! assignment but only for elements b(i,j) which are not 0
WHERE (b .ne. 0.0) c = 1.0/b
!MATMUL(b,c) is evaluated, then d is assigned the result only
!at positions where b>0.
WHERE (b .gt. 0.0) d = MATMUL(b,c)
WHERE (grade .ge. 5 )
semester = semester + 1 !student's semester increases by 1
pass = .true.
ELSEWHERE
pass = .false.
END WHERE
end program arrays

```

Θα βρείτε τον παραπάνω κώδικα στο αρχείο f90\_arrays.f90 του συνοδευτικού λογισμικού.

#### 1.4.4 Ιστορικές Παρατηρήσεις

Η Fortran είναι μία γλώσσα προγραμματισμού που έχει μακρά ιστορία, η οποία ξεκινάει από το 1950.<sup>50</sup> Δύο σημαντικοί σταθμοί στην ιστορία της, που σηματοδοτούν σημαντικές αλλαγές, είναι τα πρότυπα Fortran 77 και Fortran 90. Η μετάβαση από το ένα πρότυπο στο άλλο, σηματοδοτεί και την αλλαγή στον μορφότυπο του προγράμματος, από το λεγόμενο “fixed width format”, όπου οι εντολές πρέπει να γράφονται σε συγκεκριμένες στήλες ενός αρχείου, στο “free format” όπου αυτό δεν είναι πια απαραίτητο.<sup>51</sup> Στο βιβλίο αυτό, τα προγράμματα γράφονται αποκλειστικά στη δεύτερη μορφή, αλλά καλό είναι ο έμπειρος προγραμματιστής της Fortran να γνωρίζει τα βασικά δομικά στοιχεία και της πρώτης μορφής, έτσι ώστε να μπορεί να διαβάζει και να επεξεργάζεται παλιότερα προγράμματα.

Παρόλο που σε κάθε αλλαγή των προτύπων της Fortran εισάγονται νέα στοιχεία, γίνεται μεγάλη προσπάθεια για να υπάρχει αναδρομική συμβατότητα και για να καταλαβαίνουν οι μεταγλωττιστές, μαζί με το νέο, και το παλιό συντακτικό. Έτσι είναι δυνατόν, ένας προγραμματιστής να προσθέσει μεταγενέστερα στοιχεία σε έναν παλιό κώδικα και αυτός να συνεχίσει να τρέχει σωστά. Η πιο μεγάλη πρόσφατη αλλαγή έγινε με το πρότυπο Fortran 90, όπου εισάγονται στοιχεία αντικειμενοστραφούς και δομικού προγραμματισμού. Τα modules, η δυναμική εκχώρηση μνήμης, array section manipulation, operator overloading, pointers, έλεγχος ακρίβειας αριθμητικών πράξεων κλπ, εισάγονται με το πρότυπο αυτό. Στα πρότυπα Fortran 95, 2003 και 2008 γίνονται μικρές αλλαγές, ενισχύοντας κυρίως τα στοιχεία αντικειμενοστραφούς προγραμματισμού και το συντακτικό που ευνοεί την εκτέλεση παράλληλων εντολών. Η επόμενη αλλαγή στη Fortran αναμένεται το 2018, οι αλλαγές που θα γίνουν θα είναι λίγες και μάλλον θα ακούει στο όνομα Fortran 2015.

Στα προγράμματα Fortran που παρουσιάζουμε στο βιβλίο αυτό, χρησιμοποιούμε το free format. Τα προγράμματα που γράφουμε, τότε ακολουθούν ένα στυλ προγραμματισμού κοντύτερα στη φιλοσοφία του προτύπου Fortran 77 (λ.χ. στατική εκχώρηση μνήμης, χρήση common blocks) και τότε πρότυπα της Fortran 90 ή και μετέπειτα (χρήση modules, δυναμική εκχώρηση μνήμης). Ο αναγνώστης πρέπει να κατανοήσει ότι η γλώσσα Fortran είναι κατά το μεγαλύτερο μέρος της ενιαία και ότι

<sup>50</sup><https://en.wikipedia.org/wiki/Fortran>.

<sup>51</sup>Αρχεία με κατάληξη .f περιέχουν κώδικα σε fixed width format και αρχεία με κατάληξη .f90, .f95, .f03, .f08 κώδικα σε free format σύμφωνα με τα πρότυπα Fortran 90, 95, 2003 και 2008, αντίστοιχα.

μπορεί να χρησιμοποιεί το στυλ που βρίσκει απλούστερο ή αποδοτικότερο για το πρόγραμμα που γράφει. Οι μοντέρνοι μεταγλωττιστές, όπως η gfortran, καταλαβαίνουν και μεταγλωττίζουν αποδοτικά όποιο πρότυπο Fortran και αν χρησιμοποιηθεί.

Για τον αναγνώστη που θέλει να ειδικευτεί στην προχωρημένη χρήση της Fortran, συστήνουμε την αναφορά [10]. Για μια εμπειριστατωμένη και καλή εισαγωγή, συστήνουμε την αναφορά [9], αλλά και την παλιότερη [11]. Για να μάθετε για τη Fortran 77, αλλά και το fixed width format, ανατρέξτε στην (ελεύθερα διαθέσιμη) [12].

## 1.5 Κοιτάζοντας τα Αποτελέσματα

Η γραφική απεικόνιση των δεδομένων είναι αναπόσπαστο μέρος της ποιοτικής, αλλά και ποσοτικής κατανόησης της πληροφορίας που περιέχουν. Ένα καλό και ελεύθερα διαθέσιμο πρόγραμμα που παράγει γραφήματα υψηλής ποιότητας στις δύο και τρεις διαστάσεις είναι το gnuplot. Τα ειδικότερα πλεονεκτήματά του έναντι άλλων εφαρμογών είναι η ευελιξία στη χρήση του από τη γραμμή εντολών, αλλά και μέσα από άλλα προγράμματα, καθώς και οι μεγάλες δυνατότητες που δίνει στον χειρισμό και στον μετασχηματισμό των δεδομένων. Έχει τη δικιά του στοιχειώδη γλώσσα προγραμματισμού και, όπου αυτή δεν επαρκεί, μπορούν να γίνουν πολύπλοκες διαδικασίες χρησιμοποιώντας άλλες εφαρμογές. Ο χρήστης έχει απευθείας πρόσβαση σε πολλές μαθηματικές συναρτήσεις και σε συνάρτηση προσαρμογής των δεδομένων (fitting). Διαθέτει διαδραστικά τερματικά όπου με το ποντίκι ο χρήστης μπορεί να μετασχηματίζει τα γραφήματα. Η παράγραφος αυτή είναι εξαιρετικά συνοπτική και παρουσιάζει τα εργαλεία που είναι απολύτως απαραίτητα για τα παρακάτω κεφάλαια. Για περισσότερες πληροφορίες παραπέμπουμε στην ιστοσελίδα του gnuplot <http://gnuplot.info/> και ειδικότερα στη σελίδα με την Demo Gallery <http://gnuplot.info/screenshots/> όπου θα βρείτε αμέσως πώς γίνεται η εργασία που σας ενδιαφέρει, καθώς και στη βιβλιογραφία [14].

Για να ξεκινήσετε το gnuplot δίνετε την εντολή όπως φαίνεται παρακάτω:

```
> gnuplot
```

```
G N U P L O T  
Version X.XX  
....
```

```
The gnuplot FAQ is available from www.gnuplot.info/faq/
....
Terminal type set to 'wxt'
gnuplot>
```

Παραπάνω δείχνεται το μήνυμα καλωσορίσματος και στην τελευταία γραμμή φαίνεται το prompt του προγράμματος. Εκεί, μπορούμε να πληκτρολογήσουμε μία εντολή η οποία εκτελείται πατώντας το Enter. Στη συνέχεια, όταν θα γράφουμε το prompt αυτό, θα υπονοούμε πως το ακολουθούν εντολές που ερμηνεύονται από το gnuplot.

Το γράφημα μιας συνάρτησης γίνεται απλά με την εντολή `plot`. Το σύμβολο `x` εννοείται πως είναι η ανεξάρτητη μεταβλητή<sup>52</sup>. Έτσι, η εντολή

```
gnuplot> plot x
```

κάνει τη γραφική παράσταση της  $y = f(x) = x$  (ευθεία κλίσης 1). Για να κάνουμε ταυτόχρονα τις γραφικές παραστάσεις περισσότερων συναρτήσεων απλά τις γράφουμε μαζί ως εξής:

```
gnuplot> plot [-5:5][-2:4] x, x**2, sin(x), besj0(x)
```

Παραπάνω γίνεται το γράφημα των συναρτήσεων  $x$ ,  $x^2$ ,  $\sin x$ ,  $J_0(x)$ . Στις αγκύλες `[:]` βάζουμε τα όρια της γραφικής παράστασης στον άξονα  $x$  και  $y$  αντίστοιχα. Το `[-5:5]` καθορίζει το  $x$  να μεταβάλλεται από  $-5$  έως  $+5$ , ενώ το `[-2:4]` καθορίζει το  $y$  να μεταβάλλεται από  $-2$  έως  $+4$ . Αν σε κάποιες θέσεις δε βάλουμε αριθμό, τότε το gnuplot βάζει τα όρια αυτόματα: `[1:]` `[:5]` καθορίζει το κάτω όριο το  $x$  να είναι το 1 και το άνω όριο στο  $y$  να είναι το 5, ενώ τα απροσδιόριστα άνω και κάτω όρια αφήνονται στα ... χέρια του gnuplot.

Συχνά, θα θέλουμε να κάνουμε τη γραφική παράσταση δεδομένων που δίνονται από διακριτά ζεύγη  $(x_i, y_i)$ . Τα δεδομένα αυτά τα τοποθετούμε σε αρχεία σε στήλες. Ας υποθέσουμε πως το αρχείο με τα δεδομένα μας ονομάζεται `data` και τα περιεχόμενά του είναι:

```
# x  y1  y2
0.5 1.0 0.779
1.0 2.0 0.607
1.5 3.0 0.472
2.0 4.0 0.368
2.5 5.0 0.287
3.0 6.0 0.223
```

<sup>52</sup>Αλλάζει με την εντολή `set dummy t` για να γίνει λ.χ. `t` η ανεξάρτητη μεταβλητή.



Η πρώτη γραμμή αρχίζει με τον χαρακτήρα # και το gnuplot την αγνοεί (σχόλια για μας). Για να κάνουμε τη γραφική παράσταση της 2ης στήλης συναρτήσει της 1ης δίνουμε απλά την εντολή:

```
gnuplot> plot "data" using 1:2 with points
```

Το όνομα του αρχείου data δίνεται ανάμεσα σε εισαγωγικά, ενώ μετά την εντολή using δίνουμε τις στήλες που θα αντιστοιχούν στον άξονα  $x$  και  $y$  αντίστοιχα (1:2= στήλη 1 τα  $x_i$  και στήλη 2 τα  $y_i$ ). Η εντολή with points αναπαριστά τα ζεύγη  $(x_i, y_i)$  με σημεία.

Η εντολή

```
gnuplot> plot "data" using 1:3 with lines
```

κάνει τη γραφική παράσταση της 3ης στήλης συναρτήσει της 1ης και τα ζεύγη  $(x_i, y_i)$  ενώνονται με ευθύγραμμα τμήματα.

Οι γραφικές παραστάσεις μπορούν να συνδυαστούν:

```
gnuplot> plot "data" using 1:3 with points, exp(-0.5*x)
gnuplot> replot "data" using 1:2
gnuplot> replot 2*x
```

Στην πρώτη γραμμή κάνουμε μαζί τη γραφική παράσταση της 1ης και 3ης στήλης του αρχείου data μαζί με τη συνάρτηση  $e^{-x/2}$ . Στη δεύτερη γραμμή προσθέτουμε με την εντολή replot στην ίδια γραφική παράσταση τα σημεία της 1ης και 3ης στήλης. Και στην 3η βάζουμε μαζί και τη γραφική παράσταση της συνάρτησης  $2x$ .

Η εντολή using έχει πολλές δυνατότητες. Αν αντί για αριθμούς βάλουμε μαθηματικές εκφράσεις ανάμεσα σε παρενθέσεις [δηλ. using (...):( ...)], τότε το gnuplot τις υπολογίζει για κάθε σημείο και βάζει τα αποτελέσματα στη γραφική παράσταση. Για να μπει η τιμή μιας στήλης στη μαθηματική έκφραση χρησιμοποιούμε το ίδιο συντακτικό με την awk, δηλ. \$i αναφέρεται στη στήλη  $i=1,2,3,\dots$ . Παραδείγματα:

```
gnuplot> plot "data" using 1:($2*sin($1)*$3) with points
gnuplot> replot 2*x*sin(x)*exp(-x/2)
```

Κάνει τη γραφική παράσταση της 1ης στήλης με την αντίστοιχη τιμή της έκφρασης  $y_i \sin(x_i) z_i$ , όπου  $x_i, y_i, z_i$  οι τιμές της 1ης, 2ης και 3ης στήλης αντίστοιχα. Η δεύτερη γραμμή τοποθετεί στο σχήμα και τη γραφική

παράσταση της συνάρτησης  $2x \sin(x)e^{-x/2}$ .

```
gnuplot> plot "data" using (log($1)):(log($2**2))
gnuplot> replot 2*x+log(4)
```

Κάνει τη γραφική παράσταση του φυσικού λογαρίθμου της 1ης στήλης με το φυσικό λογάριθμο του τετραγώνου της 2ης.

Με το gnuplot μπορούμε να κάνουμε τη γραφική παράσταση των δεδομένων που τυπώνει στο stdout οποιοδήποτε πρόγραμμα εκτελείται από το φλοιό. Έστω ότι έχουμε ένα πρόγραμμα με όνομα area που τυπώνει στο stdout την ακτίνα και το εμβαδόν ενός κύκλου:

```
> ./area
R=      3.280000      area=      33.79851
R=      6.280000      area=     123.8994
R=      5.280000      area=      87.58257
R=      4.280000      area=      57.54895
```

Τα δεδομένα είναι στην 2η και 4η στήλη του stdout και μπορούμε να τα δούμε γραφικά από το gnuplot με την εντολή:

```
gnuplot> plot "< ./area" using 2:4
```

Δηλαδή στη θέση του ονόματος του αρχείου βάζουμε το όνομα της εντολής με τον χαρακτήρα < να προηγείται. Μπορούμε να συνδυάσουμε εντολές μέσω piping και να παράγουμε πολύπλοκα αποτελέσματα. Α.χ.

```
gnuplot> plot \
  "< ./area | sort -g -k 2 | awk '{print log($2),log($4)}'" \
  using 1:2
```

όπου τα δεδομένα που αναπαρίστανται γραφικά είναι το αποτέλεσμα ενός φίλτρου τριών εντολών: Αυτή που παράγει τα δεδομένα ακτίνα-εμβαδόν όπως παραπάνω, η δεύτερη sort που τα διατάσσει ανάλογα με την αριθμητική τιμή της 2ης στήλης και η τρίτη awk που τυπώνει το λογάριθμο της 2ης στήλης και το λογάριθμο της 4ης. Παρατηρήστε πώς τώρα χρησιμοποιούμε την εντολή using 1:2, αφού η τελευταία εντολή τυπώνει τα δεδομένα σε δύο μόνο στήλες.

Για να σώσουμε τις γραφικές μας παραστάσεις σε αρχεία που μπορούμε να φυλάξουμε και πιθανώς να δημοσιεύσουμε, πρέπει να αλλάξουμε το "terminal" που χρησιμοποιεί ο gnuplot σε έναν οδηγό που μεταφράζει τη γραφική παράσταση σε μία γλώσσα που καταλαβαίνουν

άλλα προγράμματα που δείχνουν εικόνες (λ.χ. PDF, postscript, jpeg, png, gif κλπ). Κατευθύνοντας την “έξοδο” του terminal σε ένα αρχείο πετυχαίνουμε το ζητούμενο. Για παράδειγμα

```
gnuplot> plot "data" using 1:3
gnuplot> set terminal jpeg
gnuplot> set output "data.jpg"
gnuplot> replot
gnuplot> set output
gnuplot> set terminal wxt
```

Η πρώτη γραμμή κάνει τη γραφική παράσταση στο τερματικό, ώστε να τη δούμε. Η δεύτερη καθορίζει πως το γράφημα θα σωθεί σε μορφή JPEG και η τρίτη το όνομα του αρχείου που θα το αποθηκεύσουμε. Στην τέταρτη επαναλαμβάνουμε το τελευταίο γράφημα (εδώ αυτό της 1ης γραμμής) και στην πέμπτη κλείνουμε το αρχείο data.jpg (μην το ξεχάσετε!). Η τελευταία γραμμή επιβάλλει η επόμενη γραφική παράσταση να γίνει πάλι στο τερματικό.

Συνήθως γραφικές παραστάσεις υψηλής ποιότητας αποθηκεύονται στη γλώσσα PDF. Επιλέξτε `set terminal pdf` και `set output "data.pdf"` στην περίπτωση αυτή.

Λίγα λόγια για τις τρισδιάστατες γραφικές παραστάσεις. Οι επόμενες εντολές δείχνουν πώς με την εντολή `splot` μπορείτε να δείτε τη γραφική παράσταση της συνάρτησης  $f(x, y) = e^{-x^2-y^2}$ . Με το ποντίκι μπορείτε να την περιστρέψτε και να τη δείτε υπό διαφορετική γωνία.

```
gnuplot> set pm3d
gnuplot> set hidden3d
gnuplot> set size ratio 1
gnuplot> set isosamples 50
gnuplot> splot [-2:2][-2:2] exp(-x**2-y**2)
```

Αν έχετε δεδομένα στη μορφή  $(x_i, y_i, z_i)$  και θέλετε να τα αναπαραστήσετε γραφικά στη μορφή  $z_i = f(x_i, y_i)$ , τακτοποιήσετε τα σε ένα αρχείο της μορφής

```
-1 -1 2.000
-1  0 1.000
-1  1 2.000

 0 -1 1.000
 0  0 0.000
 0  1 1.000
```

```
1 -1 2.000
1 0 1.000
1 1 2.000
```

Προσέξτε πως βάζουμε μία κενή γραμμή κάθε φορά που αλλάζει η τιμή του  $x$ . Αν ονομάσετε το αρχείο αυτό `data3`, δείτε τη γραφική παράσταση με τις εντολές:

```
gnuplot> set pm3d
gnuplot> set hidden3d
gnuplot> set size ratio 1
gnuplot> splot "data3" with lines
```

Κλείνουμε με δύο λόγια για τη γραφική παράσταση που δίνεται από παραμετρικές εξισώσεις. Στις δύο διαστάσεις θεωρούμε τις καμπύλες  $(x(t), y(t))$  και στις τρεις τις επιφάνειες  $(x(u, v), y(u, v), z(u, v))$ . Με τις παρακάτω εντολές κάνουμε τη γραφική παράσταση του κύκλου  $(\sin t, \cos t)$  και της σφαίρας  $(\cos u \cos v, \cos u \sin v, \sin u)$ :

```
gnuplot> set parametric
gnuplot> plot sin(t), cos(t)
gnuplot> splot cos(u)*cos(v), cos(u)*sin(v), sin(u)
```

## 1.6 Shell Scripting: Σενάρια Φλοιού

Η γλώσσα Fortran θα φανεί σε κάποιον που έχει συνηθίσει κάποια γλώσσα με περισσότερες δυνατότητες (C, C++, Java, ...) πως είναι δύσκολη, όταν κάποιος θέλει να κάνει πολύπλοκες διαδικασίες που αφορούν το σύστημα και δε θα είχε άδικο. Όταν όμως χρησιμοποιήσει τα προγράμματα που γράφει σε Fortran σε συνδυασμό με τα πανίσχυρα εργαλεία που του παρέχει το λειτουργικό σύστημα, τα προβλήματα αυτά ξεπερνιούνται και έτσι μπορεί κανείς να χρησιμοποιήσει τα πλεονεκτήματα της γλώσσας σε high performance computing, χωρίς να ανησυχεί για τις διαχειριστικές και συχνά τετριμμένες εργασίες.

Για να αποφύγει κανείς την επαναλαμβανόμενη διαδικασία εκτέλεσης των ίδιων εντολών (που εμπεριέχει και τον κίνδυνο σφάλματος), μπορεί τις εντολές που θέλει να δώσει να τις κωδικοποιήσει μέσα σε ένα αρχείο. Αυτό ονομάζεται σενάριο φλοιού (shell script) και η πιο απλή μορφή του μπορεί να είναι απλά μια σειρά από εντολές. Γράφουμε στο αρχείο `script01.csh`:

```
#!/bin/tcsh -f
gfortran area_01.f90 -o area
./area
gfortran area_02.f90 -o area
./area
gfortran area_03.f90 -o area
./area
gfortran area_04.f90 -o area
./area
```

Η πρώτη γραμμή (ακριβώς!!) αρχίζει με `#!/bin/tcsh -f` που ερμηνεύεται από το λειτουργικό σύστημα, ώστε να εκτελεστούν οι εντολές από τον φλοιό `/bin/tcsh`<sup>53</sup>. Στη συνέχεια γράφουμε τις εντολές μεταγλώττισης και εκτέλεσης των προγραμμάτων που μελετήσαμε στην προηγούμενη παράγραφο. Αρχικά, κάνουμε το αρχείο εκτελέσιμο με την εντολή<sup>54</sup>

```
> chmod u+x script01.csh
```

και στη συνέχεια με την εντολή

```
> ./script01.csh
```

τρέχουν όλες οι παραπάνω εντολές η μία μετά την άλλη. Όλα ωραία, εκτός από το γεγονός ότι πρέπει να δίνουμε τις 10 ακτίνες του κύκλου στα προγράμματα `./area` κάθε φορά που τις ζητούν. Μια λύση είναι να γράφουμε τα δεδομένα εισόδου (τις ακτίνες) σε ένα αρχείο `Input` και να δώσουμε την εντολή

```
./area < Input
```

οπότε δεν χρειάζεται να παρέχουμε τα δεδομένα διαδραστικά. Υπάρχει και πιο συμπαγής λύση, να βάλουμε τα περιεχόμενα των δεδομένων σε ένα “Here Document”, ένα “αρχείο” το οποίο ο χρήστης μπορεί να φανταστεί ότι υπάρχει όταν τρέχει το script, αλλά δεν δε θα το δει ποτέ στο filesystem! Το συντακτικό, λίγο στρυφνό για αρχή, αλλά συνηθίζεται (και γίνεται και εθισμός...) είναι ως εξής (στο αρχείο `script02.csh`):

```
#!/bin/tcsh -f
gfortran area_04.f90 -o area
```

<sup>53</sup>Γράψτε `#!/bin/bash` αν χρησιμοποιείτε τον φλοιό `bash`.

<sup>54</sup>Αυτό το κάνουμε μόνο μία φορά!

```
./area <<EOF
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
EOF
```

δηλ. το πρόγραμμα ./area παίρνει stdin από τα περιεχόμενα μεταξύ των γραμμών<sup>55</sup>:

```
./area <<EOF
...
EOF
```

Δεν υπάρχει τίποτα το ιδιαίτερο με το string “EOF” και μπορείτε να βάλετε όποιο σας αρέσει (αρκεί να είναι το ίδιο στην αρχή και το τέλος).

Η δύναμη του shell scripting είναι οι ικανότητες προγραμματισμού που παρέχει: Ορισμός μεταβλητών, loops, conditionals, ... Οι μεταβλητές ορίζονται όπως οι μεταβλητές φλοιού που αναφέραμε στην παράγραφο 1.1.2. Η τιμή μιας μεταβλητής με όνομα name είναι \$name και μπορούμε να τη θέσουμε με την εντολή set name = value. Ένα array μπορεί να οριστεί με την εντολή

```
set R = (1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0)
```

και η πρόσβαση στα δεδομένα γίνεται με το συντακτικό \$R[1] ... \$R[10]

Ας δούμε τώρα ένα πιο ... προχωρημένο σενάριο:

```
#!/bin/tcsh -f

set files = (area_01.f90 area_02.f90 area_03.f90 area_04.f90)
set R      = (1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0)

echo "Hello $USER Today is " `date`
foreach file ($files)
  echo "# _____ Working on file $file "
```

<sup>55</sup>Το EOF μπορεί να είναι οποιαδήποτε ακολουθία χαρακτήρων.

```

gfortran $file -o area
./area <<EOF
$R[1]
$R[2]
$R[3]
$R[4]
$R[5]
$R[6]
$R[7]
$R[8]
$R[9]
$R[10]
EOF
echo "# _____ Done "
if( -f AREA.DAT ) cat AREA.DAT
end

```

Οι πρώτες γραμμές με τις εντολές set θέτουν τις τιμές των μεταβλητών files (4 τιμές) και R (10 τιμές). Η εντολή echo απλά “αντηχεί” στο stdout το όρισμά της. Εδώ ο φλοιός αναπτύσσει στο όρισμα "Hello \$USER Today is " `date` την τιμή της μεταβλητής USER η οποία είναι μεταβλητή περιβάλλοντος που το λειτουργικό σύστημα θέτει να είναι το όνομα χρήστη. Στη συνέχεια στη το `date` αντικαθίσταται από το stdout της εντολής date, λ.χ. Thu May 24 22:01:40 EEST 2007.

Στη συνέχεια, αρχίζει το foreach loop:

```

foreach file ($files)
...
end

```

Η μεταβλητή \$files αναπτύσσεται στις 4 τιμές της (τα ονόματα των αρχείων Fortran area\_01.f90, area\_02.f90, area\_03.f90, area\_04.f90) και ο βρόχος εκτελείται μια φορά για κάθε τιμή. Κάθε φορά η τιμή της μεταβλητής file είναι η εκάστοτε τιμή της files. Άρα, η εντολή gfortran \$file -o area θα μεταγλωττίσει κάθε φορά ένα από τα παραπάνω 4 αρχεία και στη συνέχεια, θα εκτελέσει το εκάστοτε πρόγραμμα ./area.

Η τελευταία γραμμή στο βρόχο

```

if( -f AREA.DAT ) cat AREA.DAT

```

είναι ένα if-conditional: Εκτελεί την εντολή cat AREA.DAT, μόνο αν η συνθήκη -f AREA.DAT είναι αληθής, δηλ. το αρχείο AREA.DAT υπάρχει.

Τέλος, δίνουμε ένα παραδειγματικό script όπου μπορείτε να δείτε

με παραδείγματα τις δυνατότητες που μπορεί να προσφέρει το shell scripting. Φυσικά είναι μόνο η αρχή, διαβάστε τη βιβλιογραφία για περισσότερες λεπτομέρειες [15, 16, 17, 18, 19]. Διαβάστε προσεκτικά τις εντολές μαζί με τα σχόλια τα οποία αρχίζουν με το χαρακτήρα “#”. Στη συνέχεια, αφού γράψετε τις εντολές σε ένα αρχείο script04.csh<sup>56</sup>, μετατρέψτε το σε εκτελέσιμο με την εντολή `chmod u+x script04.csh` και εκτελέστε την εντολή

```
> ./script04.csh This is my first serious tcsh script
```

Το script θα τρέξει έχοντας ως arguments τη φράση “This is my first serious tcsh script”. Θα δείτε μέσα στο script πώς να την επεξεργαστείτε. Στη συνέχεια, το script θα σας ζητήσει δέκα ή παραπάνω ακτίνες κύκλων για να υπολογίσει την περίμετρο και το εμβαδόν τους. Πληκτρολογήστε τις και μετά διαβάστε την έξοδο του script για να καταλάβετε το αποτέλεσμα των εντολών που περιέχει. Θα είναι μια διαδικασία για την οποία δε θα μετανιώσετε να αφιερώσετε λίγο χρόνο!

```
#!/bin/tcsh -f
# Run this script as:
# ./script04.csh Hello this is a tcsh script
#
# 'command' is command substitution: it is replaced by stdout of command
set now = `date` ; set mypc = `uname -a`
# Print information: variables are expanded within double quotes
echo "I am user $user working on the computer $HOST" #HOST is predefined
echo "Today the date is      : $now"                #now is defined above
echo "My home directory is   : $home"               #home is predefined
echo "My current directory is: $cwd"                #cwd changes with cd
echo "My computer runs       : $mypc"               #mypc is defined above
echo "My process id is      : $$"                  #$$ is predefined
# Manipulate the command line: ($#argv is number of elements in array argv)
echo "The command line has $#argv arguments"
echo "The name of the command I am running is: $0"
echo "Arguments 3rd to last of the command : $argv[3-]" #third to last
echo "The last argument is                  : $argv[$#argv]" #last element
echo "All arguments                        : $argv"

# Ask user for input: enter radii of circles
echo -n "Enter radii of circles: " # variable $< stores one line of input
set Rs = ($<) #Rs is now an array with all words entered by user
if ($#Rs < 10 )then #make a test, need at least 10 of them
    echo "Need more than 10 radii. Exiting...."
    exit(1)
endif
echo "You entered $#Rs radii, the first is $Rs[1] and the last $Rs[$#Rs]"
echo "Rs= $Rs"
# Now, compute the perimeter of each circle:
foreach R ($Rs)
    # -v rad=$R set the awk variable rad equal to $R. pi=atan2(0,-1)=3.14...
```

<sup>56</sup>Θα το βρείτε και στο συνοδευτικό λογισμικό.

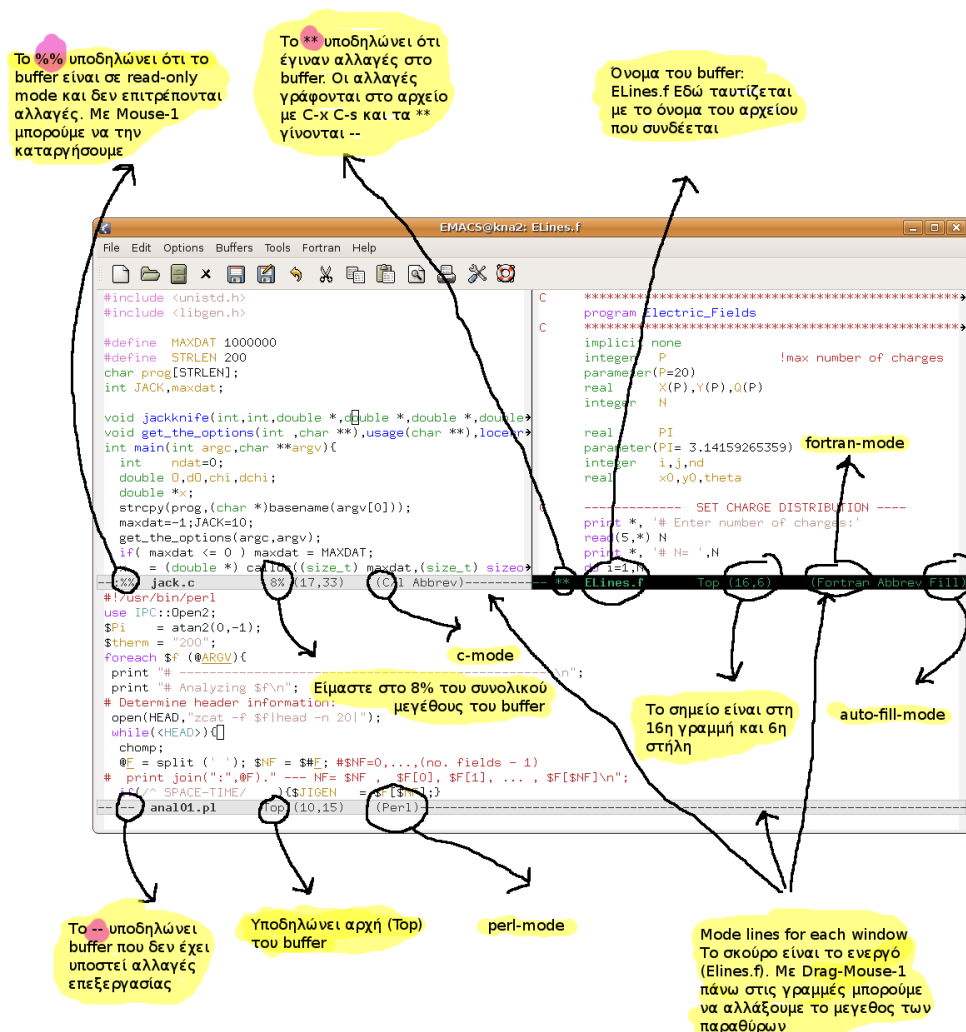


```

set l = 'awk -v rad=$R 'BEGIN{print 2*atan2(0,-1)*rad}''
echo "Circle with R= $R has perimeter $l"
end
# alias defines a command to do what you want: use awk as a calculator
alias acalc 'awk "BEGIN{print \!* }"' # \!* substitutes args of acalc
echo "Using acalc to compute 2+3=" 'acalc 2+3'
echo "Using acalc to compute cos(2*pi)=" 'acalc cos(2*atan2(0,-1))'
# Now do the same loop over radii as above in a different way
# while( expression ) is executed as long as "expression" is true
while($#Rs > 0) #executed as long as $Rs contains radii
  set R = $Rs[1] #take first element of $Rs
  shift Rs #now $Rs has one less element:old $Rs[1] has vanished
  set a = 'acalc atan2(0,-1)*${R}*${R}' # =pi*R*R calculated by acalc
  # construct a filename to save the result from the value of R:
  set file = area${R}.dat
  echo "Circle with R= $R has area $a" > $file #save result in a file
end #end while
# Now look for our files: save their names in an array files:
set files = ('ls -l area*.dat')
if( $#files == 0) echo "Sorry, no area files found"
echo "_____ "
echo "files: $files"
ls -l $files
echo "_____ "
echo "And the results for the area are:"
foreach f ($files)
  echo -n "file ${f}: "
  cat $f
end
# now play a little bit with file names:
echo "_____ "
set f = $files[1] # test permissions on first file
# -f, -r, -w, -x, -d test existence of file , rwx permissions
# the ! negates the expression (true -> false , false -> true)
echo "testing permissions on files:"
if( -f $f ) echo "$file exists"
if( -r $f ) echo "$file is readable by me"
if( -w $f ) echo "$file is writable by me"
if(! -w /bin/ls) echo "/bin/ls is NOT writable by me"
if(! -x $f ) echo "$file is NOT an executable"
if( -x /bin/ls) echo "/bin/ls is executable by me"
if(! -d $f ) echo "$file is NOT a directory"
if( -d /bin ) echo "/bin is a directory"
echo "_____ "
# transform the name of a file
set f = $cwd/$f # add the full path in $f
set filename = $f:r # removes extension .dat
set extension = $f:e # gets extension .dat
set fdir = $f:h # gets directory of $f
set base = 'basename $f' # removes directory name
echo "file is: $f"
echo "filename is: $filename"
echo "extension is: $extension"
echo "directory is: $fdir"
echo "basename is: $base"
# now transform the name to one with different extension:
set newfile = ${filename}.jpg
echo "jpeg name is: $newfile"
echo "jpeg base is:" 'basename $newfile'
if($newfile:e == jpg)echo 'basename $newfile' " is a picture"
echo "_____ "
# Now save all data in a file using a "here document"

```

```
# A here document starts with <<EOF and ends with a line
# starting exactly with EOF (EOF can be any string as below)
# In a "here document" we can use variables and command
# substitution:
cat <<AREAS >> areas.dat
# This file contains the areas of circle of given radii
# Computation done by ${user} on ${HOST}. Today is 'date'
'cat $files'
AREAS
# now see what we got:
if( -f areas.dat) cat areas.dat
# You can use a "here document" as standard input to any command:
# use gnuplot to save a plot: gnuplot does the job and exits...
gnuplot <<GNU
set terminal jpeg
set output "areas.jpg"
plot "areas.dat" using 4:7 title "areas.dat",\
    pi*x*x title "pi*R^2"
set output
GNU
# check our results: display the jpeg file using eog
if( -f areas.jpg) eog areas.jpg &
```



Σχήμα 1.5: Το παράθυρο του Emacs χωρίστηκε εδώ σε τρία παράθυρα. Ο χωρισμός έγινε πρώτα οριζόντια (C-x 2) και μετά κάθετα (C-x 3). Σέρνοντας το ποντίκι Drag-Mouse-1 πάνω στις οριζόντιες (mode lines) και κάθετες διαχωριστικές γραμμές μπορούμε να αλλάξουμε τα μεγέθη των παραθύρων. Επισημαίνουμε τις χρήσιμες πληροφορίες που βρίσκει κανείς στο mode line κάθε παραθύρου. Κάθε παράθυρο έχει το σημείο του (point) και ο δρομέας (cursor) βρίσκεται στο ενεργό παράθυρο (εδώ στο ELines.f). Παρατηρήστε πώς σημειώνεται το ανέπαφο buffer (--), το επεξεργασμένο (\*\*) και αυτό που είναι σε read only mode (%). Με κλικ με Mouse-1 πάνω στα % τα αλλάζουμε σε -- και το αντίστροφο. Με δεξί κλικ Mouse-3 πάνω στο όνομα της mode μπορούμε να ενεργοποιήσουμε επιλογή από minor modes. Με αριστερό κλικ Mouse-1 έχουμε πρόσβαση σε εντολές σχετικές με την mode. Οι αριθμοί (17,31), (16,6) και (10,15) στα mode lines υποδηλώνουν τη (γραμμή, στήλη) που βρίσκεται το σημείο στα αντίστοιχα παράθυρα.

awk	search for and process patterns in a file,
cat	display, or join, files
cd	change working directory
chmod	change the access mode of a file
cp	copy files
date	display current time and date
df	display the amount of available disk space
diff	display the differences between two files
du	display information on disk usage
echo	echo a text string to output
find	find files
grep	search for a pattern in files
gzip	compress files in the gzip (.gz) format (gunzip to uncompress)
head	display the first few lines of a file
kill	send a signal (like KILL) to a process
locate	search for files stored on the system (faster than find)
less	display a file one screen at a time
ln	create a link to a file
lpr	print files
ls	list information about files
man	search information about command in man pages
mkdir	create a directory
mv	move and/or rename a file
ps	report information on the processes run on the system
pwd	print the working directory
rm	remove (delete) files
rmdir	remove (delete) a directory
sort	sort and/or merge files
tail	display the last few lines of a file
tar	store or retrieve files from an archive file
top	dynamic real-time view of processes
wc	counts lines, words and characters in a file
whatis	list man page entries for a command
where	show where a command is located in the path (alternatively: whereis)
which	locate an executable program using "path"
zip	create compressed archive in the zip format (.zip)
unzip	get contents of zip archive

Πίνακας 1.1: Σύνοψη βασικών εντολών στο Unix.

Πίνακας 1.2: Μερικές βασικές συναρτήσεις (intrinsic functions) της Fortran.

Συνάρτηση	Περιγραφή
ABS	το μέτρο ενός μιγαδικού αριθμού, απόλυτη τιμή πραγματικού
ACOS	τόξο συνημιτόνου
ADJUSTL	μετακινεί τους μη κενούς χαρακτήρες μεταβλητής character προς τα αριστερά
ADJUSTR	μετακινεί τους μη κενούς χαρακτήρες μεταβλητής character προς τα δεξιά
AIMAG	φανταστικό μέρος μιγαδικού αριθμού
AINT	περικόπτει το κλασματικό μέρος αριθμού και διατηρεί τον τύπο της μεταβλητής
ANINT	στρογγυλοποιεί στον πλησιέστερο ακέραιο και διατηρεί τον τύπο της μεταβλητής
ASIN	τόξο ημιτόνου
ATAN	τόξο εφαπτομένης
ATAN2	τόξο εφαπτομένης του $\arg1$ διαιρεμένου με το $\arg2$ και τοποθετημένο στο σωστό τεταρτημόριο
CMPLX	μετατρέπει σε τύπο COMPLEX το $\arg1 + i \arg2$
CONJG	μιγαδικό συζυγές
COS	συνημίτονο γωνίας μετρημένης σε radians
COSH	υπερβολικό συνημίτονο
DATE_AND_TIME	τρέχουσα ημερομηνία και ώρα
DBLE	μετατρέπει μια μεταβλητή σε τύπο DOUBLE PRECISION
DIM	αν $\arg1 > \arg2$ , επιστρέφει $\arg1 - \arg2$ ; αλλιώς 0
DPROD	γινόμενο σε ακρίβεια double precision δύο αριθμών single precision
EXP	εκθετική συνάρτηση
EPSILON	δίνει ένα θετικό αριθμό που είναι αμελητέος συγκρινόμενος με 1.0
HUGE	δίνει το μεγαλύτερο αριθμό ίδιου τύπου με το όρισμα
INT	μετατρέπει σε INTEGER κόβοντας το κλασματικό μέρος

Συνεχίζεται στην επόμενη σελίδα...

Πίνακας 1.2: Συνέχεια...

Συνάρτηση	Περιγραφή
KIND	δίνει την KIND τιμή του ορίσματος
LEN	επιστρέφει το μήκος μιας μεταβλητής character
LEN_TRIM	δίνει το μήκος μιας μεταβλητής character χωρίς τους τελικούς κενούς χαρακτήρες
LGE, LGT, LLE, LLT	συναρτήσεις σύγκρισης μεταβλητών character
LOG	φυσικός λογάριθμος
LOG10	κοινός λογάριθμος
MAX	μέγιστη τιμή των ορισμάτων
MAXEXPONENT	ο μέγιστος εκθέτης ίδιου τύπου με το όρισμα
MIN	ελάχιστη τιμή των ορισμάτων
MINEXPONENT	ο ελάχιστος εκθέτης ίδιου τύπου με το όρισμα
MOD	arg1 modulo arg2
NINT	μετατρέπει το όρισμα σε INTEGER με στρογγυλοποίηση
RANDOM_NUMBER	δίνει ψευδοτυχαίους αριθμούς $0 \leq r < 1$
RANDOM_SEED	ξεκινά τη γεννήτρια ψευδοτυχαίων αριθμών ή δίνει τις παραμέτρους κατάστασής της
PRECISION	επιστρέφει τη δεκαδική ακρίβεια ίδιου τύπου με το όρισμα
REAL	το πραγματικό μέρος μιγαδικού αριθμού
REAL	μετατροπή του ορίσματος σε REAL
SIGN	αν $\arg2 < 0$ , δίνει $-\arg1$ ; αλλιώς $+\arg1$
SIN	ημίτονο γωνίας σε radians
SINH	υπερβολικό ημίτονο
SQRT	τετραγωνική ρίζα
TAN	εφαπτομένη γωνίας σε radians
TANH	υπερβολική εφαπτομένη
TINY	δίνει το μικρότερο θετικό αριθμό ίδιου τύπου με το όρισμα
TRIM	επιστρέφει την τιμή μεταβλητής character χωρίς τους τελικούς κενούς χαρακτήρες

Συνεχίζεται στην επόμενη σελίδα...

Πίνακας 1.2: Συνέχεια...

Συνάρτηση	Περιγραφή
Συναρτήσεις για επεξεργασία arrays	
ALL	δίνει .TRUE. αν όλες οι τιμές στο όρισμα είναι .TRUE.
ALLOCATED	array allocation status
ANY	δίνει .TRUE. αν όλες κάποιες τιμές στο όρισμα είναι .TRUE.
COUNT	αριθμός στοιχείων ενός array
DOT_PRODUCT	εσωτερικό γινόμενο δύο rank-one arrays
LBOUND	το μικρότερο όριο μίας διάστασης ενός array
MATMUL	πολλαπλασιασμός πινάκων
MAXLOC	θέση της μεγίστης τιμής σε ένα array
MAXVAL	μέγιστη τιμή σε ένα array
MERGE	συγχώνευση δύο arrays με εφαρμογή mask
MINLOC	θέση ελάχιστης τιμής σε ένα array
MINVAL	ελάχιστη τιμή σε ένα array
PACK	επανατοποθέτηση ενός array σε ένα array με rank ίσο με 1 με εφαρμογή mask
PRODUCT	γινόμενο τιμών ενός array
RESHAPE	αναδιαμόρφωση ενός array
SHAPE	shape ενός array ή scalar
SIZE	μέγεθος ενός array
SPREAD	αντιγραφή ενός array προσθέτοντας μία διάσταση
SUM	άθροισμα τιμών ενός array
TRANPOSE	ανάστροφος πίνακας ενός array με rank δύο
UBOUND	το μεγαλύτερο όριο μίας διάστασης ενός array
UNPACK	επανατοποθέτηση ενός array με rank ένα σε ένα array με εφαρμογή mask

Πίνακας 1.3: Περίληψη βασικών εντολών στο Emacs.

Έξοδος από τον Emacs		
αναστολή Emacs (ή εικονοποίηση σε παράθυρο)	C-z	
τελική έξοδος από Emacs	C-x C-c	
Αρχεία		
εισαγωγή αρχείου στον Emacs	C-x C-f	
εγγραφή αρχείου πίσω στον δίσκο	C-x C-s	
εγγραφή όλων των αρχείων στον δίσκο	C-x s	
εισαγωγή περιεχομένων αρχείου στο buffer	C-x i	
εναλλαγή read-only status ενός buffer	C-x C-q	
Βοήθεια		
Το σύστημα βοήθειας είναι απλό. Πληκτρολογήστε C-h (ή F1) και ακολουθήστε τις οδηγίες. Αν είστε καινούργιος χρήστης, πληκτρολογήστε C-h t για να ακολουθήσετε τις οδηγίες ενός εκπαιδευτικού εγχειρίδιου.		
κλείσιμο παράθυρου βοήθειας	C-x 1	
apropos: εμφάνιση εντολών που ταιριάζουν με λέξεις	C-h a	
περιγραφή συνάρτησης εκτελείται από ένα συγκεκριμένο πλήκτρο	C-h k	
περιγραφή συνάρτησης	C-h f	
πληροφορίες ειδικές για την επιλεγμένη mode	C-h m	
Ανάκτηση λόγω σφάλματος		
διακοπή εντολής που εκτελείται	C-g	
ανάκτηση αρχείων που χάθηκαν λόγω κατάρρευσης του συστήματος	M-x recover-session	
αναίρεση ανεπιθύμητης αλλαγής	C-x u, C-_ or C-/	
επαναφορά ενός buffer στα αρχικά του περιεχόμενα	M-x revert-buffer	
επανασχεδιασμός οθόνης	C-l	
Αυξητική Αναζήτηση		
αναζήτηση προς τα εμπρός	C-s	
αναζήτηση προς τα πίσω	C-r	
αναζήτηση χρησιμοποιώντας regular expression	C-M-s	
διακοπή τρέχουσας αναζήτησης	C-g	
Πληκτρολογήστε C-s ή C-r ξανά για να επαναλάβετε την προηγούμενη αναζήτηση.		
Κίνηση		
οντότητα	forward	backwards
χαρακτήρας	C-b	C-f
λέξη	M-b	M-f
γραμμή	C-p	C-n
αρχή (τέλος) γραμμής	C-a	C-e
αρχή (τέλος) buffer	M-<	M->

Συνεχίζεται στην επόμενη σελίδα...



Πίνακας 1.3: Συνέχεια...

μετακίνηση στην επόμενη οθόνη	C-v	
μετακίνηση στην προηγούμενη οθόνη	M-v	
μετακίνηση αριστερά	C-x <	
μετακίνηση δεξιά	C-x >	
μετακίνηση τρέχουσας γραμμής στο κέντρο της οθόνης	C-u C-l	
<b>Διαγραφή</b>		
<b>οντότητα προς διαγραφή</b>	<b>forward</b>	<b>backwards</b>
χαρακτήρας	DEL	C-d
λέξη	M-DEL	M-d
γραμμή	M-0 C-k	C-k
region	C-w	
αντιγραφή region	M-w	
επικόλληση τελευταίας αντιγραφής	C-y	
αντικατάσταση τελευταίας επικόλλησης με πιο προηγούμενη αντιγραφή	M-y	
<b>Marking</b>		
θέσε το mark εδώ	C-@ or C-SPC	
αντάλλαξε το point και το mark	C-x C-x	
επιλογή παραγράφου	M-h	
επιλογή ολοκλήρου buffer	C-x h	
<b>Διαδραστική Αντικατάσταση</b>		
αντικατάσταση κειμένου διαδραστικά	M-% or M-x query-replace	
αντικατάσταση με χρήση regular expressions	M-x query-replace-regexp	
<b>Buffers</b>		
επιλογή ενός άλλου buffer	C-x b	
λίστα όλων των buffers	C-x C-b	
διαγραφή buffer	C-x k	
<b>Πολλαπλά Παράθυρα</b>		
Όταν δίνονται δύο εντολές, η δεύτερη αφορά frames και όχι windows.		
διαγραφή των άλλων παραθύρων	C-x 1	C-x 5 1
διαχωρισμός παραθύρου οριζόντια	C-x 2	C-x 5 2
διαγραφή τρέχοντος παραθύρου	C-x 0	C-x 5 0
διαχωρισμός παραθύρου κάθετα	C-x 3	
τοποθέτηση cursor στο άλλο παράθυρο	C-x o	C-x 5 o
μεγάλωσε το παράθυρο κατά ύψος	C-x ^	
μίκρυνε το παράθυρο κατά πλάτος	C-x {	
μεγάλωσε το παράθυρο κατά πλάτος	C-x }	

Συνεχίζεται στην επόμενη σελίδα...

Πίνακας 1.3: Συνέχεια...

<b>Μορφοποίηση</b>	
στοίχιση γραμμής	TAB
εισαγωγή νέας γραμμής μετά το point	C-o
γέμισμα παραγράφου	M-q
<b>Αλλαγή Κεφαλαίων/Μικρών</b>	
γραφή λέξης με όλα κεφαλαία	M-u
γραφή λέξης με όλα μικρά	M-l
πρώτο γράμμα κεφαλαίο, τα άλλα μικρά	M-c
γραφή επιλεγμένης region με κεφαλαία	C-x C-u
γραφή επιλεγμένης region με μικρά	C-x C-l
<b>To Minibuffer</b>	
Τα επόμενα πλήκτρα λειτουργούν έτσι μόνο στο minibuffer.	
συμπλήρωση έκφρασης	TAB
συμπλήρωση έκφρασης μέχρι μία λέξη	SPC
συμπλήρωση έκφρασης και εκτέλεση εντολής	RET
διακοπή εντολής που εκτελείται	C-g
Πληκτρολογήστε C-x ESC ESC για την επανάληψη της προηγούμενης εντολής με πιθανή μεταβολή της. Πληκτρολογήστε F10 για την ενεργοποίηση των μενού.	
<b>Ορθογραφικός Έλεγχος</b>	
τρέχουσας λέξης	M-\$
επιλεγμένης region	M-x ispell-region
ολοκλήρου buffer	M-x ispell-buffer
διαρκής ορθογραφικός έλεγχος	M-x flyspell-mode
<b>Info – Διαδραστική Βοήθεια στον Emacs</b>	
έναρξη συστήματος τεκμηρίωσης Info	C-h i
κίνηση εμπρός	SPC
κίνηση πίσω	DEL
επόμενος κόμβος	n
προηγούμενος κόμβος	p
μετακίνηση προς τα επάνω	u
επιλογή επιλογής στο μενού με το όνομά της	m
επιστροφή στον προηγούμενο κόμβο που βρισκόσαστε	l
επιστροφή στα περιεχόμενα	d
κίνηση στον αρχικό κόμβο του αρχείου Info	t
επιλογή κόμβου με το όνομά του	g
έξοδος από Info	q

## ΚΕΦΑΛΑΙΟ 2

### Περιγραφή της Κίνησης

Στο κεφάλαιο αυτό θα δείξουμε πώς να προγραμματίσουμε απλές εξισώσεις τροχιάς ενός σωματιδίου και πώς να κάνουμε βασική ανάλυση των αριθμητικών αποτελεσμάτων. Χρησιμοποιούμε απλές μεθόδους απεικόνισης των τροχιών. Στην παράγραφο 2.3 μελετάμε την επίδραση των συστηματικών σφαλμάτων που υπεισέρχονται σε απλά αριθμητικά πρότυπα της κίνησης σωματιδίου που σκεδάζεται πάνω σε σκληρά και αμετακίνητα εμπόδια. Αυτό θα αποτελέσει και ένα μικρό προελούδιο στη μελέτη ολοκλήρωσης διαφορικών εξισώσεων που θα εξετάσουμε σε επόμενα κεφάλαια.

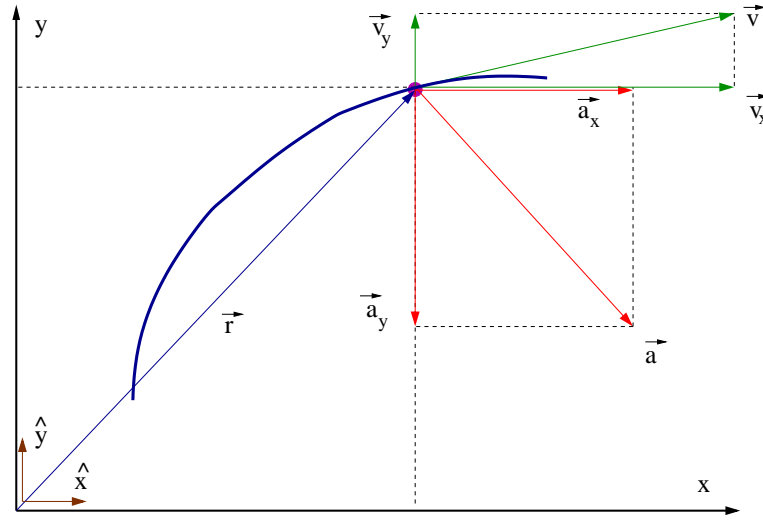
#### 2.1 Κίνηση στο Επίπεδο

Σωματίδιο κινείται στο επίπεδο και η θέση του περιγράφεται σε ένα καρτεσιανό σύστημα συντεταγμένων από τις συντεταγμένες  $(x(t), y(t))$  η οποία ως συνάρτηση του χρόνου δίνει την εξίσωση της τροχιάς του σωματιδίου. Το διάνυσμα θέσης του σωματιδίου είναι το  $\vec{r}(t) = x(t)\hat{x} + y(t)\hat{y}$ , όπου  $\hat{x}$  και  $\hat{y}$  είναι τα μοναδιαία διανύσματα στους άξονες  $x$  και  $y$  αντίστοιχα. Το διάνυσμα της ταχύτητας είναι το  $\vec{v}(t) = v_x(t)\hat{x} + v_y(t)\hat{y}$  όπου

$$\begin{aligned} \vec{v}(t) &= \frac{d\vec{r}(t)}{dt} \\ v_x(t) &= \frac{dx(t)}{dt} \quad v_y(t) = \frac{dy(t)}{dt}. \end{aligned} \quad (2.1)$$

Η επιτάχυνση  $\vec{a}(t) = a_x(t) \hat{x} + a_y(t) \hat{y}$  δίνεται από τις σχέσεις

$$\begin{aligned} \vec{a}(t) &= \frac{d\vec{v}(t)}{dt} = \frac{d^2\vec{r}(t)}{dt^2} \\ a_x(t) &= \frac{dv_x(t)}{dt} = \frac{d^2x(t)}{dt^2} \quad a_y(t) = \frac{dv_y(t)}{dt} = \frac{d^2y(t)}{dt^2}. \end{aligned} \quad (2.2)$$

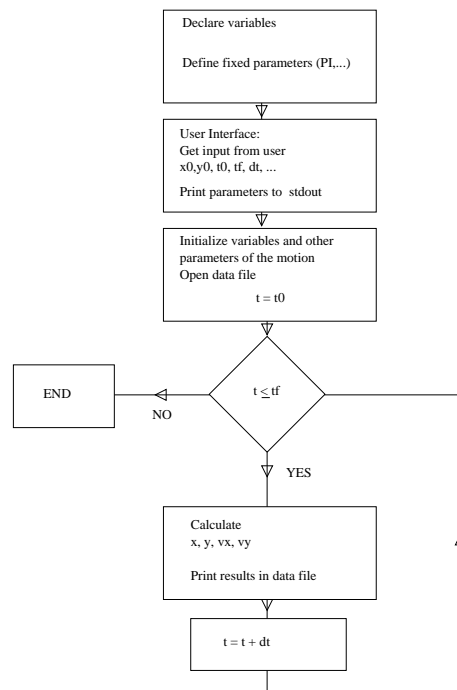


Σχήμα 2.1: Τροχιά κινητού υλικού σημείου στο επίπεδο. Φαίνονται τα διανύσματα θέσης  $\vec{r}$ , ταχύτητας  $\vec{v}$  και επιτάχυνσης  $\vec{a}$  και οι αντίστοιχες καρτεσιανές συντεταγμένες στο επιλεγμένο σύστημα αναφοράς σε ένα σημείο της τροχιάς.

Στην παράγραφο αυτή θα υποθέσουμε ότι μας δίνονται οι συναρτήσεις  $(x(t), y(t))$ . Από αυτές, παραγωγίζοντας σύμφωνα με τις παραπάνω σχέσεις, μπορούμε να πάρουμε την ταχύτητα και την επιτάχυνση. Σκοπός μας είναι να γράφουμε απλά προγράμματα τα οποία θα υπολογίζουν τις τιμές των συναρτήσεων αυτών σε ένα χρονικό διάστημα  $[t_0, t_f]$  όπου  $t_0$  είναι η αρχική χρονική στιγμή και  $t_f$  η τελική. Οι συνεχείς συναρτήσεις  $x(t), y(t), v_x(t), v_y(t)$  θα προσεγγίζονται από μια διακριτή ακολουθία τιμών των συναρτήσεων τις χρονικές στιγμές  $t_0, t_0 + \delta t, t_0 + 2\delta t, t_0 + 3\delta t, \dots$ , έτσι ώστε όλες οι τιμές  $t_0 + n\delta t \leq t_f$ <sup>1</sup>.

Αρχίζουμε σχεδιάζοντας το πρότυπο (template) ενός προγράμματος που θα κάνει τον παραπάνω υπολογισμό. Κάνοντας αυτό το σχεδιασμό προσεκτικά, το μόνο που θα μας απασχολεί κάθε φορά που θέλουμε

<sup>1</sup>Μπορεί η μεγαλύτερη από τις τιμές αυτές να είναι μικρότερη από  $t_f$  και η  $t_f$  να μη συμπεριλαμβάνεται στην ακολουθία.



Σχήμα 2.2: Λογικό διάγραμμα ενός τυπικού προγράμματος των εξισώσεων κίνησης.

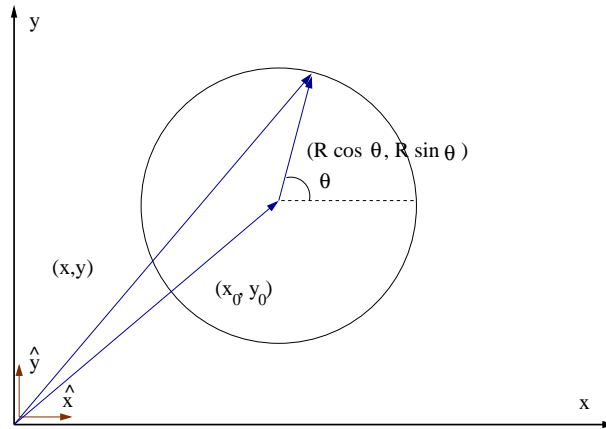
να μελετήσουμε την κίνηση ενός άλλου κινητού θα είναι η ειδική εφαρμογή των εξισώσεων κίνησης. Στο σχήμα 2.2 φαίνεται το λογικό διάγραμμα των βασικών λειτουργιών του προγράμματος. Αυτό μας βοηθάει να έχουμε μία εποπτική εικόνα, όταν προγραμματίζουμε τις λεπτομέρειες και μας βοηθάει να ελέγξουμε τη βασική λογική του αλγόριθμου που θα χρησιμοποιήσουμε. Στα ορθογώνια παραλληλόγραμμα σκιαγραφούμε τις σημαντικές εντολές των δομικών στοιχείων του προγράμματος, ενώ στους ρόμβους σημειώνουμε τις διακλαδώσεις (branching points) του προγράμματος που προκύπτουν από τις δυνατές τιμές μιας λογικής πρότασης. Με γραμμές ενώνουμε τη λογική σειρά με την οποία εκτελούνται οι διαδικασίες.

Το πρώτο κομμάτι του προγράμματος κάνει τις απαραίτητες δηλώσεις των τύπων των μεταβλητών που θα χρησιμοποιήσουμε. Αν υπάρχουν παράμετροι που παραμένουν σταθερές κατά τη διάρκεια της εκτέλεσης του προγράμματος (λ.χ.  $\pi = 3.1459 \dots$ ,  $g = 9.81$ , κλπ), τις ορίζουμε εδώ. Στη συνέχεια το πρόγραμμα αλληλεπιδρά με το χρήστη (user interface) και ζητάει τις μεταβλητές παραμέτρους που καθορίζει ο χρήστης:  $x_0$ ,  $y_0$ ,  $t_0$ ,  $t_f$ ,  $dt \dots$ . Το πρόγραμμα τυπώνει αυτές τις τιμές στο stdout για να μπορεί ο χρήστης να τις ελέγξει ως προς την ορθότητα και να

τις αποθηκεύσει για αναφορά στα δεδομένα του. Πριν τον κύριο υπολογισμό, ο χρήστης πρέπει να δώσει τις κατάλληλες αρχικές τιμές σε ορισμένες μεταβλητές, ειδικά στο χρόνο  $t = t_0$ .

Ο κύριος υπολογισμός γίνεται σε ένα βρόχο (loop) ο οποίος εκτελείται όσο ο χρόνος  $t \leq t_f$ . Υπολογίζονται οι τιμές της θέσης και της ταχύτητας  $x(t), y(t), v_x(t), v_y(t)$  και τυπώνονται μαζί με το χρόνο  $t$  σε ένα αρχείο. Εδώ θα κάνουμε τη σημαντική για μας σύμβαση στη μορφοποίηση της εξόδου. Αυτό είναι σημαντικό ώστε να έχουμε ενιαίο λογισμικό ανάλυσης των αποτελεσμάτων. Ορίζουμε σε κάθε γραμμή του αρχείου αυτού οι πρώτες πέντε στήλες να είναι οι τιμές  $t, x, y, v_x, v_y$ . Μπορούν να υπάρχουν και άλλες στήλες, αν χρειαστεί, στο ειδικό πρόβλημα που θα μελετάμε, αλλά οι πρώτες πέντε θα είναι πάντα αυτές.

Αφού γίνει αυτό, ειδικευόμαστε στο πρόβλημα που θα μελετήσουμε. Ας θεωρήσουμε αρχικά την περίπτωση ενός υλικού σημείου το οποίο εκτελεί ομαλή κυκλική κίνηση. Θεωρούμε το κέντρο του κύκλου  $(x_0, y_0)$ , την ακτίνα  $R$  και τη γωνιακή ταχύτητα  $\omega$  ότι είναι οι βασικές παράμετροι που προσδιορίζουν την κίνηση. Η θέση πάνω στον κύκλο μπορεί τότε να καθοριστεί από τη γωνία  $\theta$  όπως φαίνεται στο σχήμα 2.3. Θα ορίσουμε την αρχική χρονική στιγμή  $t_0$  να είναι  $\theta = 0$ .



Σχήμα 2.3: Η τροχιά του υλικού σημείου που εκτελεί την ομαλή κυκλική κίνηση που υπολογίζει το πρόγραμμα Circle.f90.

Έτσι οι εξισώσεις που δίνουν τη θέση του κινητού κάθε χρονική στιγμή είναι:

$$\begin{aligned} x(t) &= x_0 + R \cos(\omega(t - t_0)) \\ y(t) &= y_0 + R \sin(\omega(t - t_0)) . \end{aligned} \quad (2.3)$$

Παραγωγίζοντας ως προς  $t$  παίρνουμε την ταχύτητα

$$\begin{aligned} v_x(t) &= -\omega R \sin(\omega(t - t_0)) \\ v_y(t) &= \omega R \cos(\omega(t - t_0)) , \end{aligned} \quad (2.4)$$

και την επιτάχυνση

$$\begin{aligned} a_x(t) &= -\omega^2 R \cos(\omega(t - t_0)) = -\omega^2(x(t) - x_0) \\ a_y(t) &= -\omega^2 R \sin(\omega(t - t_0)) = -\omega^2(y(t) - y_0) . \end{aligned} \quad (2.5)$$

Από τις παραπάνω εξισώσεις παρατηρούμε τις γνωστές γεωμετρικές σχέσεις  $\vec{R} \cdot \vec{v} = 0$  ( $\vec{R} \equiv \vec{r} - \vec{r}_0$ ,  $\vec{v} \perp \vec{R}$ ,  $\vec{v}$  εφαπτόμενο στην τροχιά) και  $\vec{a} = -\omega^2 \vec{R}$  ( $\vec{R}$  και  $\vec{a}$  αντιπαράλληλα,  $\vec{a} \perp \vec{v}$ ).

Μπορούμε τώρα να σχεδιάσουμε τη δομή των δεδομένων για το πρόγραμμα που θα γράψουμε, η οποία στην περίπτωσή μας είναι πολύ απλή. Η σταθερή γωνιακή ταχύτητα  $\omega$  του υλικού σημείου αποθηκεύεται στην πραγματική (REAL) μεταβλητή omega. Το κέντρο του κύκλου  $(x_0, y_0)$ , η ακτίνα  $R$  και η γωνία  $\theta$  αντιστοιχούν στις REAL μεταβλητές x0, y0, R, theta. Οι χρονικές στιγμές που υπολογίζουμε τη θέση και ταχύτητα του κινητού καθορίζονται από τις παραμέτρους  $t_0, t_f, \delta t$  που αντιστοιχούν στο πρόγραμμα στις REAL μεταβλητές t0, tf, dt. Η τρέχουσα θέση του κινητού  $(x(t), y(t))$  υπολογίζεται και αποθηκεύεται στις REAL μεταβλητές x, y και η στιγμιαία του ταχύτητα  $(v_x(t), v_y(t))$  στις REAL μεταβλητές vx, vy. Οι δηλώσεις αυτές γίνονται στην αρχή του προγράμματος με τις εντολές:

```
real :: x0, y0, R, x, y, vx, vy, t, t0, tf, dt
real :: theta, omega
real, parameter :: PI=3.1415927
```

όπου ορίσαμε και την τιμή<sup>2</sup> του  $\pi = 3.1415927$  με τον προσδιορισμό parameter.

Το διαδραστικό με το χρήστη κομμάτι του προγράμματος (user interface) ζητάει από το χρήστη τις τιμές των παραμέτρων που του δίνουμε τη δυνατότητα να προσδιορίζει: omega, x0, y0, R, t0, tf, dt. Αρχικά το πρόγραμμα τυπώνει ένα μήνυμα προτροπής (prompt) στο χρήστη, προσδιορίζοντας τη μεταβλητή που ζητά να διαβάσει. Αυτό γίνεται με απλές print εντολές. Η ανάγνωση των τιμών των παραμέτρων γίνεται

<sup>2</sup>Θυμίζουμε στον αναγνώστη ότι οι μεταβλητές REAL μονής ακρίβειας (4 bytes) έχουν ακρίβεια περίπου 7 δεκαδικών ψηφίων.

από το stdin με εντολές read. Αφού διαβαστούν οι παράμετροι, το πρόγραμμα τυπώνει τις τιμές που διάβασε στο stdout<sup>3</sup>:

```
print *, '# Enter omega: '
read *, omega
print *, '# Enter center of circle (x0,y0) and radius R: '
read *, x0, y0, R
print *, '# Enter t0,tf,dt: '
read *, t0,tf,dt
print *, '# omega= ',omega
print *, '# x0= ',x0, ' y0= ',y0, ' R= ',R
print *, '# t0= ',t0, ' tf= ',tf, ' dt= ',dt
```

Στη συνέχεια, το πρόγραμμα θέτει την αρχική κατάσταση των δεδομένων ώστε να γίνει ο υπολογισμός. Αυτό, εκτός από το να θέσει την τιμή του αρχικού χρόνου  $t = t_0$ , περιλαμβάνει και βασικό έλεγχο για τη νομιμότητα των παραμέτρων που εισήγαγε ο χρήστης. Οι έλεγχοι είναι απαραίτητοι, γιατί, όταν γράφουμε ένα πρόγραμμα, κάνουμε ορισμένες υποθέσεις απαραίτητες για την ορθή λειτουργία του προγράμματος που μπορεί ο χρήστης από σφάλμα ή άγνοια να μην έχει σεβαστεί. Όταν, για παράδειγμα, γράφουμε την έκφραση  $2.0*PI/omega$ , υποθέτουμε ότι η τιμή του  $omega$  είναι μη μηδενική, ώστε να γίνει η διαίρεση χωρίς να προκύψει μοιραίο σφάλμα κατά την εκτέλεση του προγράμματος. Το πρόγραμμά μας για να λειτουργήσει όπως το σχεδιάσαμε θα απαιτήσουμε να έχουμε  $R > 0$  και  $\omega > 0$ . Αυτό θα το ελέγξουμε με μία εντολή if και, αν δεν πληρούνται οι υποθέσεις, σταματάμε τελείως το πρόγραμμα με την εντολή stop<sup>4</sup>. Το πρόγραμμα, επίσης, θα ανοίξει το αρχείο Circle.dat στο οποίο θα αποθηκεύσουμε τις υπολογισμένες τιμές της θέσης και ταχύτητας του κινητού.

```
if(R .le. 0.0) stop 'Illegal value of R'
if(omega .le. 0.0) stop 'Illegal value of omega'
print *, '# T= ',2.0*PI/omega
open(unit=11,file='Circle.dat')
```

Αν  $R \leq 0$  ή  $\omega \leq 0$ , τότε εκτελούνται οι αντίστοιχες εντολές stop οι

<sup>3</sup>Αυτό γίνεται για να αποφευχθούν τυπογραφικά λάθη και για να ενημερωθεί ο χρήστης για την τιμή που πραγματικά διάβασε το πρόγραμμα στη μνήμη του υπολογιστή. Επίσης, οδηγώντας το stdout σε ένα αρχείο στο σκληρό δίσκο, ο χρήστης μπορεί να αποθηκεύσει τα δεδομένα που εισήγαγε για μελλοντική αναφορά και για χρήση στην ανάλυση των αποτελεσμάτων του.

<sup>4</sup>Παρατηρήστε πως δεν ελέγξαμε όλες τις υποθέσεις που κάνουμε στο πρόγραμμα. Προσθέστε εσείς τις αναγκαίες σχετικές εντολές.



οποίες σταματούν την εκτέλεση του προγράμματος. Μετά την εντολή `stop` βάζουμε (προαιρετικά) ένα μήνυμα στο χρήστη, ώστε να ξέρει γιατί σταμάτησε το πρόγραμμα, το οποίο τυπώνεται στο `stdout` όταν εκτελείται η εντολή `stop`. Επίσης, υπολογίζουμε και τυπώνουμε την περίοδο της κυκλικής κίνησης  $T = 2\pi/\omega$ . Στην εντολή `open` επιλέξαμε το `unit 11` για να γράφουμε στο `Circle.dat`. Η επιλογή του αριθμού αυτού είναι ελεύθερη για τον προγραμματιστή που μπορεί με ασφάλεια να διαλέξει έναν οποιοδήποτε αριθμό από 10 έως 99<sup>5</sup>.

Τώρα είμαστε στη φάση που μπορεί να γίνει ο υπολογισμός. Αυτό θα γίνει με ένα βρόχο της μορφής

```
t = t0
do while(t .le. tf)
.....
t = t + dt
enddo
```

Η πρώτη εντολή (που τη δείξαμε και παραπάνω) θέτει την αρχική τιμή του χρόνου. Οι εντολές που περιλαμβάνονται ανάμεσα στο `do while` (συνθήκη) και `enddo` εκτελούνται όσο η συνθήκη είναι αληθής. Προσέξτε την εντολή `t = t + dt`, χωρίς την οποία ο βρόχος θα εκτελείται επ' αόριστον. Στο ειδικό πρόβλημα που μελετάμε, οι εντολές που μπαίνουν στη θέση των ..... υπολογίζουν τη θέση και την ταχύτητα και τις τυπώνουν στο αρχείο `Circle.dat`:

```
theta = omega * (t-t0)
x = x0+R*cos(theta)
y = y0+R*sin(theta)
vx = -omega*R*sin(theta)
vy = omega*R*cos(theta)
write(11,*)t,x,y,vx,vy
```

Στον υπολογισμό χρησιμοποιούμε τις συναρτήσεις `sin` και `cos` που είναι κτισμένες μέσα στη Fortran και οι οποίες δέχονται ως όρισμα μία γωνία μετρούμενη σε radians. Χρησιμοποιούμε την ενδιάμεση μεταβλητή `theta` για να υπολογίσουμε τη φάση  $\theta(t) = \omega(t-t_0)$ . Με την εντολή `write(11,*)` γράφουμε στο αρχείο `Circle.dat` έχοντας κάνει παραπάνω τη σύνδεση με το `unit 11` με την εντολή `open`.

Το πρόγραμμα το πληκτρολογούμε στο αρχείο `Circle.f90`. Η κατάληξη `“.f90”` υποδηλώνει στο μεταγλωττιστή ότι το αρχείο περιέχει

<sup>5</sup>Μερικοί αριθμοί μπορεί να χρησιμοποιούνται από το σύστημα για ειδικά αρχεία όπως λ.χ. το 5 για το `stdin`, το 6 για το `stdout` και το 0 για το `stderr`.

κώδικα στη γλώσσα Fortran. Για να το μεταγλωττίσουμε και να το τρέξουμε, δίνουμε τις εντολές:

```
> gfortran Circle.f90 -o c1
> ./c1
```

Με το διακόπτη `-o c1` δίνουμε την εντολή στο μεταγλωττιστή `gfortran` να ονομάσει το εκτελέσιμο αρχείο<sup>6</sup> `c1`. Η δεύτερη εντολή (`./c1`) φορτώνει τις μεταγλωττισμένες εντολές στη μνήμη του υπολογιστή για εκτέλεση. Στη συνέχεια, το πρόγραμμά μας ζητάει τα δεδομένα και εκτελεί τον υπολογισμό. Μια πλήρης τυπική συνεδρία εκτέλεσης του προγράμματος δείχνεται παρακάτω:

```
> gfortran Circle.f90 -o c1
> ./c1
# Enter omega:
1.0
# Enter center of circle (x0,y0) and radius R:
1.0 1.0 0.5
# Enter t0 ,tf ,dt:
0.0 20.0 0.01
# omega= 1.
# x0= 1. y0= 1. R= 0.5
# t0= 0. tf= 20. dt= 0.009999999978
# T= 6.28318548
```

Οι γραμμές που αρχίζουν από “#” τυπώνονται από το πρόγραμμα, ενώ οι γραμμές με αριθμούς χωρίς τη “#” είναι οι αριθμητικές τιμές των παραμέτρων που εισάγει ο χρήστης. Αφού πληκτρολογήσει τα δεδομένα, ο χρήστης χρησιμοποιεί το πλήκτρο Enter για να τα εισάγει στη μνήμη του υπολογιστή που τρέχει το πρόγραμμα. Εδώ  $\omega = 1.0$ ,  $x_0 = y_0 = 1.0$ ,  $R = 0.5$ ,  $t_0 = 0.0$ ,  $t_f = 20.0$  και  $\delta t = 0.01$ .

Το πρόγραμμα μπορείτε να το εκτελέσετε πολλές φορές για διαφορετικές τιμές των παραμέτρων με τη βοήθεια του editor. Σε ένα αρχείο με όνομα λ.χ. `Circle.in` τυπώστε τα δεδομένα που θέλετε να εισάγετε στο πρόγραμμα:

```
1.0          omega
1.0 1.0 0.5  (x0, y0) , R
0.0 20.0 0.01 t0 tf dt
```

<sup>6</sup>Θυμίζουμε ότι αν δε βάλουμε το διακόπτη αυτό, το εκτελέσιμο αρχείο θα ονομασθεί από το μεταγλωττιστή `a.out` από προεπιλογή.

Σε κάθε γραμμή του αρχείου βάζουμε τις τιμές των παραμέτρων που διαβάζει κάθε εντολή read με τη σωστή σειρά. Αφού διαβάσει όλες τις παραμέτρους της εντολής read, η Fortran αγνοεί την υπόλοιπη γραμμή. Η επόμενη εντολή read θα διαβάσει τα δεδομένα από μια νέα σειρά του αρχείου. Έτσι, δίπλα στις παραμέτρους βάζουμε σχόλια για το χρήστη που του θυμίζουν τη σειρά που διαβάζονται οι μεταβλητές. Το πρόγραμμα τώρα μπορούμε να το τρέξουμε με την εντολή:

```
> ./cl < Circle.in > Circle.out
```

Η εντολή ./cl τρέχει τις εντολές που περιέχονται στο εκτελέσιμο αρχείο cl. Το < Circle.in οδηγεί τα περιεχόμενα του αρχείου Circle.in στο standard input (stdin) του προγράμματος, εξαναγκάζοντας το cl να διαβάσει τα δεδομένα από το αρχείο αυτό. Το > Circle.out οδηγεί το stdout του προγράμματος cl στο αρχείο Circle.out του οποίου τα περιεχόμενα μπορούμε να τα δούμε, αφού τρέξουμε το πρόγραμμα, με την εντολή cat:

```
> cat Circle.out
# Enter omega:
# Enter center of circle (x0,y0) and radius R:
# Enter t0,tf,dt:
# omega= 1.
# x0= 1. y0= 1. R= 0.5
# t0= 0. tf= 20. dt= 0.009999999978
# T= 6.28318548
```

Κλείνουμε την παράγραφο αυτή, παραθέτοντας για διευκόλυνση του αναγνώστη ολόκληρο το πρόγραμμα που πληκτρολογήσαμε στο αρχείο Circle.f90:

```
!=====
!File Circle.f90
!Constant angular velocity circular motion
!Set (x0,y0) center of circle , its radius R and omega.
!At t=t0, the particle is at theta=0
!
program Circle
  implicit none
!
!Declaration of variables
  real :: x0,y0,R,x,y,vx,vy,t,t0,tf,dt
  real :: theta,omega
  real, parameter :: PI=3.1415927
```

```

!
!Ask user for input:
print *, '# Enter omega: '
read *, omega
print *, '# Enter center of circle (x0,y0) and radius R: '
read *, x0, y0, R
print *, '# Enter t0,tf,dt: '
read *, t0, tf, dt
print *, '# omega= ', omega
print *, '# x0= ', x0, ' y0= ', y0, ' R= ', R
print *, '# t0= ', t0, ' tf= ', tf, ' dt= ', dt
!
!Initialize
if(R .le. 0.0) stop 'Illegal value of R'
if(omega .le. 0.0) stop 'Illegal value of omega'
print *, '# T= ', 2.0*PI/omega
open(unit=11, file='Circle.dat')
!
!Compute:
t = t0
do while(t .le. tf)
  theta = omega * (t-t0)
  x = x0+R*cos(theta)
  y = y0+R*sin(theta)
  vx = -omega*R*sin(theta)
  vy = omega*R*cos(theta)
  write(11,*)t,x,y,vx,vy
  t = t + dt
enddo
close(11)
end program Circle

```

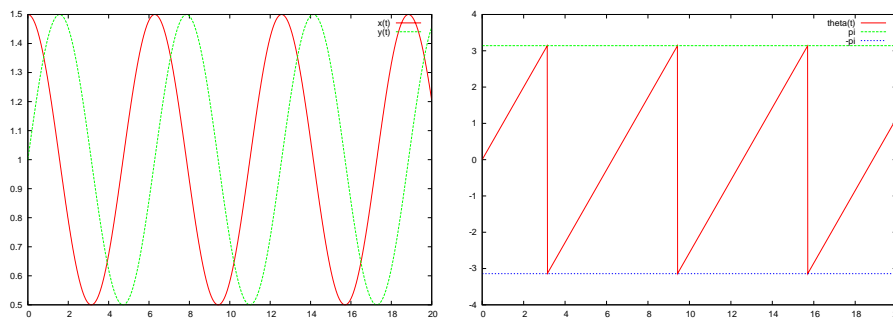
### 2.1.1 Απεικόνιση των Δεδομένων

Οι γραφικές παραστάσεις των αποτελεσμάτων που πήραμε στην προηγούμενη παράγραφο μελετώνται με τη βοήθεια του προγράμματος gnuplot. Ύπενθυμίζουμε ότι τα δεδομένα, το πρόγραμμά μας τα αποθηκεύει στο αρχείο Circle.dat σε πέντε στήλες: Η 1η είναι ο χρόνος  $t$ , η 2η και 3η οι συντεταγμένες  $x$ ,  $y$  και η 4η και 5η οι συντεταγμένες της ταχύτητας  $v_x$ ,  $v_y$ . Τα διαγράμματα  $x(t)$  και  $y(t)$  παράγονται από τις εντολές (που δίνονται μέσα από το gnuplot) και μπορείτε να τα δείτε στο (αριστερό) σχήμα 2.4:

```

gnuplot> plot "Circle.dat" using 1:2 with lines title "x(t)"
gnuplot> replot "Circle.dat" using 1:3 with lines title "y(t)"

```



Σχήμα 2.4: Τα διαγράμματα  $(x(t), y(t))$  (αριστερά) και  $\theta(t)$  (δεξιά) των δεδομένων που παράγονται από το πρόγραμμα Circle.f90 για  $\omega = 1.0$ ,  $x_0 = y_0 = 1.0$ ,  $R = 0.5$ ,  $t_0 = 0.0$ ,  $t_f = 20.0$  και  $\delta t = 0.01$ .

Η δεύτερη εντολή (replot) βάζει τη δεύτερη γραφική παράσταση μαζί με την πρώτη.

Ας δούμε τώρα πώς μπορούμε να φτιάξουμε τη γραφική παράσταση της γωνίας  $\theta(t)$ . Αυτό μπορούμε να το κάνουμε μέσα από το gnuplot, χωρίς να γράψουμε κάποιο καινούργιο πρόγραμμα, χρησιμοποιώντας τα δεδομένα μέσα από το αρχείο Circle.dat. Παρατηρούμε ότι  $\theta(t) = \tan^{-1}((y - y_0)/(x - x_0))$ . Η συνάρτηση atan2 (που υπάρχει και στη Fortran) είναι διαθέσιμη στο gnuplot<sup>7</sup>. Για να βρούμε πώς δουλεύει, χρησιμοποιούμε τη βοήθεια στο gnuplot:

```
gnuplot> help atan2
The 'atan2(y,x)' function returns the arc tangent (inverse
tangent) of the ratio of the real parts of its arguments.
'atan2' returns its argument in radians or degrees, as
selected by 'set angles', in the correct quadrant.
```

Άρα αρκεί να καλέσουμε τη συνάρτηση αυτή με εντολή της μορφής atan2(y-y0,x-x0). Στην περίπτωσή μας  $x_0=y_0=1$ , ενώ τα  $x, y$  είναι αντίστοιχα στη 2η και 3η στήλη κάθε γραμμής του αρχείου Circle.dat. Θα φτιάξουμε μία κατάλληλη έκφραση στην εντολή using στο gnuplot, όπως και στη σελίδα 65, όπου \$2 είναι η τιμή της 2ης και \$3 είναι η τιμή της 3ης στήλης:

```
gnuplot> x0 = 1 ; y0 = 1
gnuplot> plot "Circle.dat" using 1:(atan2($3-y0,$2-x0)) \
with lines title "theta(t)",pi,-pi
```

<sup>7</sup>Όπως και όλες οι μαθηματικές συναρτήσεις που υπάρχουν στη μαθηματική βιβλιοθήκη της γλώσσας C. Δώστε την εντολή help functions για να δείτε σχετικά.

Η δεύτερη εντολή δίνεται σε μία γραμμή την οποία σπάσαμε με το χαρακτήρα `\` ώστε να χωράει στο κείμενο<sup>8</sup>. Προσέξτε πώς ορίσαμε τις τιμές των μεταβλητών `x0,y0` μέσα στο `gnuplot` και τις χρησιμοποιήσαμε στην έκφραση `atan2($3-x0,$2-y0)` αντί (ισοδύναμα) να γράψαμε `atan2($3-1,$2-1)`. Επίσης, μαζί με τη γραφική παράσταση της  $\theta(t)$  κάνουμε και τις γραφικές παραστάσεις των σταθερών συναρτήσεων  $f_1(t) = \pi$ ,  $f_2(t) = -\pi$ , ώστε να ελέγξουμε τα όρια των τιμών της  $\theta(t)$ . Η μεταβλητή<sup>9</sup> `pi` στο `gnuplot` έχει ορισμένη εσωτερικά την τιμή  $\pi$ . Το αποτέλεσμα μπορείτε να το δείτε στο (δεξί) σχήμα 2.4.

Οι συνιστώσες των ταχυτήτων  $(v_x(t), v_y(t))$  ως συνάρτηση του χρόνου και η τροχιά του υλικού σημείου  $\vec{r}(t)$  μπορούν να απεικονιστούν με τις εντολές:

```
gnuplot> plot "Circle.dat" using 1:4 title "v_x(t)" with lines
gnuplot> replot "Circle.dat" using 1:5 title "v_y(t)" with lines
gnuplot> plot "Circle.dat" using 2:3 title "x-y" with lines
```

Η τελευταία εντολή τοποθετεί τα σημεία  $x, y$  στο επίπεδο, αφού αυτά βρίσκονται στις στήλες 2 και 3 αντίστοιχα.

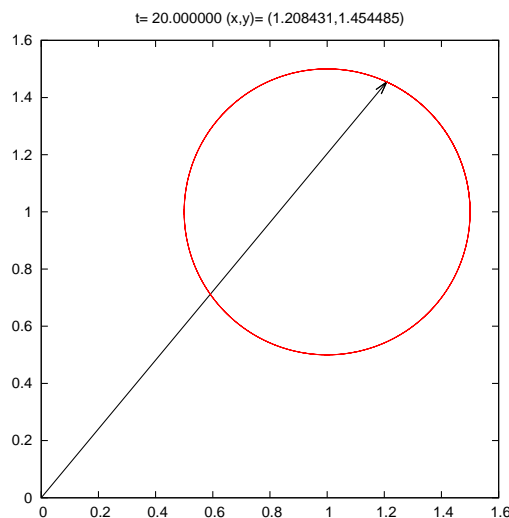
Κλείνουμε αυτήν την παράγραφο δείχνοντας πώς μπορούμε να δούμε το υλικό σημείο να κινείται κάνοντας στοιχειώδες animation με το `gnuplot`. Για το σκοπό αυτό, έχουμε ένα αρχείο `animate2D.gnu` στο συνοδευτικό λογισμικό το οποίο θα πρέπει να αντιγράψετε μέσα στον κατάλογο που έχετε το αρχείο των δεδομένων σας `Circle.dat` και από όπου θα δώσετε και την εντολή `gnuplot`. Δεν είναι σκοπός εδώ να σας εξηγήσουμε πώς δουλεύει, αλλά πώς να το χρησιμοποιείτε<sup>10</sup>. Στο σχήμα 2.5 φαίνεται το τελικό αποτέλεσμα. Οι εντολές είναι απλές. Αρκεί να ορίσουμε από ποιο αρχείο να διαβάσουμε τα δεδομένα<sup>11</sup>, τον αρχικό χρόνο απεικόνισης `t0` της τροχιάς, τον τελικό `tf` καθώς και το βήμα `dt`. Οι χρόνοι αυτοί δεν είναι ανάγκη να είναι οι ίδιοι με αυτούς που βάλαμε στα δεδομένα, όταν τρέχαμε το πρόγραμμα που βρίσκεται στο αρχείο `Circle.f90`. Οι εντολές που δίνουμε είναι οι

<sup>8</sup> Αυτό επιτρέπεται να το κάνετε και μέσα στο πρόγραμμα `gnuplot` αν το θέλετε.

<sup>9</sup> Δώστε την εντολή `show variables` για να δείτε τις ορισμένες μεταβλητές στο `gnuplot`.

<sup>10</sup> Φυσικά μπορείτε να δείτε τα περιεχόμενά του και να μαντέψετε πώς δουλεύει, είναι αρκετά απλό.

<sup>11</sup> Μπορεί να είναι οποιοδήποτε αρχείο του οποίου η 1η, 2η και 3η στήλη έχει το χρόνο  $t$  και  $x$  και  $y$  συνιστώσες της θέσης του υλικού σημείο στο επίπεδο αντίστοιχα.



Σχήμα 2.5: Η τροχιά του υλικού σημείο όπως απεικονίζεται κάθε χρονική στιγμή από το πρόγραμμα `animate2D.gnu` του συνοδευτικού λογισμικού που τρέχει μέσα από το `gnuplot`. Φαίνεται το διάνυσμα θέσης και στον τίτλο του διαγράμματος η χρονική στιγμή  $t$  και αντίστοιχη θέση  $(x,y)$ . Τα δεδομένα είναι από το πρόγραμμα `Circle.f90` που περιγράφεται στο κείμενο.

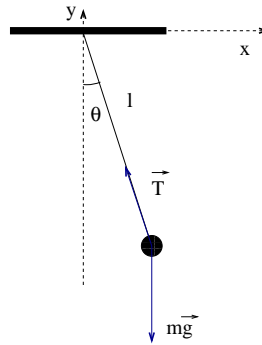
```
gnuplot> file = "Circle.dat"
gnuplot> set xrange [0:1.6]; set yrange [0:1.6]
gnuplot> t0 = 0; tf = 20 ; dt = 0.1
gnuplot> load "animate2D.gnu"
```

Η πρώτη εντολή καθορίζει το αρχείο από όπου το `animate2D.gnu` θα διαβάσει τα δεδομένα. Η δεύτερη ορίζει τα όρια στους άξονες  $x$  και  $y$ . Η τρίτη εντολή καθορίζει τις παραμέτρους του χρόνου που θα χρησιμοποιηθούν στο animation (αρχικός, τελικός και βήμα). Και η τέταρτη είναι η εντολή που “τρέχει” το animation. Αν θέλετε να ξανατρέξει το animation, αρκεί να δώσετε τις δύο τελευταίες εντολές όσες φορές θέλετε. Λ.χ. αν θέλετε να τρέξετε το animation στα ίδια δεδομένα με τη “μισή ταχύτητα”, αρκεί να ξαναορίσετε το `dt=0.05`, να επαναφέρετε το χρόνο `t0=0` και να δώσετε μόνο τις εντολές

```
gnuplot> t0 = 0; dt = 0.05
gnuplot> load "animate2D.gnu"
```

### 2.1.2 Άλλα Παραδείγματα

Ας εφαρμόσουμε όσα κάναμε για την απλή κυκλική κίνηση και σε διαφορετικά παραδείγματα κίνησης στο επίπεδο. Το πρώτο πρόβλημα που θα παρουσιάσουμε είναι αυτό του απλού εκκρεμούς που εκτελεί μικρές ταλαντώσεις γύρω από την κατακόρυφο και που φαίνεται στο σχήμα 2.6. Η κίνηση περιγράφεται από τη χρονική εξάρτηση του μοναδικού



Σχήμα 2.6: Το απλό εκκρεμές του οποίου η κίνηση για  $\theta \ll 1$  περιγράφεται από το πρόγραμμα SimplePendulum.f90.

βαθμού ελευθερίας του συστήματος, της γωνίας  $\theta(t)$ . Η κίνηση είναι περιοδική με γωνιακή συχνότητα  $\omega = \sqrt{g/l}$  και περίοδο  $T = 2\pi/\omega$ . Η γωνιακή ταχύτητα υπολογίζεται από τη σχέση  $\dot{\theta} \equiv d\theta/dt$  και παίρνουμε:

$$\begin{aligned}\theta(t) &= \theta_0 \cos(\omega(t - t_0)) \\ \dot{\theta}(t) &= -\omega\theta_0 \sin(\omega(t - t_0))\end{aligned}\quad (2.6)$$

Με τις παραπάνω σχέσεις έχουμε επιλέξει τις αρχικές συνθήκες  $\theta(t_0) = \theta_0$ ,  $\dot{\theta}(t_0) = 0$ . Για να γράψουμε τις εξισώσεις της τροχιάς στο καρτεσιανό σύστημα συντεταγμένων του σχήματος 2.6 χρησιμοποιούμε τις σχέσεις

$$\begin{aligned}x(t) &= l \sin(\theta(t)) \\ y(t) &= -l \cos(\theta(t)) \\ v_x(t) &= \frac{dx(t)}{dt} = l\dot{\theta}(t) \cos(\theta(t)) \\ v_y(t) &= \frac{dy(t)}{dt} = l\dot{\theta}(t) \sin(\theta(t)) .\end{aligned}\quad (2.7)$$

Αυτές είναι οι αντίστοιχες εξισώσεις των (2.3) και (2.4) για την περίπτωση της ομαλής κυκλικής κίνησης. Έτσι, ο σχεδιασμός του προγράμματος γίνεται με παρόμοιο τρόπο. Η τελική του μορφή, την οποία



βρίσκουμε στο αρχείο SimplePendulum.f90 στο συνοδευτικό λογισμικό είναι:

```
!=====
! File SimplePendulum.f90
! Set pendulum original position at theta0 with no initial speed
!
program SimplePendulum
  implicit none
!
! Declaration of variables
  real :: l,x,y,vx,vy,t,t0,tf,dt
  real :: theta,theta0,dtheta_dt,omega
  real, parameter :: PI=3.1415927,g=9.81
!
! Ask user for input:
  print *, '# Enter l: '
  read *, l
  print *, '# Enter theta0: '
  read *, theta0
  print *, '# Enter t0,tf,dt: '
  read *, t0,tf,dt
  print *, '# l= ',l, ' theta0= ',theta0
  print *, '# t0= ',t0, ' tf= ',tf, ' dt= ',dt
!
! Initialize
  omega = sqrt(g/l)
  print *, '# omega= ',omega, ' T= ',2.0*PI/omega
  open(unit=11,file='SimplePendulum.dat')
!
! Compute:
  t = t0
  do while(t .le. tf)
    theta = theta0*cos(omega*(t-t0))
    dtheta_dt = -omega*theta0*sin(omega*(t-t0))
    x = l*sin(theta)
    y = -l*cos(theta)
    vx = l*dtheta_dt*cos(theta)
    vy = l*dtheta_dt*sin(theta)
    write(11,100)t,x,y,vx,vy,theta,dtheta_dt
    t = t + dt
  enddo
  close(11)
100 FORMAT(7G15.7)
end program SimplePendulum
```

Τα μόνα σημεία που θα επισημάνουμε στον αναγνώστη είναι ότι την επιτάχυνση της βαρύτητας  $g$  τη θέτουμε σταθερή παράμετρο στο πρό-

γραμμά, ο χρήστης μπορεί να καθορίσει το μήκος του εκκρεμούς  $l$  και ότι το αρχείο των δεδομένων περιέχει εκτός από τις στήλες 1-5 και άλλες 2 στις οποίες μπορούμε να δούμε τη γωνία και στιγμιαία γωνιακή ταχύτητα του εκκρεμούς. Η εντολή `write(11,100)` γράφει στο unit 11 με τη μορφή (format) που καθορίζει η εντολή `FORMAT` με label 100. Αυτό γίνεται ώστε τα δεδομένα να τυπώνονται σε μία γραμμή (δείτε τη συζήτηση στη σελίδα 52).

Μια απλή συνεδρία μελέτης του προβλήματος συνοψίζεται παρακάτω<sup>12</sup>:

```
> gfortran SimplePendulum.f90 -o sp
> ./sp
# Enter l:
1.0
# Enter theta0:
0.314
# Enter t0 , tf , dt:
0 20 0.01
# l=      1.      theta0=  0.31400001
# t0=      0.      tf=      20. dt=  0.00999999978
# omega=  3.132092 T=      2.0060668
> gnuplot
gnuplot> plot "SimplePendulum.dat" u 1:2 w lines t "x(t)"
gnuplot> plot "SimplePendulum.dat" u 1:3 w lines t "y(t)"
gnuplot> plot "SimplePendulum.dat" u 1:4 w lines t "v_x(t)"
gnuplot> replot "SimplePendulum.dat" u 1:5 w lines t "v_y(t)"
gnuplot> plot "SimplePendulum.dat" u 1:6 w lines t "th(t)"
gnuplot> replot "SimplePendulum.dat" u 1:7 w lines t "th'(t)"
gnuplot> plot [-0.6:0.6][-1.1:0.1] "SimplePendulum.dat" \
u 2:3 w lines t "x-y"

gnuplot> file = "SimplePendulum.dat"
gnuplot> t0=0;tf=20.0;dt=0.1
gnuplot> set xrange [-0.6:0.6]; set yrange [-1.1:0.1]
gnuplot> load "animate2D.gnu"
```

Το επόμενο πρόβλημα θα είναι η μελέτη της βολής υλικού σημείου κοντά στην επιφάνεια της γης<sup>13</sup> αγνοώντας όλες τις δυνάμεις τριβής του αέρα. Όπως γνωρίζουμε, η τροχιά του σωματιδίου και η ταχύτητά του

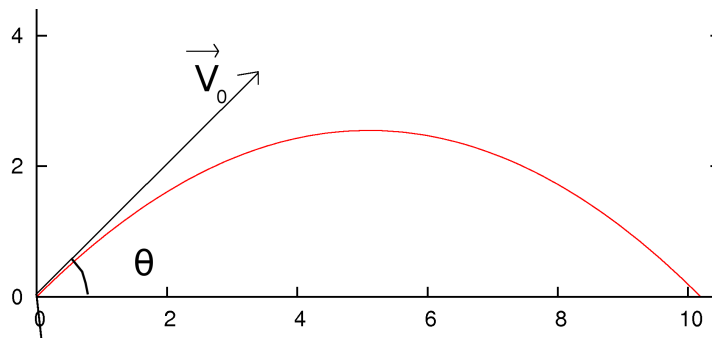
<sup>12</sup>Παρατηρήστε πως συντομεύσαμε την εντολή “using 1:2 with lines title” με την “u 1:2 w lines t”. Αυτό μπορούμε να το κάνουμε με κάθε εντολή στο gnuplot, αρκεί να υπάρχει μοναδική εντολή που να προσδιορίζεται με τη συντόμευση.

<sup>13</sup>Δηλ.  $\vec{g}$  = σταθ. και η επίδραση της δύναμης Coriolis αμελητέα.

δίνονται από τις παραμετρικές εξισώσεις, με παράμετρο το χρόνο

$$\begin{aligned}x(t) &= v_{0x}t \\y(t) &= v_{0y}t - \frac{1}{2}gt^2 \\v_x(t) &= v_{0x} \\v_y(t) &= v_{0y} - gt,\end{aligned}\tag{2.8}$$

όπου έχουμε υποθέσει τις αρχικές συνθήκες  $x(0) = y(0) = 0$ ,  $v_x(0) = v_{0x} = v_0 \cos \theta$  και  $v_y(0) = v_{0y} = v_0 \sin \theta$  σύμφωνα με το σχήμα 2.7.



Σχήμα 2.7: Βολή υλικού σημείου κοντά στην επιφάνεια της γης με  $x(0) = y(0) = 0$ ,  $v_x(0) = v_{0x} = v_0 \cos \theta$  και  $v_y(0) = v_{0y} = v_0 \sin \theta$ .

Η δομή του προγράμματος είναι παρόμοια με τα προηγούμενα. Ο χρήστης εισάγει το μέτρο της αρχικής ταχύτητας και τη γωνία  $\theta$  σε μοίρες που σχηματίζει με την οριζόντια διεύθυνση. Εδώ παίρνουμε το χρόνο  $t_0 = 0$ . Το πρόγραμμα υπολογίζει τις  $v_{0x}$  και  $v_{0y}$  και τις τυπώνει στο χρήστη στο stdout. Τα δεδομένα σώζονται στο αρχείο Projectile.dat. Το πρόγραμμα δίνεται ολόκληρο παρακάτω και βρίσκεται στο αρχείο Projectile.f90 του συνοδευτικού λογισμικού.

```
!=====
! File Projectile.f90
!Shooting a projectile near the earth surface.
!No air resistance.
!Starts at (0,0), set (v0,theta).
!-----
program Projectile
  implicit none
!-----
!Declaration of variables
  real :: x0,y0,R,x,y,vx,vy,t,tf,dt
```

```

real :: theta,v0x,v0y,v0
real, parameter :: PI=3.1415927,g=9.81
!
!Ask user for input:
print *,'# Enter v0,theta (in degrees):'
read *,v0,theta
print *,'# Enter tf,dt:'
read *,tf,dt
print *,'# v0= ',v0,' theta= ',theta,' o (degrees)'
print *,'# t0= ',0.0,' tf= ',tf,' dt= ',dt
!
!Initialize
if( v0 .le. 0.0) stop 'Illegal value of v0<=0'
if( theta .le. 0.0 .or. theta .ge. 90.0) &
    stop 'Illegal value of theta'
theta = (PI/180.0)*theta !convert to radians
v0x = v0*cos(theta)
v0y = v0*sin(theta)
print *,'# v0x = ',v0x,' v0y= ',v0y
open(unit=11,file='Projectile.dat')
!
!Compute:
t = 0.0
do while(t .le. tf)
    x = v0x * t
    y = v0y * t - 0.5*g*t*t
    vx = v0x
    vy = v0y - g*t
    write(11,*)t,x,y,vx,vy
    t = t + dt
enddo
close(11)
end program Projectile

```

Για διευκόλυνση του αναγνώστη, δίνουμε πάλι τις εντολές μιας τυπικής συνεδρίας μελέτης του προβλήματος:

```

> gfortran Projectile.f90 -o pj
> ./pj
# Enter v0,theta (in degrees):
10 45
# Enter tf,dt:
1.4416 0.001
# v0= 10.0000000 theta= 45.000000 o (degrees)
# t0= 0.0000000 tf= 1.4416000 dt= 1.00000005E-03
# v0x = 7.0710678 v0y= 7.0710678
> gnuplot
gnuplot> plot "Projectile.dat" using 1:2 w lines t "x(t)"

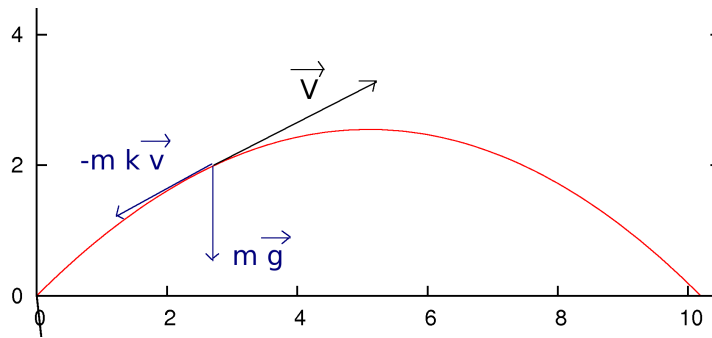
```

```

gnuplot> replot "Projectile.dat" using 1:3 w lines t "y(t)"
gnuplot> plot "Projectile.dat" using 1:4 w lines t "v_x(t)"
gnuplot> replot "Projectile.dat" using 1:5 w lines t "v_y(t)"
gnuplot> plot "Projectile.dat" using 2:3 w lines t "x-y"
gnuplot> file = "Projectile.dat"
gnuplot> set xrange [0:10.3]; set yrange [0:10.3]
gnuplot> t0=0;tf=1.4416;dt=0.05
gnuplot> load "animate2D.gnu"

```

Στη συνέχεια, ας δούμε το σύστημα όταν στο υλικό σημείο δρα ομαλή αντίσταση από ένα ρευστό της μορφής  $\vec{F} = -mk\vec{v}$ , δηλ. μια δύναμη που είναι αντίθετη με την ταχύτητα σε κάθε χρονική στιγμή. Οι λύσεις των εξισώσεων κίνησης



Σχήμα 2.8: Οι δυνάμεις που ασκούνται στο σωματίδιο του σχήματος 2.7, όταν υποθέσουμε και την ύπαρξη αντίστασης του αέρα ανάλογης του μέτρου της ταχύτητας  $\vec{F} = -mk\vec{v}$ .

$$\begin{aligned}
 a_x &= \frac{dv_x}{dt} = -kv_x \\
 a_y &= \frac{dv_y}{dt} = -kv_y - g
 \end{aligned}
 \quad (2.9)$$

με αρχικές συνθήκες  $x(0) = y(0) = 0$ ,  $v_x(0) = v_{0x} = v_0 \cos \theta$  και  $v_y(0) = v_{0y} = v_0 \sin \theta$  έχουν λύση<sup>14</sup>

$$\begin{aligned}
 v_x(t) &= v_{0x} e^{-kt} \\
 v_y(t) &= \left( v_{0y} + \frac{g}{k} \right) e^{-kt} - \frac{g}{k} \\
 x(t) &= \frac{v_{0x}}{k} (1 - e^{-kt}) \\
 y(t) &= \frac{1}{k} \left( v_{0y} + \frac{g}{k} \right) (1 - e^{-kt}) - \frac{g}{k} t
 \end{aligned}
 \quad (2.10)$$

<sup>14</sup>Ο αναγνώστης καλείται να αποδείξει τις εξισώσεις (2.10) σαν άσκηση.

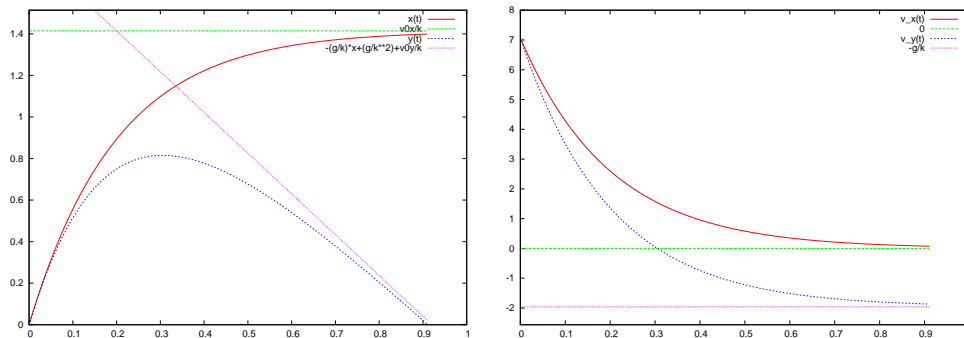
Οι εξισώσεις αυτές προγραμματίζονται με παρόμοιο τρόπο όπως και στην περίπτωση της απουσίας της αντίστασης του αέρα. Η μόνη διαφορά είναι ότι ο χρήστης εισάγει τη σταθερά  $k$  στα δεδομένα και φυσικά η μορφή των εξισώσεων. Το πρόγραμμα βρίσκεται στο αρχείο ProjectileAirResistance.f90 στο συνοδευτικό υλικό και παρατίθεται αυτούσιο παρακάτω:

```
!=====
!File ProjectileAirResistance.f90
!Shooting a projectile near the earth surface
!with air resistance.
!Starts at (0,0), set k, (v0,theta).
!
program ProjectileAirResistance
  implicit none
!
!Declaration of variables
  real :: x0,y0,R,x,y,vx,vy,t,tf,dt,k
  real :: theta,v0x,v0y,v0
  real, parameter :: PI=3.1415927,g=9.81
!
!Ask user for input:
  print *, '# Enter k, v0,theta (in degrees): '
  read *,k, v0,theta
  print *, '# Enter tf,dt: '
  read *, tf,dt
  print *, '# k = ',k
  print *, '# v0= ',v0, ' theta= ',theta, 'o (degrees)'
  print *, '# t0= ',0.0, ' tf= ',tf, ' dt= ',dt
!
!Initialize
  if( v0 .le. 0.0) stop 'Illegal value of v0<=0'
  if( k .le. 0.0) stop 'Illegal value of k <=0'
  if( theta .le. 0.0 .or. theta .ge. 90.0) &
    stop 'Illegal value of theta'
  theta = (PI/180.0)*theta !convert to radians
  v0x = v0*cos(theta)
  v0y = v0*sin(theta)
  print *, '# v0x = ',v0x, ' v0y= ',v0y
  open(unit=11,file='ProjectileAirResistance.dat')
!
!Compute:
  t = 0.0
  do while(t .le. tf)
    x = (v0x/k)*(1.0-exp(-k*t))
    y = (1.0/k)*(v0y+(g/k))*(1.0-exp(-k*t))-(g/k)*t
    vx = v0x*exp(-k*t)
```

```

vy = (v0y+(g/k))*exp(-k*t)-(g/k)
write(11,*)t,x,y,vx,vy
t = t + dt
enddo
close(11)
end program ProjectileAirResistance

```



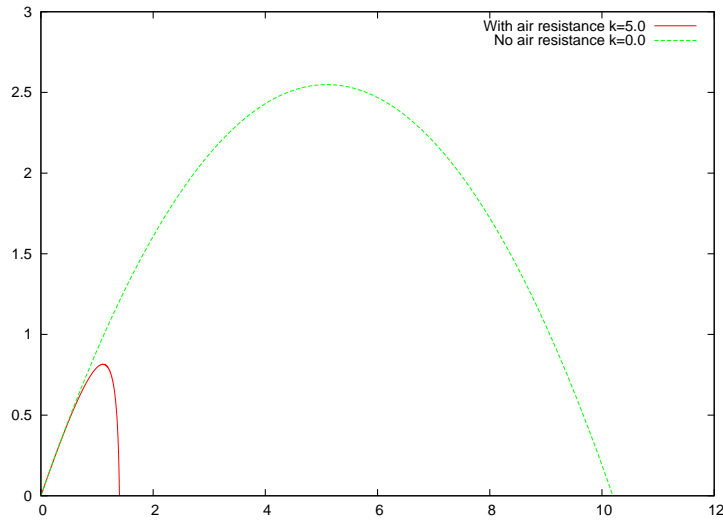
Σχήμα 2.9: Τα διαγράμματα  $x(t), y(t)$  (αριστερά) και  $v_x(t), v_y(t)$  (δεξιά) των δεδομένων που παράγονται από το πρόγραμμα ProjectileAirResistance.f90 για  $k = 5.0$ ,  $v_0 = 10.0$ ,  $\theta = \pi/4$ ,  $t_f = 0.91$  και  $\delta t = 0.001$ . Φαίνονται και οι ασύμπτωτες των συναρτήσεων καθώς  $t \rightarrow \infty$ .

Δίνουμε πάλι τις εντολές μιας τυπικής συνεδρίας μελέτης του προβλήματος:

```

> gfortran ProjectileAirResistance.f90 -o pja
> ./pja
# Enter k, v0, theta (in degrees):
5.0 10.0 45
# Enter tf, dt:
0.91 0.001
# k = 5.
# v0= 10. theta= 45.0 (degrees)
# t0= 0. tf= 0.910000026 dt= 0.00100000005
# v0x = 7.07106781 v0y= 7.07106781
> gnuplot
gnuplot> v0x = 10*cos(pi/4) ; v0y = 10*sin(pi/4)
gnuplot> g = 9.81 ; k = 5
gnuplot> plot [:][:v0x/k+0.1] "ProjectileAirResistance.dat" \
using 1:2 with lines title "x(t)", v0x/k
gnuplot> replot "ProjectileAirResistance.dat" \
using 1:3 with lines title "y(t)", \
-(g/k)*x+(g/k**2)+v0y/k
gnuplot> plot [:][: -g/k-0.6:] "ProjectileAirResistance.dat" \

```



Σχήμα 2.10: Οι τροχιές των βλημάτων που βάλλονται με  $v_0 = 10.0$ ,  $\theta = \pi/4$  όταν δεν υπάρχει αντίσταση του αέρα και όταν υπάρχει με  $k = 5.0$ .

```

gnuplot> using 1:4 with lines title "v_x(t)",0
gnuplot> replot "ProjectileAirResistance.dat" \
gnuplot> using 1:5 with lines title "v_y(t)",-g/k
gnuplot> plot "ProjectileAirResistance.dat" \
gnuplot> using 2:3 with lines title "With air resistance k=5.0"
gnuplot> replot "Projectile.dat" \
gnuplot> using 2:3 with lines title "No air resistance k=0.0"
gnuplot> file = "ProjectileAirResistance.dat"
gnuplot> set xrange [0:1.4]; set yrange [0:1.4]
gnuplot> t0=0;tf=0.91;dt=0.01
gnuplot> load "animate2D.gnu"

```

Όπως και παραπάνω διπλώσαμε τις εντολές που δε χωρούσαν σε δύο γραμμές. Ορίσαμε τις μεταβλητές `gnuplot` `v0x`, `v0y`, `g` και `k` να έχουν τις τιμές που χρησιμοποιήσαμε όταν τρέξαμε το πρόγραμμα. Μπορούμε έτσι και κατασκευάζουμε τις ασύμπτωτες των συναρτήσεων<sup>15</sup>. Τα αποτελέσματα φαίνονται στα σχήματα 2.9 και 2.10.

Το τελευταίο παράδειγμα που θα μελετήσουμε στην παράγραφο αυτή είναι αυτό του ανισότροπου αρμονικού ταλαντωτή. Αυτό είναι ένα υλικό σημείο υπό την επίδραση της δύναμης

$$F_x = -m\omega_1^2 x \quad F_y = -m\omega_2^2 y \quad (2.11)$$

<sup>15</sup>Θυμίζουμε στον αναγνώστη ότι η ανεξάρτητη μεταβλητή στο `gnuplot` είναι η  $x$  από προεπιλογή.



όπου οι “σταθερές των ελατηρίων”  $k_1 = m\omega_1^2$  και  $k_2 = m\omega_2^2$  είναι διαφορετικές στις δύο ορθογώνιες κατευθύνσεις που ορίζονται από τους καρτεσιανούς άξονες  $x$  και  $y$ . Οι λύσεις των εξισώσεων αυτών για  $x(0) = A$ ,  $y(0) = 0$ ,  $v_x(0) = 0$  και  $v_y(0) = \omega_2 A$  είναι

$$\begin{aligned} x(t) &= A \cos(\omega_1 t) & y(t) &= A \sin(\omega_2 t) \\ v_x(t) &= -\omega_1 A \sin(\omega_1 t) & v_y(t) &= \omega_2 A \cos(\omega_2 t). \end{aligned} \quad (2.12)$$

Για ορισμένες σχέσεις μεταξύ των κυκλικών συχνοτήτων  $\omega_1$  και  $\omega_2$  οι καμπύλες των τροχιών του υλικού σημείου είναι κλειστές και αυτοτέμνονται σε ένα συγκεκριμένο αριθμό από σημεία. Θα αφήσουμε τον αναγνώστη να βρει τις σχέσεις αυτές και να τις επιβεβαιώσει από τα αριθμητικά αποτελέσματα που θα προκύψουν από το παρακάτω πρόγραμμα, το οποίο βρίσκεται στο αρχείο `Lissajoux.f90` στο συνοδευτικό λογισμικό.

```
!=====
! File Lissajoux.f90
! Lissajoux curves (special case)
! x(t)= cos(o1 t), y(t)= sin(o2 t)
!
program Lissajoux
  implicit none
!
! Declaration of variables
  real x0,y0,R,x,y,vx,vy,t,t0,tf,dt
  real o1,o2,T1,T2
  real, parameter :: PI=3.1415927
!
! Ask user for input:
  print *, '# Enter omega1 and omega2:'
  read *, o1,o2
  print *, '# Enter tf,dt:'
  read *, tf,dt
  print *, '# o1= ',o1, ' o2= ',o2
  print *, '# t0= ',0.0, ' tf= ',tf, ' dt= ',dt
!
! Initialize
  if(o1.le.0.0 .or. o2.le.0.0) stop 'omega1 or omega2<=0'
  T1 = 2.0*PI/o1
  T2 = 2.0*PI/o2
  print *, '# T1= ',T1, ' T2= ',T2
  open(unit=11,file='Lissajoux.dat')
!
! Compute:
  t = 0.0
```

```

do while(t .le. tf)
  x = cos(o1*t)
  y = sin(o2*t)
  vx = -o1*sin(o1*t)
  vy = o2*cos(o2*t)
  write(11,*)t,x,y,vx,vy
  t = t + dt
enddo
close(11)
end program Lissajous

```

Στο παραπάνω πρόγραμμα έχουμε θέσει  $A = 1$ . Ο χρήστης εισάγει τις δύο συχνότητες  $\omega_1$  και  $\omega_2$  και τους αντίστοιχους χρόνους. Εύκολα μπορούμε να το μελετήσουμε με τις εντολές

```

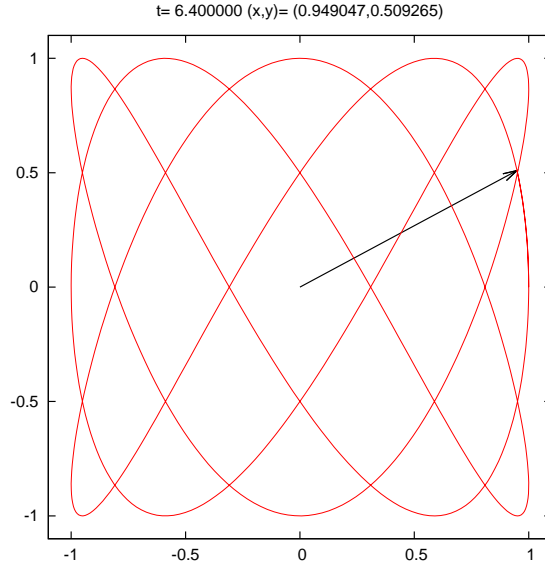
> gfortran Lissajous.f90 -o lsj
> ./lsj
# Enter omega1 and omega2:
3 5
# Enter tf,dt:
10.0 0.01
# o1= 3. o2= 5.
# t0= 0. tf= 10. dt= 0.00999999978
# T1= 2.09439516 T2= 1.2566371
>gnuplot
gnuplot> plot "Lissajous.dat" using 1:2 w l t "x(t)"
gnuplot> replot "Lissajous.dat" using 1:3 w l t "y(t)"
gnuplot> plot "Lissajous.dat" using 1:4 w l t "v_x(t)"
gnuplot> replot "Lissajous.dat" using 1:5 w l t "v_y(t)"
gnuplot> plot "Lissajous.dat" using 2:3 w l t "x-y for 3:5"
gnuplot> file = "Lissajous.dat"
gnuplot> set xrange [-1.1:1.1]; set yrange [-1.1:1.1]
gnuplot> t0=0;tf=10;dt=0.1
gnuplot> load "animate2D.gnu"

```

Στο σχήμα 2.11 δείχνουμε το αποτέλεσμα για την τροχιά του ασύμμετρου αρμονικού ταλαντωτή με  $\omega_1 = 3$  και  $\omega_2 = 5$ .

## 2.2 Κίνηση στο Χώρο

Στην παράγραφο αυτή θα κάνουμε μια απλή γενίκευση των μεθόδων που περιγράψαμε στις προηγούμενες παραγράφους για να μελετήσουμε την κίνηση ενός υλικού σωματιδίου στο χώρο. Οι μόνες αλλαγές που θα κάνουμε, θα είναι η προσθήκη μιας ακόμα εξίσωσης για τη συντεταγμένη  $z(t)$  και τη συνιστώσα της ταχύτητας  $v_z(t)$  και η μέθοδος απεικόνισης



Σχήμα 2.11: Οι τροχιά του ασύμμετρου αρμονικού ταλαντωτή με  $\omega_1 = 3$  και  $\omega_2 = 5$ .

των τροχιών στο gnuplot. Τα προγράμματα που θα γράψουμε θα έχουν παρόμοια δομή με αυτά που γράψαμε για την κίνηση σωματιδίου στο επίπεδο.

Το πρώτο παράδειγμα που θα εξετάσουμε είναι το κωνικό εκκρεμές του σχήματος 2.12. Αυτό κινείται στο επίπεδο  $xy$  με σταθερή γωνιακή ταχύτητα  $\omega$ . Οι εξισώσεις κίνησης δίνονται από τις σχέσεις

$$T_z = T \cos \theta = mg \quad T_{xy} = T \sin \theta = m\omega^2 r, \quad (2.13)$$

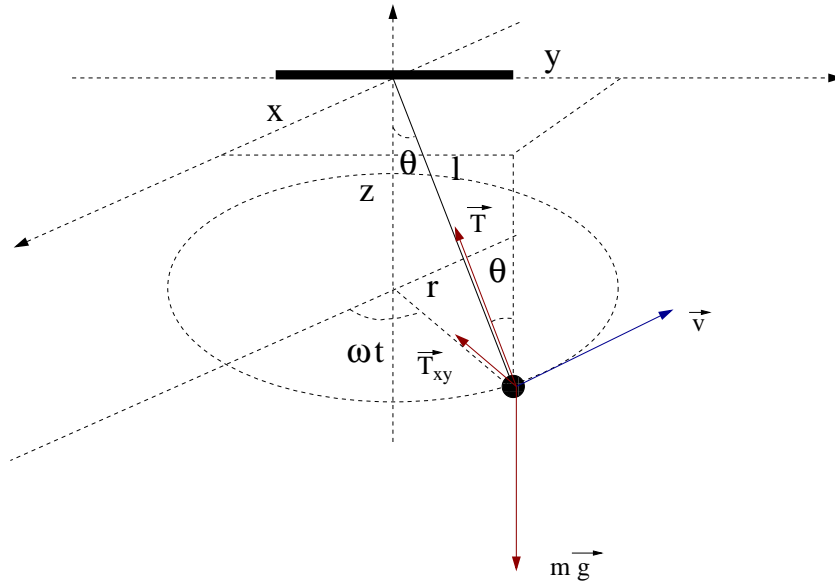
όπου φυσικά  $r = l \sin \theta$ . Λύνοντας τις παραπάνω εξισώσεις<sup>16</sup> βρίσκουμε:

$$\begin{aligned} x(t) &= r \cos \omega t \\ y(t) &= r \sin \omega t \\ z(t) &= -l \cos \theta \end{aligned} \quad (2.14)$$

όπου στις παραπάνω αντικαθιστούμε τις τιμές

$$\begin{aligned} \cos \theta &= \frac{g}{\omega^2 l} \\ \sin \theta &= \sqrt{1 - \cos^2 \theta} \\ r &= \frac{g \sin \theta}{\omega^2 \cos \theta} \end{aligned} \quad (2.15)$$

<sup>16</sup>Φυσικά επιλέγοντας τις κατάλληλες αρχικές συνθήκες (γράψτε τις...).



Σχήμα 2.12: Το κωνικό εκκρεμές του προγράμματος ConicalPendulum.f90.

Για τις ταχύτητες έχουμε

$$\begin{aligned} v_x &= -r\omega \sin \omega t \\ v_y &= r\omega \cos \omega t \\ v_z &= 0. \end{aligned} \quad (2.16)$$

Από τα παραπάνω προκύπτει ότι

$$\omega \geq \omega_{\min} = \sqrt{\frac{g}{l}}, \quad (2.17)$$

και ότι καθώς  $\omega \rightarrow \infty$ ,  $\theta \rightarrow \pi/2$ .

Στο πρόγραμμα που θα γράψουμε ο χρήστης δίνει τις παραμέτρους  $l$  και  $\omega$ , τον τελικό χρόνο  $t_f$  και το βήμα χρόνου  $\delta t$  (παίρνουμε  $t_0 = 0$ ). Η σύμβαση για την έξοδο των δεδομένων από το πρόγραμμα είναι ότι γράφονται σε ένα αρχείο, μία γραμμή ανά χρονική στιγμή, όπου οι 7 πρώτες στήλες είναι οι τιμές των  $t$ ,  $x$ ,  $y$ ,  $z$ ,  $v_x$ ,  $v_y$  και  $v_z$ . Επειδή η γραμμή είναι μεγάλη και δε θέλουμε να τη σπάσει η Fortran, δίνουμε την κατάλληλη εντολή FORMAT. Δείτε τη συζήτηση στη σελίδα 52. Το πρόγραμμα παρατίθεται παρακάτω:

```
!=====
```

```

!File ConicalPendulum.f90
!Set pendulum angular velocity omega and display motion in 3D
!
program ConicalPendulum
  implicit none
!
!Declaration of variables
  real :: l,r,x,y,z,vx,vy,vz,t,tf,dt
  real :: theta,cos_theta,sin_theta,omega
  real, parameter :: PI=3.1415927,g=9.81
!
!Ask user for input:
  print *,'# Enter l,omega: '
  read *,l,omega
  print *,'# Enter tf,dt: '
  read *,tf,dt
  print *,'# l= ',l,' omega= ',omega
  print *,'# T= ',2.0*PI/omega,' omega_min= ',sqrt(g/l)
  print *,'# t0= ',0.0,' tf= ',tf,' dt= ',dt
!
!Initialize
  cos_theta = g/(omega*omega*l)
  if( cos_theta .ge. 1) stop 'cos(theta)>= 1'
  sin_theta = sqrt(1.0-cos_theta*cos_theta)
  z = -g/(omega*omega) !they remain constant throught
  vz= 0.0 !the motion
  r = g/(omega*omega)*sin_theta/cos_theta
  open(unit=11,file='ConicalPendulum.dat')
!
!Compute:
  t = 0.0
  do while(t .le. tf)
    x = r*cos(omega*t)
    y = r*sin(omega*t)
    vx = -r*sin(omega*t)*omega
    vy = r*cos(omega*t)*omega
    write(11,100)t,x,y,z,vx,vy,vz
    t = t + dt
  enddo
  close(11)
100 FORMAT(20G15.7)
end program ConicalPendulum

```

Για να το μεταγλωττίσουμε και να το τρέξουμε, κάνουμε τα γνωστά:

```

> gfortran ConicalPendulum.f90 -o cpd
> ./cpd
# Enter l,omega:

```

```

1.0 6.28
# Enter tf,dt:
10.0 0.01
# l= 1. omega= 6.28000021
# T= 1.00050724 omega_min= 3.132092
# t0= 0. tf= 10. dt= 0.00999999978

```

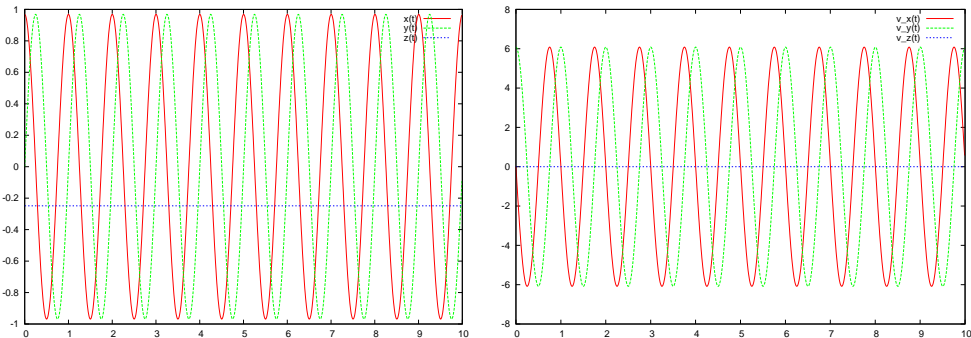
Τα αποτελέσματα θα τα βρούμε στο αρχείο ConicalPendulum.dat. Για να δούμε τις γραφικές παραστάσεις των συναρτήσεων  $x(t)$ ,  $y(t)$ ,  $z(t)$ ,  $v_x(t)$ ,  $v_y(t)$ ,  $v_z(t)$ , εκτελούμε τις γνωστές εντολές μέσα από το gnuplot:

```

> gnuplot
gnuplot> plot "ConicalPendulum.dat" u 1:2 w l t "x(t)"
gnuplot> replot "ConicalPendulum.dat" u 1:3 w l t "y(t)"
gnuplot> replot "ConicalPendulum.dat" u 1:4 w l t "z(t)"
gnuplot> plot "ConicalPendulum.dat" u 1:5 w l t "v_x(t)"
gnuplot> replot "ConicalPendulum.dat" u 1:6 w l t "v_y(t)"
gnuplot> replot "ConicalPendulum.dat" u 1:7 w l t "v_z(t)"

```

Το αποτέλεσμα φαίνονται στο σχήμα 2.13.



Σχήμα 2.13: Οι γραφικές παραστάσεις των συναρτήσεων  $x(t)$ ,  $y(t)$ ,  $z(t)$ ,  $v_x(t)$ ,  $v_y(t)$ ,  $v_z(t)$  προγράμματος ConicalPendulum.f90 για  $\omega = 6.28$ ,  $l = 1.0$ .

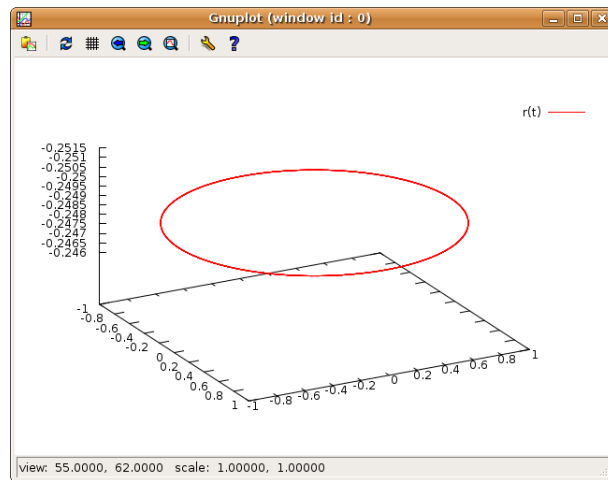
Για να δούμε την τρισδιάστατη τροχιά στο χώρο, θα χρησιμοποιήσουμε την εντολή splot στο gnuplot:

```

gnuplot> splot "ConicalPendulum.dat" u 2:3:4 w l t "r(t)"

```

Το αποτέλεσμα φαίνεται στο σχήμα 2.14. Μπορούμε να κάνουμε κλικ πάνω στην τροχιά και να περιστρέψουμε την καμπύλη, ώστε να τη δούμε από διαφορετικές οπτικές γωνίες. Μπορούμε να αλλάξουμε τα όρια



Σχήμα 2.14: Η γραφική παράσταση της τροχιάς  $\vec{r}(t)$  του υλικού σημείου του προγράμματος ConicalPendulum.f90 για  $\omega = 6.28$ ,  $l = 1.0$ . Φαίνεται το παράθυρο του gnuplot όπου μπορούμε να κάνουμε κλικ πάνω στην τροχιά και να περιστρέψουμε την καμπύλη, ώστε να τη δούμε από διαφορετικές οπτικές γωνίες. Κάτω αριστερά βλέπουμε την οπτική διεύθυνση που δίνεται αντίστοιχα από τις γωνίες  $\theta = 55.0$  μοίρες (γωνία με άξονα  $z$ ) και  $\phi = 62$  μοίρες (γωνία με άξονα  $x$ ), όμοια με τις σφαιρικές συντεταγμένες  $(\theta, \phi)$ .

στους άξονες ορίζοντάς τα ρητά στην εντολή `splot`:

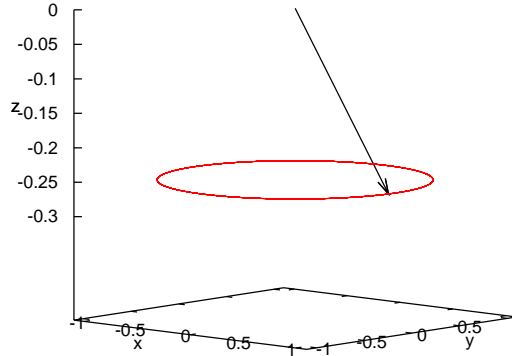
```
gnuplot> splot [-1.1:1.1][-1.1:1.1][-0.3:0.0] \
  "ConicalPendulum.dat" using 2:3:4 w l t "r(t)"
```

Στο συνοδευτικό λογισμικό περιλαμβάνεται και το αρχείο `animate3D.gnu` με το οποίο μπορούμε να κάνουμε απλά animations της τροχιάς της κίνησης ενός υλικού σημείου στο χώρο. Οι εντολές που πρέπει να δώσουμε είναι ανάλογες με αυτές που δίνουμε στην περίπτωση του `animate2D.gnu` με τη μόνη διαφορά ότι καλό είναι να ορίσουμε τα όρια και στον άξονα των  $z$ . Για να δούμε την τροχιά του κωνικού εκκρεμούς από τα δεδομένα που παραγάγαμε παραπάνω, δίνουμε τις εντολές:

```
gnuplot> set xrange [-1.1:1.1]; set yrange [-1.1:1.1]
gnuplot> set zrange [-0.3:0]
gnuplot> t0=0;tf=10;dt=0.1
gnuplot> load "animate3D.gnu"
```

Το αποτέλεσμα το βλέπουμε στο σχήμα 2.15. Περιττό να πούμε πως το πρόγραμμα `animate3D.gnu` μπορεί να χρησιμοποιηθεί πάνω σε οποιο-

t= 10.100000 (x,y,z)= (0.964311,-0.090732,-0.248742)



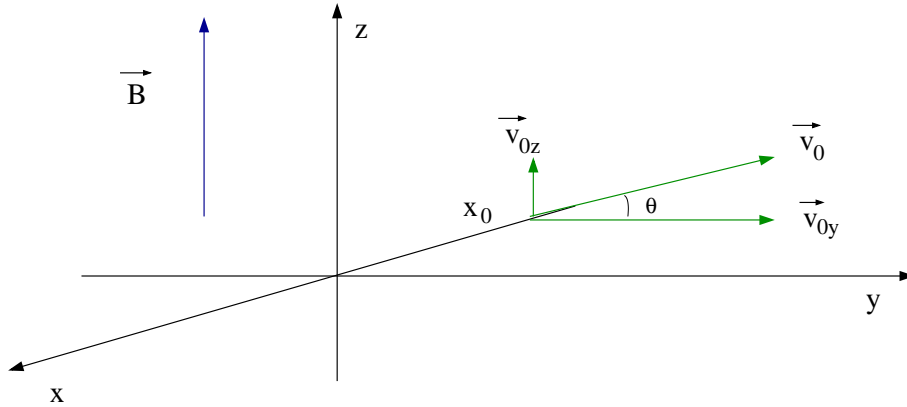
Σχήμα 2.15: Η τροχιά  $\vec{r}(t)$  του υλικού σημείου του προγράμματος ConicalPendulum.f90 για  $\omega = 6.28$ ,  $l = 1.0$  όπως φαίνεται με το πρόγραμμα animate3D.gnu. Στον τίτλο βλέπουμε την τρέχουσα χρονική στιγμή και τις συντεταγμένες του υλικού σημείου.

δῆποτε αρχείο που περιέχει δεδομένα της μορφής t x y z στις πρώτες τέσσερις στήλες του. Απλά αλλάζουμε την τιμή της μεταβλητής file στο όνομα του αρχείου που θα μελετήσουμε. Όπως και με το animate2D.gnu μπορούμε να αλλάξουμε μόνο όποιες από τις μεταβλητές file, t0, tf, dt είναι αναγκαίο πριν επαναλάβουμε το animation με την εντολή load "animate3D.gnu".

Στη συνέχεια, θα μελετήσουμε την τροχιά ενός φορτισμένου σωματιδίου που εισέρχεται σε ένα ομογενές μαγνητικό πεδίο  $\vec{B} = B\hat{z}$  τη χρονική στιγμή  $t_0 = 0$  από τη θέση  $\vec{r}_0 = x_0\hat{x}$  με ταχύτητα  $\vec{v}_0 = v_{0y}\hat{y} + v_{0z}\hat{z}$  όπως φαίνεται στο σχήμα 2.16. Στο φορτίο εξασκείται από το μαγνητικό πεδίο η δύναμη Lorentz  $\vec{F} = q(\vec{v} \times \vec{B}) = qBv_y\hat{x} - qBv_x\hat{y}$ . Οι εξισώσεις κίνησης είναι

$$\begin{aligned} a_x &= \frac{dv_x}{dt} = \omega v_y & \omega &\equiv \frac{qB}{m} \\ a_y &= \frac{dv_y}{dt} = -\omega v_x \\ a_z &= 0. \end{aligned} \tag{2.18}$$





Σχήμα 2.16: Σωματίδιο τη χρονική στιγμή  $t_0 = 0$  στη θέση  $\vec{r}_0 = x_0 \hat{x}$  με ταχύτητα  $\vec{v}_0 = v_{0y} \hat{y} + v_{0z} \hat{z}$  μέσα σε ομογενές μαγνητικό πεδίο  $\vec{B} = B \hat{z}$ .

Ολοκληρώνοντας τις παραπάνω εξισώσεις και λαμβάνοντας υπόψη τις αρχικές συνθήκες παίρνουμε

$$\begin{aligned} v_x(t) &= v_{0y} \sin \omega t \\ v_y(t) &= v_{0y} \cos \omega t \\ v_z(t) &= v_{0z}. \end{aligned} \quad (2.19)$$

Με μία ακόμα ολοκλήρωση παίρνουμε τη θέση του σωματιδίου σε συνάρτηση με το χρόνο

$$\begin{aligned} x(t) &= \left( x_0 + \frac{v_{0y}}{\omega} \right) - \frac{v_{0y}}{\omega} \cos \omega t = x_0 \cos \omega t \\ y(t) &= \frac{v_{0y}}{\omega} \sin \omega t = -x_0 \sin \omega t \quad \text{με} \quad x_0 = -\frac{v_{0y}}{\omega} \\ z(t) &= v_{0z} t, \end{aligned} \quad (2.20)$$

όπου κάναμε την επιλογή  $x_0 = -v_{0y}/\omega$  για να κάνουμε το κέντρο της κυκλικής τροχιάς να συμπίπτει με την αρχή των αξόνων. Το γεωμετρικό σχήμα της τροχιάς είναι μια έλικα με ακτίνα  $R = -x_0$  και βήμα  $v_{0z}T = 2\pi v_{0z}/\omega$ .

Με τα παραπάνω είναι εύκολο τώρα να γράψουμε ένα πρόγραμμα που να υπολογίζει την τροχιά του παραπάνω φορτίου. Οι παράμετροι που θα εισάγει ο χρήστης είναι το μέτρο της ταχύτητας  $v_0$ , τη γωνία  $\theta$  σε μοίρες (βλ. σχήμα 2.16) και τη συχνότητα  $\omega$ . Προφανώς θα έχουμε  $v_{0y} = v_0 \cos \theta$  και  $v_{0z} = v_0 \sin \theta$ . Η αρχική θέση υπολογίζεται τότε από την  $x_0 = -v_{0y}/\omega$ . Το πρόγραμμα δίνεται αυτούσιο παρακάτω και μπορείτε να το βρείτε στο αρχείο ChargeInB.f90 στο συνοδευτικό λογισμικό:

```

=====
! File ChargeInB.f90
! A charged particle of mass m and charge q enters a magnetic
! field B in +z direction. It enters with velocity
!  $v_{0x}=0, v_{0y}=v_0 \cos(\theta), v_{0z}=v_0 \sin(\theta)$ ,  $0 \leq \theta < \pi/2$ 
! at the position  $x_0 = -v_{0y}/\omega$ ,  $\omega = q B/m$ 
!
! Enter v0 and theta and see trajectory from
! t0=0 to tf at step dt
!
=====
program ChargeInB
  implicit none
!
! Declaration of variables
  real :: x,y,z,vx,vy,vz,t,tf,dt
  real :: x0,y0,z0,v0x,v0y,v0z,v0
  real :: theta,omega
  real, parameter :: PI=3.1415927
!
! Ask user for input:
  print *, '# Enter omega: '
  read *, omega
  print *, '# Enter v0, theta (degrees):'
  read *, v0, theta
  print *, '# Enter tf, dt:'
  read *, tf, dt
  print *, '# omega= ', omega, ', T= ', 2.0*PI/omega
  print *, '# v0= ', v0, ', theta= ', theta, 'o (degrees)'
  print *, '# t0= ', 0.0, ', tf= ', tf, ', dt= ', dt
!
! Initialize
  if(theta.lt.0.0 .or. theta.ge.90.0) stop 'Illegal 0<theta<90'
  theta = (PI/180.0)*theta !convert to radians
  v0y = v0*cos(theta)
  v0z = v0*sin(theta)
  print *, '# v0x= ', 0.0, ', v0y= ', v0y, ', v0z= ', v0z
  x0 = - v0y/omega
  print *, '# x0= ', x0, ', y0= ', 0.0, ', z0= ', 0.0
  print *, '# xy plane: Circle with center (0,0) and R= ', ABS(x0)
  print *, '# step of helix: s=v0z*T= ', v0z*2.0*PI/omega
  open(unit=11, file='ChargeInB.dat')
!
! Compute:
  t = 0.0
  vz = v0z
  do while(t .le. tf)
    x = x0*cos(omega*t)
    y = -x0*sin(omega*t)

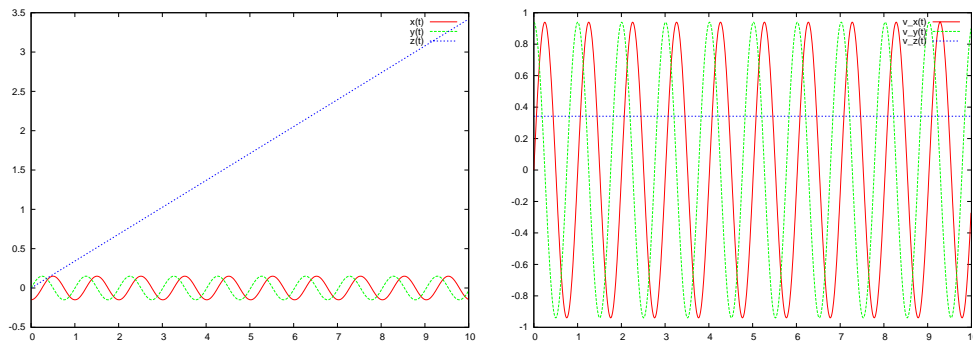
```

```

z = v0z*t
vx = v0y*sin(omega*t)
vy = v0y*cos(omega*t)
write(11,100)t,x,y,z,vx,vy,vz
t = t + dt
enddo
close(11)
100 FORMAT(20G15.7)
end program ChargeInB

```

Παραθέτουμε εδώ τις εντολές μιας τυπικής συνεδρίας τα αποτελέσματα της οποίας δείχνονται στο σχήμα 2.17 και στο σχήμα 2.18.



Σχήμα 2.17: Οι γραφικές παραστάσεις των συναρτήσεων  $x(t), y(t), z(t), v_x(t), v_y(t), v_z(t)$  του προγράμματος ChargeInB.f90 για  $\omega = 6.28$ ,  $x_0 = 1.0$ ,  $\theta = 20$  μοίρες.

```

> gfortran ChargeInB.f90 -o chg
> ./chg
# Enter omega:
6.28
# Enter v0, theta (degrees):
1.0 20
# Enter tf,dt:
10 0.01
# omega= 6.28000021 T= 1.00050724
# v0= 1. theta= 20.0 (degrees)
# t0= 0. tf= 10. dt= 0.00999999978
# v0x= 0. v0y= 0.939692616 v0z= 0.342020124
# x0= -0.149632573 y0= 0. z0= 0.
# xy plane: Circle with center (0,0) and R= 0.149632573
# step of helix: s=v0z*T= 0.342193604
> gnuplot
gnuplot> plot "ChargeInB.dat" u 1:2 w lines title "x(t)"

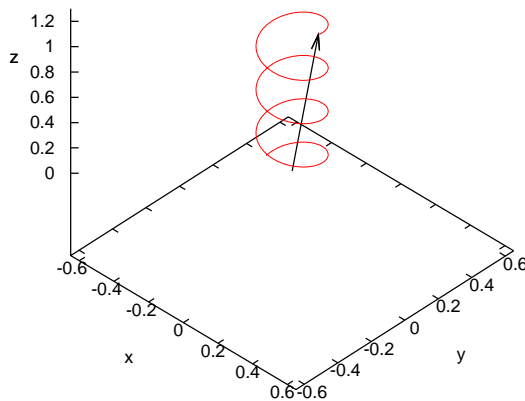
```

```

gnuplot> replot "ChargeInB.dat" u 1:3 w lines title "y(t)"
gnuplot> replot "ChargeInB.dat" u 1:4 w lines title "z(t)"
gnuplot> plot "ChargeInB.dat" u 1:5 w lines title "v_x(t)"
gnuplot> replot "ChargeInB.dat" u 1:6 w lines title "v_y(t)"
gnuplot> replot "ChargeInB.dat" u 1:7 w lines title "v_z(t)"
gnuplot> splot "ChargeInB.dat" u 2:3:4 w lines title "r(t)"
gnuplot> file = "ChargeInB.dat"
gnuplot> set xrange [-0.65:0.65]; set yrange [-0.65:0.65]
gnuplot> set zrange [0:1.3]
gnuplot> t0=0;tf=3.5;dt=0.1
gnuplot> load "animate3D.gnu"

```

t= 3.500000 (x,y,z)= (0.149623,0.001671,1.197069)



Σχήμα 2.18: Η τροχιά  $\vec{r}(t)$  του υλικού σημείου του προγράμματος ChargeInB.f90 για  $\omega = 6.28$ ,  $v_0 = 1.0$ ,  $\theta = 20$  μοίρες όπως φαίνεται με το πρόγραμμα animate3D.gnu. Στον τίτλο βλέπουμε την τρέχουσα χρονική στιγμή και τις συντεταγμένες του υλικού σημείου.

## 2.3 Κίνηση μετ' Εμποδίων

Στην παράγραφο αυτή θα μελετήσουμε την κίνηση ενός σωματιδίου η οποία είναι ευθύγραμμη και ομαλή σε πεπερασμένα διαστήματα, αλλά σε συγκεκριμένα σημεία του χώρου έχουμε ελαστική κρούση σε ένα τοίχωμα αμετακίνητο και αδιαπέραστο. Η κρούση αυτή θα δούμε ότι πε-

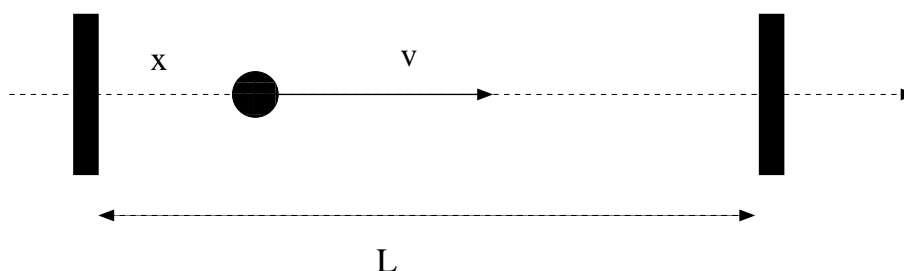
ριγράφεται από τους αλγόριθμους που θα προγραμματίσουμε προσεγγιστικά. Έτσι, θα μελετήσουμε για πρώτη φορά τα συστηματικά σφάλματα που εισάγονται από τους αλγόριθμους που χρησιμοποιούμε<sup>17</sup>, όταν φτιάχνουμε ένα μοντέλο για το σύστημα που μελετάμε.

### 2.3.1 Το Μονοδιάστατο Κουτί

Το πιο απλό παράδειγμα που θα μελετήσουμε είναι η κίνηση ενός σωματιδίου σε ένα “μονοδιάστατο κουτί”. Το σωματίδιο κινείται ελεύθερα πάνω στον άξονα των  $x$  στο διάστημα  $0 < x < L$  όπως φαίνεται στο σχήμα 2.19. Όταν φτάνει στα άκρα του διαστήματος, ανακλάται στα τοιχώματα και η ταχύτητά του αναστρέφεται. Η δυναμική του ενέργεια είναι

$$V(x) = \begin{cases} 0 & 0 < x < L \\ +\infty & \text{αλλού} \end{cases}, \quad (2.21)$$

που έχει σχήμα ενός απειρόβαθου πηγαδιού δυναμικού. Η δύναμη  $F = -dV(x)/dx = 0$  μέσα στο κουτί, όπου το σωματίδιο κινείται ελεύθερα, και  $F = \pm\infty$  στα τοιχώματα, όπου ανακλάται.



Σχήμα 2.19: Σωματίδιο σε μονοδιάστατο κουτί με τοιχώματα στο  $x = 0$  και  $x = L$ .

Το σύστημα είναι πολύ απλό. Αρκεί να δώσουμε την αρχική θέση του σωματιδίου  $x_0$  και την αρχική του ταχύτητα  $v_0$  (το πρόσημό της δηλώνει και την αρχική κατεύθυνση κίνησης). Όσο δε συγκρούεται με τα τοιχώματα η κίνηση είναι ευθύγραμμη και ομαλή και ισχύουν οι απλές σχέσεις

$$\begin{aligned} x(t) &= x_0 + v_0(t - t_0) \\ v(t) &= v_0. \end{aligned} \quad (2.22)$$

<sup>17</sup>Και στις προηγούμενες παραγράφους υπήρχε (μικρό) συστηματικό σφάλμα στα συστήματα που μελετήσαμε, το οποίο οφειλόταν όμως στα σφάλματα των αριθμητικών πράξεων που εκτελούσε ο υπολογιστής - αναπαράσταση πραγματικών, ακρίβεια πράξεων κινητής υποδιαστολής κλπ. Οι αλγόριθμοι ήταν “ακριβείς”.

Επίσης για οποιαδήποτε μεταβολή του χρόνου  $\delta t$ , έτσι ώστε να μην υπάρχει σύγκρουση με τα τοιχώματα μέσα στο χρονικό διάστημα  $(t, t + \delta t)$ , ισχύει (ακριβώς, όχι προσεγγιστικά)

$$\begin{aligned}x(t + \delta t) &= x(t) + v(t)\delta t \\v(t + \delta t) &= v(t).\end{aligned}\tag{2.23}$$

Θα μπορούσαμε λοιπόν να χρησιμοποιήσουμε τις παραπάνω σχέσεις για να γράψουμε το πρόγραμμά μας και όταν το σωματίο συγκρούεται με κάποιο από τα τοιχώματα να αντιστρέψουμε την ταχύτητα:  $v(t) \rightarrow -v(t)$ . Η πηγή των δυσκολιών μας κρύβεται στη λέξη “όταν”. Αφού στον υπολογιστή το χρονικό διάστημα  $\delta t$  είναι αναγκαστικά πεπερασμένο δεν μπορούμε να εντοπίσουμε τη χρονική στιγμή της σύγκρουσης με ακρίβεια μεγαλύτερη από  $\delta t$ : Αν λ.χ. χρησιμοποιώντας τις Σχέσεις (2.23) το σωματίδιο τη χρονική στιγμή  $t + \delta t$  ξεπεράσει το τοίχωμα, η κρούση θα μπορούσε να έχει συμβεί οποιαδήποτε χρονική στιγμή μέσα στο διάστημα  $(t, t + \delta t)$ . Ο αλγόριθμος όμως, θα αλλάξει τη φορά της ταχύτητας τη χρονική στιγμή  $t + \delta t$  εισάγοντας ένα συστηματικό σφάλμα στον υπολογισμό. Πιο συγκεκριμένα, αυτό μπορεί να γίνει με το βρόχο

```
do while(t .le. tf)
  write(11,*)t,x,v
  x = x + v*dt
  t = t + dt
  if(x .lt. 0.0 .or. x .gt. L) v = -v
enddo
```

όπου η συνθήκη σύγκρουσης δίνεται στην προτελευταία γραμμή με τη λογική πρόταση “Αν  $x$  μικρότερο του 0 ή το  $x$  μεγαλύτερο του  $L$ ”. Φαίνεται η αδυναμία του αλγόριθμου να ελέγξει την ακριβή χρονική στιγμή της κρούσης.

Ας δούμε τώρα ολόκληρο το πρόγραμμα, το οποίο βρίσκεται στο αρχείο box1D\_1.f90 του συνοδευτικού λογισμικού. Ο χρήστης μπορεί να καθορίσει το μήκος του κουτιού  $L$  και τις αρχικές συνθήκες  $x_0$ ,  $v_0$ . Δίνει τον αρχικό και τελικό χρόνο της κίνησης  $t_0$ ,  $t_f$ , καθώς και το βήμα υπολογισμού  $dt$  και ο υπολογισμός ... ξεκινάει:

```
!=====
! File box1D_1.f90
! Motion of a free particle in a box 0<x<L
! Use integration with time step dt: x = x + v*dt
!=====
```

```

program box1D
  implicit none
  !
  !Declaration of variables
  real :: L,x0,v0,t0,tf,dt,t,x,v
  !
  !Ask user for input:
  print *, '# Enter L: '
  read *, L
  print *, '# L = ', L
  if( L .le. 0.0) stop 'L must be positive.'
  print *, '# Enter x0,v0: '
  read *, x0,v0
  print *, '# x0= ', x0, ' v0= ', v0
  if(x0 .lt. 0.0 .or. x0 .gt. L) stop 'illegal value of x0.'
  if(v0 .eq. 0.0 ) stop 'illegal value of v0 = 0.'
  print *, '# Enter t0,tf,dt: '
  read *, t0,tf,dt
  print *, '# t0= ', t0, ' tf= ', tf, ' dt= ', dt
  !
  !Initialize
  t = t0
  x = x0
  v = v0
  open(unit=11, file='box1D_1.dat')
  !
  !Compute:
  do while(t .le. tf)
    write(11,*)t,x,v
    x = x + v*dt
    t = t + dt
    if(x .lt. 0.0 .or. x .gt. L) v = -v
  enddo
  close(11)
end program box1D

```

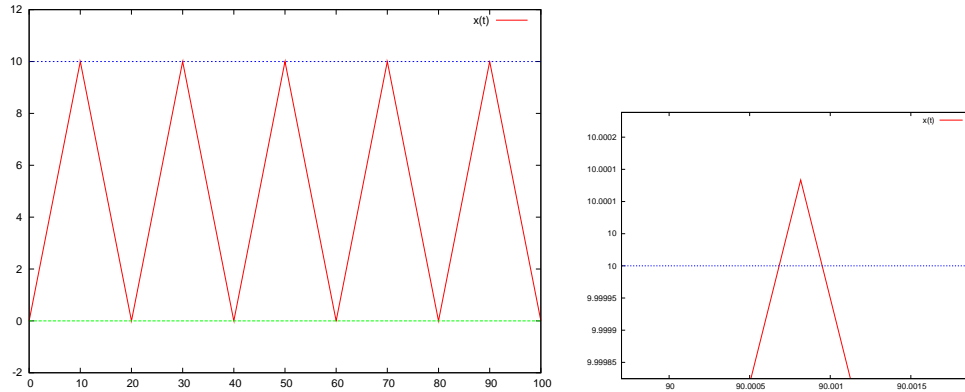
Τα δεδομένα τα βρίσκουμε σε τρεις στήλες στο αρχείο box1D\_1.dat. Η μεταγλώττιση και το τρέξιμο γίνεται κατά τα γνωστά, ενώ η γραφική παράσταση της τροχιάς γίνεται με το gnuplot:

```

> gfortran box1D_1.f90 -o box1
> ./box1
# Enter L:
10
# L = 10.
# Enter x0,v0:
0 1.0
# x0= 0. v0= 1.

```

```
# Enter t0 , tf , dt:
0 100 0.01
# t0= 0. tf= 100. dt= 0.009999999978
> gnuplot
gnuplot> plot "box1D_1.dat" using 1:2 w lines title "x(t)",\
0 notitle,L notitle
gnuplot> plot [:][-1.2:1.2] "box1D_1.dat" \
using 1:3 w lines title "v(t)"
```



Σχήμα 2.20: Η τροχιά  $x(t)$  σωματιδίου σε μονοδιάστατο κουτί με  $L = 10$ ,  $x_0 = 0.0$ ,  $v_0 = 1.0$ ,  $t_0 = 0$ ,  $\delta t = 0.01$ . Στο δεξί διάγραμμα μεγεθύνουμε λεπτομέρεια όταν  $t \approx 90$ . Φαίνονται τα συστηματικά σφάλματα στον προσδιορισμό της χρονικής στιγμής της κρούσης  $t_k = 90$  και της μέγιστης τιμής της  $x(t)$ ,  $x_m = L = 10.0$ .

Τα αποτελέσματα για την τροχιά  $x(t)$  δείχνονται στο σχήμα 2.20. Είναι φανερό η επίδραση του συστηματικού σφάλματος στα αποτελέσματα. Με τις απλές αρχικές συνθήκες που επιλέξαμε, οι κρούσεις συμβαίνουν κάθε  $T/2 = L/v = 10$  μονάδες χρόνου. Άρα στο κομμάτι του διαγράμματος που μεγεθύνουμε στο αριστερό του σχήματος 2.20, η αντιστροφή της κίνησης θα έπρεπε να είχε συμβεί όταν  $t = 90$ ,  $x = L = 10$ .

Ο αναγνώστης θα έχει ήδη καταλάβει ότι το παραπάνω πρόβλημα βελτιώνεται, αν πάρουμε το  $\delta t$  ολοένα και πιο μικρό. Άρα, αν έχουμε την απαραίτητη υπολογιστική δύναμη, μπορούμε να κάνουμε το σφάλμα όσο μικρό θέλουμε; Ναι, αυτό μέχρι ένα σημείο γίνεται. Όμως η επόμενη θέση καθορίζεται από την πράξη  $x+v\delta t$  και η επόμενη χρονική στιγμή από το άθροισμα  $t+\delta t$ . Τι θα γίνει αν παίρνοντας το  $\delta t$  αρκετά μικρό, το γινόμενο  $v\delta t$  γίνει περισσότερο από επτά τάξεις μεγέθους μικρότερο από την τρέχουσα τιμή του  $x$ ; Ή το  $\delta t \sim 10^{-7} t^{18}$ ; Αφού τα  $x$ ,  $v$ ,  $\delta t$  εί-

<sup>18</sup> Αυτό το πρόβλημα θα το λύσουμε αργότερα, όταν για το βήμα  $i$  θα ορίσουμε  $t=t_0+i\delta t$ . Γιατί αυτή η μέθοδος είναι προτιμητέα;



ναι μεταβλητές τύπου REAL, η ακρίβεια στην αναπαράστασή τους είναι περίπου επτά δεκαδικά ψηφία. Για να κάνει ο υπολογιστής πρόσθεση δύο αριθμών, πρέπει να μετατρέψει τους δύο αριθμούς σε αναπαράσταση που θα έχουν τον ίδιο εκθέτη. Άρα ο αριθμός των σημαντικών ψηφίων του μικρότερου αριθμού που θα μπορέσει να αποθηκεύσει στη μνήμη ο υπολογιστής θα μειωθεί<sup>19</sup> και το αποτέλεσμα της πρόσθεσης χάνει μέρος της υπάρχουσας ακρίβειας των δεδομένων.

Αυτό όσο και να προσπαθούμε δε διορθώνεται<sup>20</sup>. Η μόνη μας ελπίδα είναι να επινοήσουμε έναν καλύτερο αλγόριθμο. Αυτό τις περισσότερες φορές φορές δεν είναι δυνατόν, αλλά στο πρόβλημα που μελετάμε η λύση είναι απλή. Λ.χ. θεωρήστε τη σχέση που δίνει τη θέση του κινητού στην ευθύγραμμη ομαλή κίνηση:

$$x(t) = x_0 + v_0(t - t_0). \quad (2.24)$$

Ας χρησιμοποιήσουμε την παραπάνω σχέση για τα τμήματα της κίνησης μεταξύ των συγκρούσεων. Το μόνο που έχουμε να κάνουμε είναι σε κάθε σύγκρουση με ένα από τα τοιχώματα να αντιστρέφουμε τη  $v_0$ , να ορίζουμε το  $x_0$  να είναι η τρέχουσα θέση του κινητού και  $t_0$  να είναι η τρέχουσα χρονική στιγμή. Αυτό επιτυγχάνεται με το βρόχο:

```
t = t0
do while(t .le. tf)
  x = x0 + v0*(t-t0)
  write(11,*)t,x,v0
  if( x .lt. 0.0 .or. x .gt. L)then
    x0 = x
    t0 = t
    v0 = -v0
  endif
  t = t + dt
```

Στον παραπάνω αλγόριθμο δε γλιτώνουμε από το σφάλμα προσδιορισμού της στιγμής της κρούσης, αλλά δεν έχουμε το πρόβλημα “αστάθειας” που είχε ο προηγούμενος όταν  $dt \rightarrow 0$ . Έτσι, έχουμε τη δυνατότητα να απομονώσουμε τη συνεισφορά κάθε τύπου σφαλμάτων. Παρακάτω δί-

<sup>19</sup>Εστω  $x=1.0$  και  $v*dt=3.456789e-5$ . Τότε  $x+v*dt=1.000000+0.000035=1.000035$  και θα χάσουμε 5 σημαντικά ψηφία από την ακρίβεια αναπαράστασης του  $v*dt$ . Τι θα γίνει αν  $v*dt=3.456789e-9$ ;

<sup>20</sup>Φυσικά θα μπορούσαμε να διαλέξουμε μεταβλητές REAL(8), REAL(16), .... Αυτό μπορεί να μας λύσει το πρόβλημα για δεδομένο  $tf$ , αλλά ανάλογο πρόβλημα ακρίβειας θα συναντήσουμε, αν υπολογίσουμε την τροχιά για (αρκετά) μεγαλύτερο  $tf$ . Δοκιμάστε το...

νουμε το πρόγραμμα που χρησιμοποιεί τον παραπάνω αλγόριθμο που μπορείτε να βρείτε στο συνοδευτικό λογισμικό στο αρχείο box1D\_2.f90:

```
!=====
! File box1D_2.f90
! Motion of a free particle in a box   $0 < x < L$ 
! Use constant velocity equation:  $x = x_0 + v_0 \cdot (t - t_0)$ 
! Reverse velocity and redefine  $x_0, t_0$  on boundaries
!
program box1D
  implicit none
!
! Declaration of variables
  real :: L, x0, v0, t0, tf, dt, t, x, v
!
! Ask user for input:
  print *, '# Enter L: '
  read *, L
  print *, '# L = ', L
  if (L .le. 0.0) stop 'L must be positive.'
  print *, '# Enter x0, v0: '
  read *, x0, v0
  print *, '# x0= ', x0, ' v0= ', v0
  if (x0 .lt. 0.0 .or. x0 .gt. L) stop 'illegal value of x0.'
  if (v0 .eq. 0.0) stop 'illegal value of v0 = 0.'
  print *, '# Enter t0, tf, dt: '
  read *, t0, tf, dt
  print *, '# t0= ', t0, ' tf= ', tf, ' dt= ', dt
!
! Initialize
  t = t0
  open(unit=11, file='box1D_2.dat')
!
! Compute:
  do while (t .le. tf)
    x = x0 + v0 * (t - t0)
    write(11, *) t, x, v0
    if (x .lt. 0.0 .or. x .gt. L) then
      x0 = x
      t0 = t
      v0 = -v0
    endif
    t = t + dt
  enddo
  close(11)
end program box1D
```

Το πρόγραμμα το μεταγλωττίζουμε και το τρέχουμε όπως το προηγού-

μενο. Τα αποτελέσματα θα τα βρούμε στο αρχείο `box1D_2.dat`.

Ο παραπάνω αλγόριθμος μπορεί να βελτιωθεί και να μας δώσει την ακριβή λύση. Σας το αφήνουμε για άσκηση<sup>21</sup>.

### 2.3.2 Σφάλματα

Ας δούμε με λίγο περισσότερη λεπτομέρεια την επίδραση των συστηματικών σφαλμάτων στα αποτελέσματα που πήραμε από τη μελέτη του απλού προβλήματος που περιγράψαμε στην προηγούμενη παράγραφο. Θεωρήσαμε δύο κατηγορίες σφαλμάτων: Πρώτα, το συστηματικό σφάλμα στον προσδιορισμό της χρονικής στιγμής της κρούσης. Αυτό μειώνεται όταν ελαττώνουμε το βήμα χρόνου  $\delta t$ . Μετά, το σφάλμα πρόσθεσης αριθμών με μεγάλη διαφορά τάξης μεγέθους και την αστάθεια που αυτή προκαλεί. Αυτό αυξάνεται, όταν ελαττώνουμε αρκετά το βήμα χρόνου  $\delta t$ . Άρα, τα δύο σφάλματα ανταγωνίζονται και ο μελετητής του προβλήματος πρέπει να προσδιορίσει τη βέλτιστη επιλογή για το  $\delta t$ . Αυτή η κατάσταση παρουσιάζεται σε πολλά ενδιαφέροντα προβλήματα για τα οποία δε γνωρίζουμε την ακριβή τους λύση, οπότε είναι πολύ διδακτικό να τη μελετήσουμε στο απλό πρόβλημα που επιλέξαμε.

Όταν δε γνωρίζουμε τη λύση, ο έλεγχος αυτών των συστηματικών σφαλμάτων γίνεται μελετώντας τη συμπεριφορά της λύσης, καθώς ελαττώνεται το  $\delta t$ . Αν παρατηρηθεί σύγκλιση των λύσεων για μια περιοχή αρκετά μικρών  $\delta t$ , τότε καταλήγουμε στο συμπέρασμα ότι η λύση είναι το όριο αυτό με ακρίβεια συγκρινόμενη με τη διαφορά των λύσεων για τα μικρότερα  $\delta t$ .

Χρησιμοποιήσαμε δύο αλγόριθμους, ο πρώτος υλοποιήθηκε στο πρόγραμμα που γράψαμε στο αρχείο `box1D_1.f90` και ο δεύτερος στο `box1D_2.f90`. Θα αναφερόμαστε σε αυτούς στην παράγραφο αυτή ως η “μέθοδος 1” και “μέθοδος 2” αντίστοιχα. Θα δοκιμάσουμε να δούμε πώς μπορούμε να κάνουμε την ανάλυση της σύγκλισης των αποτελεσμάτων καθώς  $\delta t \rightarrow 0$ . Για κάθε μέθοδο θα καθορίσουμε όλες τις παραμέτρους εκτός από το  $\delta t$ . Στην ανάλυση παρακάτω θα χρησιμοποιήσουμε  $L = 10$ ,  $v_0 = 1.0$ ,  $x_0 = 0.0$ ,  $t_0 = 0.0$ ,  $t_f = 95.0$ , άρα το υλικό σημείο θα πρέπει να κάνει μία κρούση ανά 10 μονάδες χρόνου. Θα πάρουμε διαδοχικά μικρότερες τιμές του  $\delta t$  και θα υπολογίσουμε την τελική θέση του κινητού  $x(t \approx 95)$ <sup>22</sup> ως συνάρτηση του  $\delta t$ . Θα μελετήσουμε αν, πόσο καλά και πόσο γρήγορα συγκλίνει αυτή η τιμή σε ένα όριο, καθώς  $\delta t \rightarrow 0$ <sup>23</sup>.

<sup>21</sup>Δείτε το αρχείο `box1D_3.dat` στο συνοδευτικό λογισμικό.

<sup>22</sup>Προσέξτε το  $\approx$ !

<sup>23</sup>Φυσικά εδώ γνωρίζουμε ότι θα πρέπει να πάρουμε  $x(95) = 5$ .

Η ανάλυση αποτελείται από πολλές επαναλαμβανόμενες εντολές: Η μεταγλώττιση του κώδικα, ο καθορισμός των παραμέτρων, το τρέξιμο και ο υπολογισμός τις τιμές  $x(t \approx 95)$  για πολλές τιμές του  $\delta t$ . Σε ένα αρχείο `box1D_anal.in` γράφουμε τις τιμές των παραμέτρων που θα εισάγουμε στο πρόγραμμα:

```
10          L
0 1.0      x0 v0
0 95  0.05 t0 tf dt
```

Στη συνέχεια, μεταγλωττίζουμε το πρόγραμμα

```
> gfortran box1D_1.f90 -o box
```

και το τρέχουμε με την εντολή

```
> cat box1D_anal.in | ./box
```

που στέλνει τα περιεχόμενα του αρχείου `box1D_anal.in` στο `stdin` του προγράμματος `./box`. Το αποτέλεσμα που ζητάμε είναι στην τελευταία γραμμή του αρχείου `box1D_1.dat`:

```
> tail -n 1 box1D_1.dat
94.9511948  5.45000267  -1.
```

Η τρίτη τιμή είναι αυτή της ταχύτητας και δε μας ενδιαφέρει. Σε ένα αρχείο, λ.χ. `box1D_anal.dat` τοποθετούμε το  $\delta t$  και τις δύο πρώτες τιμές από την παραπάνω εντολή. Αλλάζουμε την τιμή του  $\delta t \rightarrow \delta t/2$  στο αρχείο `box1D_anal.in` 12 φορές μέχρι να γίνει ίση με 0.000012 και επαναλαμβάνουμε<sup>24</sup>. Επαναλαμβάνουμε τη διαδικασία<sup>25</sup> για τη μέθοδο 2 τοποθετώντας τα αποτελέσματα για το  $x(t \approx 95)$  σε νέες στήλες στο αρχείο `box1D_anal.dat`. Το αποτέλεσμα είναι

```
# -----
# dt      t1_95      x1(95)      x2(95)
# -----
0.050000  94.95119  5.450003  5.550126
```

<sup>24</sup>Ο σεφ προτείνει: Δοκιμάστε την εντολή `sed 's/0.05/0.025/' box1D_anal.in | ./box` αλλάζοντας το 0.025 με την τιμή του  $\delta t$  που επιθυμείτε να μελετήσετε.

<sup>25</sup>Σπescιαλιτέ: Δείτε το σενάριο φλοιού `box1D_anal.csh` στο συνοδευτικό λογισμικό για ιδέες αυτοματοποίησης της διαδικασίας.

0.025000	94.97849	5.275011	5.174837
0.012500	94.99519	5.124993	5.099736
0.006250	94.99850	4.987460	5.063134
0.003125	94.99734	5.021894	5.035365
0.001563	94.99923	5.034538	5.017764
0.000781	94.99939	4.919035	5.011735
0.000391	94.99979	4.695203	5.005493
0.000195	95.00000	5.434725	5.001935
0.000098	94.99991	5.528124	5.000745
0.000049	94.99998	3.358000	5.000330
0.000024	94.99998	2.724212	5.000232
0.000012	94.99999	9.240705	5.000158

Η μελέτη της σύγκλισης μπορεί να φανεί εποπτικά στο αριστερό σχήμα 2.21. Η 1η μέθοδος βελτιώνει την ακρίβειά της μεγιστοποιώντας την όταν  $\delta t \approx 0.01$ , ενώ για  $\delta t < 0.0001$ , το σφάλμα γίνεται  $> 10\%$  και η μέθοδος δίνει άχρηστα αποτελέσματα. Η 2η μέθοδος έχει πολύ καλύτερη συμπεριφορά από την 1η.

Παρατηρούμε ότι, καθώς μικραίνει το  $\delta t$ , η τελική τιμή του  $t$  πλησιάζει την αναμενόμενη  $t_f = 95$ . Γιατί όμως δεν παίρνουμε  $t = 95$ , ειδικά όταν  $t/\delta t$  είναι ακέραιος αριθμός; Επίσης παρατηρούμε στην 9η γραμμή των δεδομένων του πίνακα ( $dt=0.000195$ ): Αφού  $95/0.000195$  δεν είναι ακέραιος, γιατί  $t = 95$ ; Ακόμα χειρότερα: Αν θεωρητικά αναμένουμε να γίνουν  $95/\delta t$  βήματα, πόσα γίνονται στην πραγματικότητα; Κάθε φορά που κάνετε μια μέτρηση μετρήστε πόσες γραμμές έχει το αρχείο `box1D_1.dat` με την εντολή<sup>26</sup>

```
> wc -l box1D_1.dat
```

και συγκρίνετε με τον αναμενόμενο αριθμό. Το αποτέλεσμα είναι ενδιαφέρον:

#	dt	N	N0
0.050000	1900	1900	
0.025000	3800	3800	
0.012500	7601	7600	
0.006250	15203	15200	
0.003125	30394	30400	
0.001563	60760	60780	
0.000781	121751	121638	
0.000391	243753	242966	

<sup>26</sup>Εναλλακτικά, βάλτε ένα μετρητή μέσα στο βρόχο του προγράμματος.

```
0.000195 485144 487179
0.000098 962662 969387
0.000049 1972589 1938775
0.000024 4067548 3958333
0.000012 7540956 7916666
```

όπου η 2η στήλη έχει τον αριθμό των βημάτων που έγιναν και η 3η τον αριθμό των βημάτων που έπρεπε να γίνουν. Παρατηρούμε ότι η ακρίβεια ελαττώνεται, όταν μικραίνουμε το  $\delta t$  και στο τέλος η διαφορά είναι περίπου 5%! Ειδικά στην τελευταία γραμμή, αν οι πράξεις γίνονταν με μεγαλύτερη ακρίβεια, θα έπρεπε ο τελικός χρόνος να ήταν  $t_f = 0.000012 \times 7540956 \approx 90.5$  κάτι που αλλάζει δραματικά το αποτέλεσμα σε μια περιοδική κίνηση με περίοδο συγκρινόμενη με το σφάλμα στο χρόνο... Επειδή η 1η μέθοδος προχωράει το χρόνο στις εξισώσεις κίνησης ανάλογα με τον αριθμό των βημάτων, καταλαβαίνουμε ότι στην πραγματικότητα η τιμή που παίρνουμε για τη θέση είναι για άλλο χρόνο από αυτόν που νομίζουμε.

Άρα μια σημαντική πηγή σφαλμάτων είναι το σφάλμα συσσώρευσης στον υπολογισμό του χρόνου που γίνεται μεγαλύτερο, καθώς μικραίνει το  $\delta t$ . Πώς θα μπορούσαμε να βελτιώσουμε τη συμπεριφορά αυτή; Μια σημαντική βελτίωση μπορεί να γίνει, αν αντί να υπολογίζουμε το χρόνο προσθετικά, τον υπολογίζουμε πολλαπλασιαστικά. Έστω  $i$  ένας μετρητής των βημάτων που έχουμε κάνει. Τότε

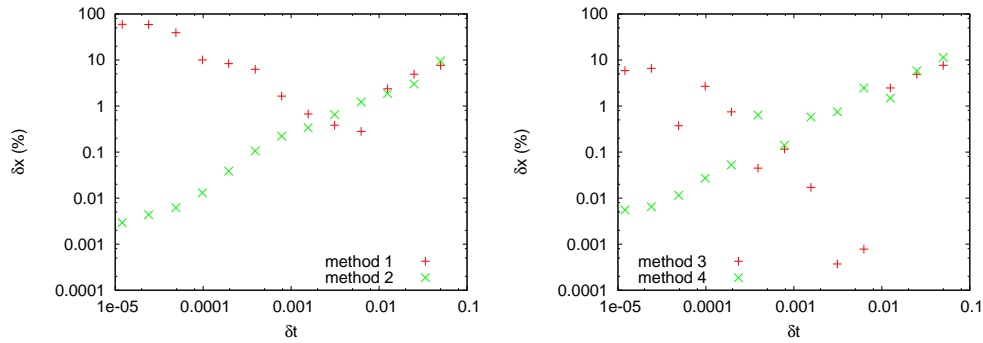
```
! t = t + dt      ! Not accurate, avoid
t = t0 + i*dt     ! Better accuracy, prefer
```

Ο κύριος βρόχος του προγράμματος στο box1D\_1.f90 γίνεται:

```
t = t0
x = x0
v = v0
i = 0
do while(t .le. tf)
  write(11,*)t,x,v
  i = i + 1
  x = x + v*dt
  t = t0 + i*dt
  if(x .lt. 0.0 .or. x .gt. L) v = -v
enddo
```

Το πλήρες πρόγραμμα θα το βρείτε στο αρχείο box1D\_4.f90 του συνοδευτικού λογισμικού. Ας ονομάσουμε τη μέθοδο αυτή “μέθοδο 3”. Παρόμοια αλλαγή κάνουμε και στο αρχείο box1D\_2.f90 που θα βρείτε

στο box1D\_5.f90 και καλούμε τη μέθοδο αυτή “μέθοδο 4”. Επαναλαμβάνουμε την παραπάνω ανάλυση και βρίσκουμε ότι το πρόβλημα ακριβούς προσδιορισμού του χρόνου πρακτικά εξαφανίζεται. Το αποτέλεσμα της ανάλυσης του σφάλματος το δείχνουμε στο δεξί σχήμα 2.21. Η μέθοδος 2 και 4 δεν παρουσιάζει σημαντική διαφορά. Στη μέθοδο 1 και 3 η διαφορά είναι δραματική με το σφάλμα να μειώνεται μέχρι και περισσότερο από 10 φορές. Το πρόβλημα του αυξανόμενου σφάλματος κα-



Σχήμα 2.21: Το επί % σφάλμα  $\delta x = 2|x_i(95) - x(95)|/|x_i(95) + x(95)| \times 100$  όπου  $x_i(95)$  η τιμή που υπολογίζει η μέθοδος  $i = 1, 2, 3, 4$  και  $x(95)$  η τιμή που υπολογίζει ο ακριβής αλγόριθμος σύμφωνα με το κείμενο.

θώς μειώνουμε το  $\delta t$  δεν εξαφανίζεται. Αλλά τώρα καταλαβαίνουμε ότι προέρχεται από τη συσσώρευση σφάλματος στην επαναληπτική σχέση  $x = x + v \cdot \delta t$ . Αυτόν τον τύπο σφάλματος είναι δυσκολότερο να τον βελτιώσουμε και παρουσιάζεται συχνά σε μεθόδους επίλυσης διαφορικών εξισώσεων που θα μελετήσουμε στο επόμενο κεφάλαιο.

### 2.3.3 Το Δισδιάστατο Κουτί

Ας γενικεύσουμε τη μελέτη μας όταν το υλικό σημείο κινείται στις δύο διαστάσεις, στο επίπεδο. Το σωματίο είναι περιορισμένο μέσα σε ένα κουτί  $0 < x < L_x$ ,  $0 < y < L_y$  και σκεδάζεται ελαστικά πάνω στα τοιχώματά του. Βρίσκεται δηλαδή σε ένα ορθογώνιο απειρόβαθο πηγάδι δυναμικού. Ο χρήστης τοποθετεί τη χρονική στιγμή  $t_0$  το υλικό σημείο σε μια αρχική θέση  $(x_0, y_0)$  με αρχική ταχύτητα  $(v_{0x}, v_{0y})$  και το πρόγραμμα υπολογίζει την τροχιά που ακολουθεί το υλικό σημείο μέχρι το χρόνο  $t_f$  με βήμα  $\delta t$ . Μία τέτοια τροχιά φαίνεται στο σχήμα 2.23.

Θα γενικεύσουμε τον αλγόριθμο που χρησιμοποιήσαμε στο πρόγραμμα box2D\_1.f90 για να μελετήσουμε την κίνηση στο μονοδιάστατο

κουτί. Αν είναι γνωστή η θέση και η ταχύτητα του κινητού σε μια χρονική στιγμή  $t$ , τότε τη χρονική στιγμή  $t + \delta t$  η θέση και η ταχύτητά του θα δίνεται από τις σχέσεις

$$\begin{aligned}x(t + \delta t) &= x(t) + v_x(t)\delta t \\y(t + \delta t) &= y(t) + v_y(t)\delta t \\v_x(t + \delta t) &= v_x(t) \\v_y(t + \delta t) &= v_y(t).\end{aligned}\tag{2.25}$$

Η σύγκρουση με τα τοιχώματα προτυποποιείται με ανάκλαση της κάθετης στα τοιχώματα συνιστώσας της ταχύτητας, όταν η αντίστοιχη συντεταγμένη περάσει τα όρια του τοίχου. Φυσικά αυτό εισάγει το συστηματικό σφάλμα που συζητήσαμε στην προηγούμενη παράγραφο. Αποφεύγουμε το συστηματικό σφάλμα συσσώρευσης στον υπολογισμό του χρόνου εισάγοντας έναν μετρητή βημάτων  $i$ , και έτσι το κεντρικό κομμάτι του προγράμματος είναι:

```
i = i + 1
t = t0 + i * dt
x = x + vx*dt
y = y + vy*dt
if(x .lt. 0.0 .or. x .gt. Lx) vx = -vx
if(y .lt. 0.0 .or. y .gt. Ly) vy = -vy
```

Παραθέτουμε το πλήρες πρόγραμμα παρακάτω το οποίο θα βρείτε στο αρχείο box2D\_1.f90. Εκτός από τα εισαγωγικά και τις αρχικοποιήσεις, εισάγαμε και δύο μετρητές κρούσεων με τα τοιχώματα  $n_x$  και  $n_y$ :

```
!=====
! File box2D_1.f90
! Motion of a free particle in a box 0<x<Lx 0<y<Ly
! Use integration with time step dt: x = x + vx*dt y=y+vy*dt
!-----
program box2D
  implicit none
!-----
! Declaration of variables
  real(8) :: Lx, Ly, x0, y0, v0x, v0y, t0, tf, dt, t, x, y, vx, vy
  integer :: i, nx, ny
!-----
! Ask user for input:
  print *, '# Enter Lx, Ly: '
  read *, Lx, Ly
  print *, '# Lx = ', Lx, ', Ly= ', Ly
```



```

if( Lx .le. 0.0) stop 'Lx must be positive.'
if( Ly .le. 0.0) stop 'Ly must be positive.'
print *, '# Enter x0,y0,v0x,v0y:'
read *,x0,y0,v0x,v0y
print *, '# x0= ',x0, ' y0= ',y0, ' v0x= ',v0x, ' v0y= ',v0y
if(x0 .lt. 0.0 .or. x0 .gt. Lx) stop 'illegal value of x0.'
if(y0 .lt. 0.0 .or. y0 .gt. Ly) stop 'illegal value of y0.'
if(v0x**2+v0y**2.eq. 0.0 ) stop 'illegal value of v0=0.'
print *, '# Enter t0,tf,dt:'
read *,t0,tf,dt
print *, '# t0= ',t0, ' tf= ',tf, ' dt= ',dt
!
! Initialize
i = 0
nx = 0 ; ny = 0
t = t0
x = x0 ; y = y0
vx = v0x; vy = v0y
open(unit=11,file='box2D_1.dat')
!
! Compute:
do while(t .le. tf)
  write(11,*)t,x,y,vx,vy
  i = i + 1
  t = t0 + i *dt
  x = x + vx*dt
  y = y + vy*dt
  if(x .lt. 0.0 .or. x .gt. Lx) then
    vx = -vx
    nx = nx + 1
  endif
  if(y .lt. 0.0 .or. y .gt. Ly) then
    vy = -vy
    ny = ny + 1
  endif
enddo
close(11)
print *, '# Number of collisions:'
print *, '# nx= ',nx, ' ny= ',ny
end program box2D

```

Μια τυπική συνεδρία μελέτης μιας τροχιάς δίνεται παρακάτω:

```

> gfortran box2D_1.f90 -o box
> ./box
# Enter Lx,Ly:
10.0 5.0
# Lx = 10. Ly= 5.

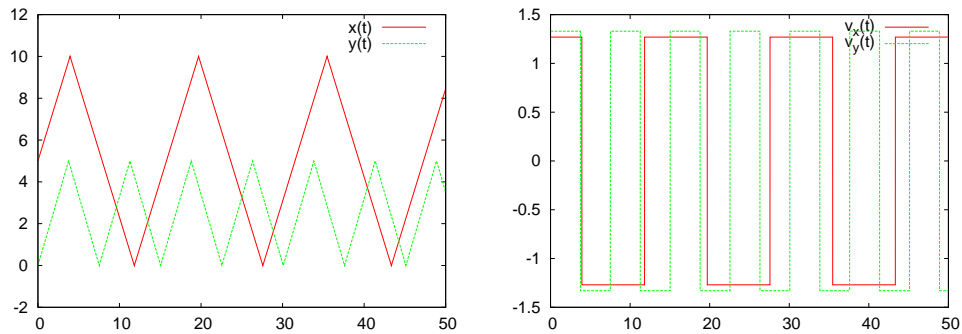
```

```

# Enter x0,y0,v0x,v0y:
5.0 0.0 1.27 1.33
# x0= 5. y0= 0. v0x= 1.27 v0y= 1.33
# Enter t0,tf,dt:
0 50 0.01
# t0= 0. tf= 50. dt= 0.01
# Number of collisions:
# nx= 6 ny= 13
> gnuplot
gnuplot> plot "box2D_1.dat" using 1:2 w lines title "x (t)"
gnuplot> replot "box2D_1.dat" using 1:3 w lines title "y (t)"
gnuplot> plot "box2D_1.dat" using 1:4 w lines title "vx(t)"
gnuplot> replot "box2D_1.dat" using 1:5 w lines title "vy(t)"
gnuplot> plot "box2D_1.dat" using 2:3 w lines title "x-y"

```

Παρατηρούμε την τελευταία γραμμή εξόδου από το πρόγραμμα: Το υλικό σημείο ανακλάται από τα κάθετα τοιχώματα του κουτιού 6 φορές ( $n_x = 6$ ) και από τα οριζόντια 13 ( $n_y = 13$ ). Με τις εντολές αυτές παίρνουμε τις γραφικές παραστάσεις που φαίνονται στα σχήματα 2.22 και 2.23.



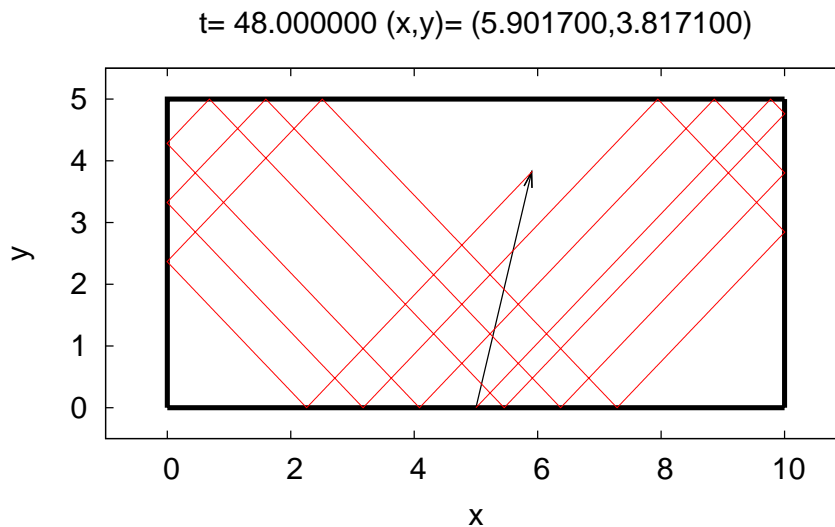
Σχήμα 2.22: Τα αποτελέσματα για το υλικό σημείο που κινείται μέσα σε δισδιάστατο κουτί που παίρνουμε από το πρόγραμμα box2D\_1.f90. Οι παράμετροι είναι  $L_x = 10$ ,  $L_y = 5$ ,  $x_0 = 5$ ,  $y_0 = 0$ ,  $v_{0x} = 1.27$ ,  $v_{0y} = 1.33$ ,  $t_0 = 0$ ,  $t_f = 50$ ,  $\delta t = 0.01$ .

Για να κάνουμε animation της τροχιάς αντιγράφουμε το αρχείο box2D\_animate.gnu από το συνοδευτικό λογισμικό στον κατάλογο που έχουμε τα δεδομένα μας και δίνουμε τις εντολές στο gnuplot:

```

gnuplot> file = "box2D_1.dat"
gnuplot> Lx = 10 ; Ly = 5
gnuplot> t0 = 0 ; tf = 50; dt = 1
gnuplot> load "box2D_animate.gnu"
gnuplot> t0 = 0 ; dt = 0.5; load "box2D_animate.gnu"

```



Σχήμα 2.23: Η τροχιά του υλικού σημείου του σχήματος 2.22 τη χρονική στιγμή  $t = 48$ . Η αρχή του βέλους είναι η αρχική θέση του υλικού σημείου και το τέλος η στιγμιαία του θέση. Το κουτί περιγράφεται από τις παχιές γραμμές.

Στην τελευταία γραμμή επαναλάβουμε το animation στη ... μισή ταχύτητα. Μπορείτε να χρησιμοποιήσετε επίσης το ήδη γνωστό πρόγραμμα `animate2D.gnu` που παρουσιάσαμε στην παράγραφο 2.1.1. Οι αλλαγές που κάναμε στο `box2D_animate.gnu` απλά προσθέτουν ένα σχέδιο του κουτιού και υπολογίζουν αυτόματα τα όρια σχεδιασμού της γραφικής παράστασης. Επίσης, το διάνυσμα που παρακολουθεί την κίνηση του σωματιδίου δεν είναι το διάνυσμα θέσης, αλλά αυτό που ενώνει την αρχική με την τελική θέση του.

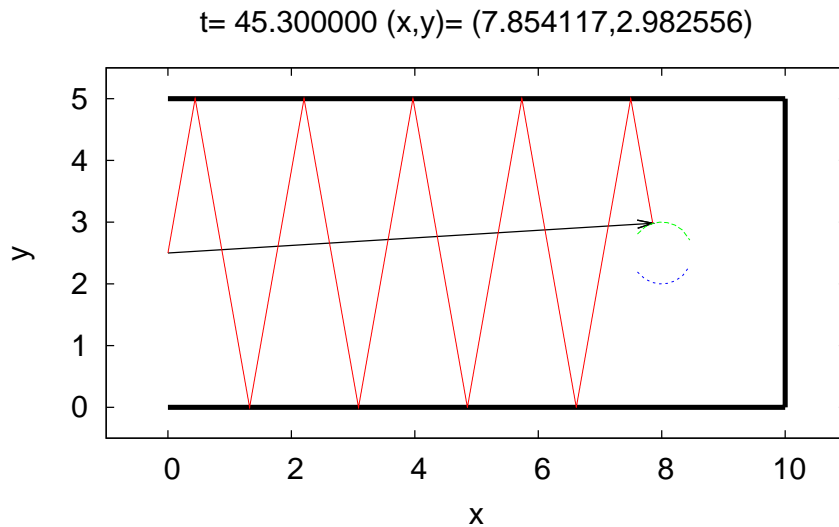
Το επόμενο βήμα είναι να ελέγξουμε την ακρίβεια των αποτελεσμάτων μας. Αυτό γίνεται στο πνεύμα της αντίστοιχης παρουσίασης του μονοδιάστατου προβλήματος και αφήνεται σαν άσκηση για τον αναγνώστη.

## 2.4 Εφαρμογές

Στην παράγραφο αυτή θα παρουσιάσουμε απλά προβλήματα ελεύθερης κίνησης μετ' εμποδίων για εξάσκηση στον προγραμματισμό τους.

Θα αρχίσουμε με ... μίνι γκολφ. Ο παίκτης ρίχνει το (σημειακό) μπα-

λάκι σε ένα ορθογώνιο επίπεδο κουτί μήκους πλευρών  $L_x$ ,  $L_y$  το οποίο είναι ανοιχτό στην πλευρά  $x = 0$ . Το κουτί έχει μια τρύπα κέντρου  $(x_c, y_c)$  και ακτίνας  $R$ . Αν η μπάλα πέσει μέσα, ο παίκτης κερδίζει. Αν η μπάλα βγει από την ανοιχτή πλευρά, ο παίκτης χάνει. Για να ελέγξουμε αν το υλικό σημείο μπήκε στην περιοχή της τρύπας όταν είναι στη θέση  $(x, y)$  αρκεί να ελέγξουμε αν  $(x - x_c)^2 + (y - y_c)^2 \leq R^2$ .



Σχήμα 2.24: Η τροχιά του υλικού σημείου που παράγεται από το πρόγραμμα MiniGolf.f90 με παραμέτρους αυτές που δίνονται στο κείμενο. Απεικονίζεται η στιγμή της ... επιτυχίας! Τη χρονική στιγμή  $t = 45.3$  το υλικό σημείο μπαίνει στην περιοχή της τρύπας με κέντρο  $(8, 2.5)$  και ακτίνα  $0.5$ .

Το υλικό σημείο υποτίθεται πως για  $t_0 = 0$  βρίσκεται στη θέση  $(0, L_y/2)$  και ο παίκτης το εκτοξεύει με ταχύτητα μέτρου  $v_0$  με γωνία  $\theta$  μοίρες που σχηματίζει με τον άξονα  $x$ . Το πρόγραμμα δίνεται παρακάτω:

```
!=====
! File MiniGolf.f
! Motion of a free particle in a box  0<x<Lx 0<y<Ly
! The box is open at x=0 and has a hole at (xc,yc) of radius R
! Ball is shot at (0,Ly/2) with speed v0, angle theta (degrees)
! Use integration with time step dt: x = x + vx*dt y=y+vy*dt
! Ball stops in hole (success) or at x=0 (failure)
```

```

!
program MiniGolf
  implicit none
!
!Declaration of variables
  real(8)          :: Lx,Ly,x0,y0,v0x,v0y
  real(8)          :: t0,tf,dt,t,x,y,vx,vy
  real(8)          :: v0,theta,xc,yc,R,R2
  real(8), parameter :: PI=3.14159265358979324D0
  integer          :: i,nx,ny
  character(7)     :: result
!
!Ask user for input:
  print *, '# Enter Lx,Ly:'
  read  *,Lx,Ly
  print *, '# Lx = ',Lx, ' Ly= ',Ly
  if( Lx .le. 0.0) stop 'Lx must be positive.'
  if( Ly .le. 0.0) stop 'Ly must be positive.'
  print *, '# Enter hole position and radius: (xc,yc), R:'
  read  *,xc,yc,R
  print *, '# (xc,yc)= ( ',xc, ' , ',yc, ' ) R= ',R
  print *, '# Enter v0, theta(degrees):'
  read  *,v0,theta
  print *, '# v0= ',v0, ' theta= ',theta, ' degrees'
  if(v0 .le. 0.0D0 ) stop 'illegal value of v0.'
  if(ABS(theta).ge. 90.0D0) stop 'illegal value of theta.'
  print *, '# Enter dt:'
  read  *,dt
  print *, '# dt= ',dt
!
!Initialize
  t0 = 0.0D0
  x0 = 0.00001D0 ! small but non-zero
  y0 = Ly/2.0
  R2 = R*R
  theta = (PI/180.0D0)*theta
  v0x = v0*cos(theta)
  v0y = v0*sin(theta)
  print *, '# x0= ',x0, ' y0= ',y0, ' v0x= ',v0x, ' v0y= ',v0y
  i = 0
  nx = 0 ; ny = 0
  t = t0
  x = x0 ; y = y0
  vx = v0x; vy = v0y
  open(unit=11,file='MiniGolf.dat')
!
!Compute:
  do while( .TRUE. ) !forever!
    write(11,*)t,x,y,vx,vy

```

```

i = i + 1
t = t0 + i*dt
x = x + vx*dt
y = y + vy*dt
if(x .gt. Lx) then
  vx = -vx
  nx = nx + 1
endif
if(y .lt. 0.0 .or. y .gt. Ly) then
  vy = -vy
  ny = ny + 1
endif
if(x .le. 0.0D0)then
  result = 'Failure'
  exit !exit do loop
endif
if( ((x-xc)*(x-xc)+(y-yc)*(y-yc)) .le. R2)then
  result = 'Success'
  exit !exit do loop
endif
enddo
close(11)
print *, '# Number of collisions:'
print *, '# Result= ', result, ' nx= ', nx, ' ny= ', ny
end program MiniGolf

```

Για να το τρέξουμε κάνουμε τα συνηθισμένα:

```

> gfortran MiniGolf.f90 -o mg
> ./mg
# Enter Lx,Ly:
10 5
# Lx = 10. Ly= 5.
# Enter hole position and radius: (xc,yc), R:
8 2.5 0.5
# (xc,yc)= ( 8. , 2.5 ) R= 0.5
# Enter v0, theta(degrees):
1 80
# v0= 1. theta= 80. degrees
# Enter dt:
0.01
# dt= 0.01
# x0= 1.E-05 y0= 2.5 v0x= 0.173648178 v0y= 0.984807753
# Number of collisions:
# Result= Success nx= 0 ny= 9

```

Φτιάξτε τα διαγράμματα της θέσης και της ταχύτητας με το χρόνο καθώς και της τροχιάς κατά τα γνωστά. Για διασκέδαση του αναγνώστη

παρέχουμε και ειδικό πρόγραμμα animation (μπορείτε να χρησιμοποιήσετε και το `animate2D.gnu`). Αντιγράψτε από το συνοδευτικό λογισμικό το αρχείο `MiniGolf_animate.gnu`, ξεκινήστε το `gnuplot` και δώστε τις εντολές:

```
gnuplot> file = "MiniGolf.dat"
gnuplot> Lx = 10; Ly = 5
gnuplot> xc = 8; yc = 2.5 ; R = 0.5
gnuplot> t0 = 0; dt = 0.1
gnuplot> load "MiniGolf_animate.gnu"
```

Το σχήμα 2.24 αποτελεί αδιάφυστη μαρτυρία της επιτυχίας μας!

Η επόμενη εφαρμογή μας θα είναι τρισδιάστατη. Θα μελετήσουμε την κίνηση ενός υλικού σημείου το οποίο κινείται μέσα σε ένα κύλινδρο ακτίνας  $R$  και ύψους  $L$ . Η κρούσεις στα τοιχώματα και τις βάσεις του κυλίνδρου είναι ελαστικές και ο κύλινδρος είναι αμετακίνητος και απαραμόρφωτος. Στο πρόγραμμα που θα γράψουμε παίρνουμε τον κύλινδρο να έχει τον άξονα συμμετρίας τον άξονα των  $z$ . Η μία βάση του τοποθετείται στο επίπεδο  $z = 0$  και η άλλη στο  $z = L$ . Η διάταξη φαίνεται στο σχήμα 2.26.

Το πρόβλημα δεν παρουσιάζει καμία δυσκολία σε σχέση με τα προηγούμενα όσο αφορά την κίνηση στον άξονα των  $z$ , το σωματίο ανακλάται πάνω στα επίπεδα τοιχώματα των βάσεων του κυλίνδρου. Το μόνο καινούργιο είναι η προβολή της κίνησης στο επίπεδο  $x - y$ , όπου το υλικό σημείο φαίνεται να κινείται σε ένα επίπεδο και να συγκρούεται ελαστικά στο εσωτερικό ενός κύκλου ακτίνας  $R$  και κέντρο πάνω στον άξονα των  $z$ . Η κατάσταση περιγράφεται στο σχήμα 2.25. Κατά την ελαστική ανάκλαση του σωματίου απλά  $v_r \rightarrow -v_r$ , ενώ η  $v_\theta$  μένει η ίδια. Η ταχύτητα του σωματιδίου πριν την κρούση είναι

$$\begin{aligned}\vec{v} &= v_x \hat{x} + v_y \hat{y} \\ &= v_r \hat{r} + v_\theta \hat{\theta}\end{aligned}\quad (2.26)$$

ενώ μετά

$$\begin{aligned}\vec{v}' &= v'_x \hat{x} + v'_y \hat{y} \\ &= -v_r \hat{r} + v_\theta \hat{\theta}\end{aligned}\quad (2.27)$$

Από τις σχέσεις

$$\begin{aligned}\hat{r} &= \cos \theta \hat{x} + \sin \theta \hat{y} \\ \hat{\theta} &= -\sin \theta \hat{x} + \cos \theta \hat{y},\end{aligned}\quad (2.28)$$

και τις σχέσεις  $v_r = \vec{v} \cdot \hat{r}$ ,  $v_\theta = \vec{v} \cdot \hat{\theta}$ , παίρνουμε

$$\begin{aligned} v_r &= v_x \cos \theta + v_y \sin \theta \\ v_\theta &= -v_x \sin \theta + v_y \cos \theta. \end{aligned} \quad (2.29)$$

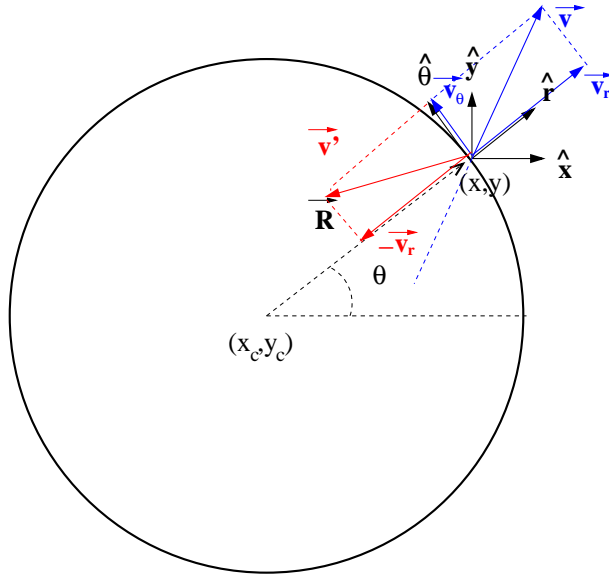
Αντιστρέφοντάς τις παίρνουμε

$$\begin{aligned} v_x &= v_r \cos \theta - v_\theta \sin \theta \\ v_y &= v_r \sin \theta + v_\theta \cos \theta. \end{aligned} \quad (2.30)$$

Με το μετασχηματισμό  $v_r \rightarrow -v_r$  η νέα ταχύτητα σε καρτεσιανές συντεταγμένες θα είναι

$$\begin{aligned} v'_x &= -v_r \cos \theta - v_\theta \sin \theta \\ v'_y &= -v_r \sin \theta + v_\theta \cos \theta. \end{aligned} \quad (2.31)$$

Ο μετασχηματισμός  $v_x \rightarrow v'_x$ ,  $v_y \rightarrow v'_y$  θα υλοποιηθεί σε μια υπορου-



Σχήμα 2.25: Ελαστική ανάκλαση υλικού σημείου που πέφτει στο εσωτερικό κύκλου ακτίνας  $R = |\vec{R}|$  και κέντρου  $\vec{r}_c = x_c \hat{x} + y_c \hat{y}$  στο σημείο  $\vec{r} = x \hat{x} + y \hat{y}$ . Έχουμε  $\vec{R} = (x - x_c) \hat{x} + (y - y_c) \hat{y}$ . Η ταχύτητα αρχικά είναι  $\vec{v} = v_r \hat{r} + v_\theta \hat{\theta}$  όπου  $\hat{r} \equiv \vec{R}/R$ . Μετά την ανάκλαση  $v_r \rightarrow -v_r$  και η νέα ταχύτητα του υλικού σημείου είναι  $\vec{v}' = -v_r \hat{r} + v_\theta \hat{\theta}$

τίνα `reflectVonCircle(vx,vy,x,y,xc,yc,R)`. Στην είσοδο της δίνουμε την αρχική ταχύτητα  $(v_x, v_y)$ , το σημείο κρούσης  $(x, y)$ , το κέντρο του



κύκλου  $(x_c, y_c)$  και την ακτίνα του κύκλου<sup>27</sup>  $R$ . Στην έξοδο, τα  $(v_x, v_y)$  έχουν αντικατασταθεί με τις νέες τιμές<sup>28</sup>  $(v'_x, v'_y)$ .

Το πρόγραμμα που προκύπτει θα το βρείτε στο αρχείο Cylinder3D.f90:

```
!=====
! File Cylinder3D.f90
! Motion of a free particle in a cylinder with axis the z-axis,
! radius R and 0<z<L
! Use integration with time step dt: x = x + vx*dt
!                                     y = y + vy*dt
!                                     z = z + vz*dt
! Use subroutine reflectVonCircle for colisions at r=R
!-----
program Cylinder3D
  implicit none
!-----
! Declaration of variables
  real(8)  :: x0, y0, z0, v0x, v0y, v0z, t0, tf, dt, t, x, y, z, vx, vy, vz
  real(8)  :: L, R, R2, vxy, rxy, r2xy, xc, yc
  integer  :: i, nr, nz
!-----
! Ask user for input:
  print *, '# Enter R, L: '
  read  *, R, L
  print *, '# R= ', R, ' L= ', L
  if( R .le. 0.0) stop 'R must be positive.'
  if( L .le. 0.0) stop 'L must be positive.'
  print *, '# Enter x0, y0, z0, v0x, v0y, v0z: '
  read  *, x0, y0, z0, v0x, v0y, v0z
  rxy = DSQRT(x0*x0+y0*y0)
  print *, '# x0 = ', x0, ' y0 = ', y0, ' z0= ', z0, ' rxy= ', rxy
  print *, '# v0x= ', v0x, ' v0y= ', v0y, ' v0z= ', v0z
  if(rxy .gt. R) stop 'illegal value of rxy > R'
  if(z0 .lt. 0.0D0) stop 'illegal value of z0 < 0'
  if(z0 .gt. L) stop 'illegal value of z0 > L'
  if(v0x**2+v0y**2+v0z**2.eq.0.0) stop 'illegal value of v0 = 0.'
  print *, '# Enter t0, tf, dt: '
  read  *, t0, tf, dt
  print *, '# t0= ', t0, ' tf= ', tf, ' dt= ', dt
!-----
! Initialize
  i = 0
  nr = 0 ; nz = 0
```

<sup>27</sup>Προφανώς ιδανικά θα ισχύει  $R^2 = (x - x_c)^2 + (y - y_c)^2$  αλλά επειδή υπάρχει συστηματικό σφάλμα στη θέση της κρούσης, επιλέγουμε το  $R$  να δίνεται.

<sup>28</sup>Επίσης, το υλικό σημείο, όπως θα δούμε, τοποθετείται ακριβώς πάνω στον κύκλο.

```

t = t0
x = x0 ; y = y0 ; z = z0
vx = v0x; vy = v0y; vz = v0z
R2 = R*R
xc = 0.0D0 !center of circle which is the projection of the
yc = 0.0D0 !cylinder on the xy plane
open(unit=11,file='Cylinder3D.dat')
!-----
!Compute:
do while(t .le. tf)
  write(11,100)t,x,y,z,vx,vy,vz
  i = i + 1
  t = t0 + i *dt
  x = x + vx*dt
  y = y + vy*dt
  z = z + vz*dt
  if(z .lt. 0.0 .or. z .gt. L) then
    vz = -vz ! reflection on cylinder caps
    nz = nz + 1
  endif
  r2xy = x*x+y*y
  if( r2xy .gt. R2)then
    call reflectVonCircle(vx,vy,x,y,xc,yc,R)
    nr = nr + 1
  endif
enddo
close(11)
print *, '# Number of collisions:'
print *, '# nr= ',nr, ' nz= ',nz

100 FORMAT(100G28.16)
end program Cylinder3D
!-----
!=====
!-----
subroutine reflectVonCircle(vx,vy,x,y,xc,yc,R)
  implicit none
  real(8) :: vx,vy,x,y,xc,yc,R
  real(8) :: theta,cth,sth,vr,vth

  theta = atan2(y-yc,x-xc)
  cth = cos(theta)
  sth = sin(theta)

  vr = vx*cth + vy *sth
  vth = -vx*sth + vy *cth

  vx = -vr*cth - vth*sth !reflect vr -> -vr
  vy = -vr*sth + vth*cth

```

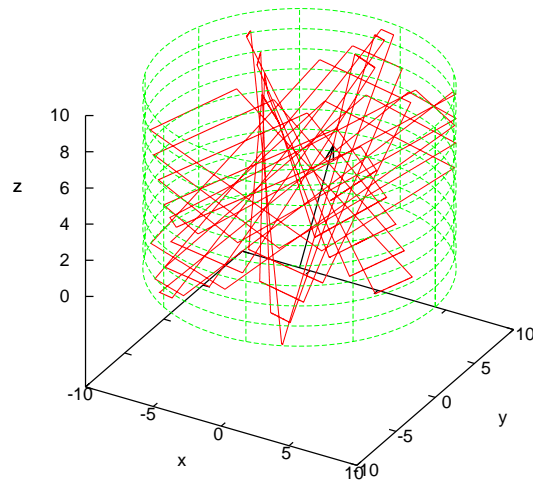
```

x      = xc      + R*cth    !put x,y on the circle
y      = yc      + R*sth
end subroutine reflectVonCircle

```

Να επισημάνουμε μόνο τις ακόλουθες λεπτομέρειες: Χρησιμοποιούμε καταρχήν τη συνάρτηση `atan2` για τον προσδιορισμό της γωνίας  $\theta$ . Η συνάρτηση αυτή με ορίσματα `atan2(y,x)` επιστρέφει τη γωνία  $\theta = \tan^{-1}(y/x)$  στο σωστό τεταρτημόριο που βρίσκεται το σημείο  $(x,y)$ . Η γωνία που ζητάμε δίνεται από την εντολή `atan2(y-yc,x-xc)`. Στη συνέχεια εφαρμόζουμε τις σχέσεις (2.29) και (2.31), ενώ στις τελευταίες δύο γραμμές επιβάλλουμε μετά την κρούση το υλικό σημείο να είναι πάνω στο σημείο του κύκλου  $(x_c + R \cos \theta, y_c + R \sin \theta)$ .

t= 500.000000 (x,y,z)= (2.227212,0.469828,7.088600)



Σχήμα 2.26: Η τροχιά του σωματιδίου που ανακλάται στο εσωτερικό κυλίνδρου με  $R = 10$ ,  $L = 10$  σύμφωνα με το πρόγραμμα `Cylinder3D.f90`. Έχουμε επιλέξει  $\vec{r}_0 = 1.0\hat{x} + 2.2\hat{y} + 3.1\hat{z}$ ,  $\vec{v}_0 = 0.93\hat{x} - 0.89\hat{y} + 0.74\hat{z}$ ,  $t_0 = 0$ ,  $t_f = 500.0$ ,  $\delta t = 0.01$ .

Η μεταγλώττιση και το τρέξιμο γίνονται κατά τα γνωστά:

```

> gfortran Cylinder3D.f90 -o c1
> ./c1
# Enter R, L:
10.0 10.0
# R= 10. L= 10.

```

```
# Enter x0,y0,z0,v0x,v0y,v0z:
1.0 2.2 3.1 0.93 -0.89 0.74
# x0 = 1. y0 = 2.2 z0= 3.1 rxy= 2.41660919
# v0x= 0.93 v0y= -0.89 v0z= 0.74
# Enter t0,tf,dt:
0.0 500.0 0.01
# t0= 0. tf= 500. dt= 0.01
# Number of collisions:
# nr= 33 nz= 37
```

Για να φτιάξουμε τις γραφικές παραστάσεις της θέσης/ταχύτητας συναρτήσει του χρόνου δίνουμε τις εντολές gnuplot κατά τα γνωστά:

```
gnuplot> file="Cylinder3D.dat"
gnuplot> plot file using 1:2 with lines title " x(t)",\
           file using 1:3 with lines title " y(t)",\
           file using 1:4 with lines title " z(t)"
gnuplot> plot file using 1:5 with lines title "v_x(t)",\
           file using 1:6 with lines title "v_y(t)",\
           file using 1:7 with lines title "v_z(t)"
```

Μπορούμε να δούμε και την απόσταση του κινητού από τον άξονα του κυλίνδρου  $r(t) = \sqrt{x(t)^2 + y(t)^2}$  ως συνάρτηση με το χρόνο με την εντολή

```
gnuplot> plot file using 1:(sqrt($2**2+$3**2)) w l t "r(t)"
```

Για να φτιάξουμε το σχήμα της τροχιάς μαζί με τον κύλινδρο δίνουμε τις εντολές

```
gnuplot> L = 10 ; R = 10
gnuplot> set urange [0:2.0*pi]
gnuplot> set vrange [0:L]
gnuplot> set parametric
gnuplot> splot file using 2:3:4 with lines notitle,\
           R*cos(u),R*sin(u),v notitle
```

Με την εντολή `set parametric` μπορούμε να φτιάξουμε τη γραφική παράσταση μιας δισδιάστατης επιφάνειας της μορφής  $\vec{r}(u, v) = x(u, v) \hat{x} + y(u, v) \hat{y} + z(u, v) \hat{z}$ . Ο κύλινδρος (χωρίς τις βάσεις του) δίνεται από τις παραμετρικές εξισώσεις  $\vec{r}(u, v) = R \cos u \hat{x} + R \sin u \hat{y} + v \hat{z}$  με  $u \in [0, 2\pi)$ ,  $v \in [0, L]$ .

Για το animation της τροχιάς αντιγράψτε το αρχείο `Cylinder3D_animate.gnu` στον κατάλογο που κάνετε την ανάλυση και μέσα από το gnuplot δώστε τις εντολές

```
gnuplot> R=10;L=10;t0=0;tf=500;dt=10
gnuplot> load "Cylinder3D_animate.gnu"
```

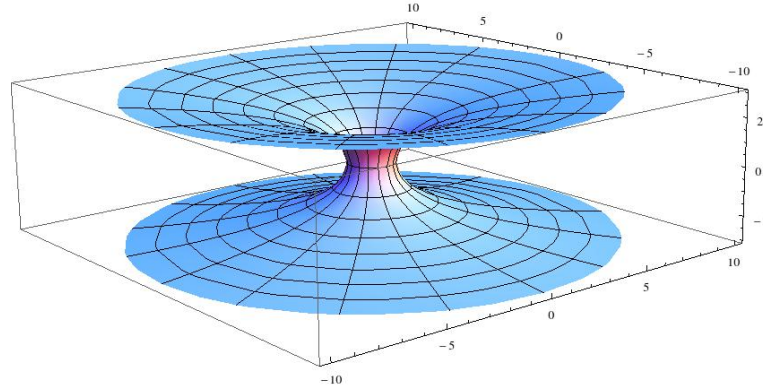
βάζοντας φυσικά τις παραμέτρους που χρησιμοποιήσατε εσείς. Το αποτέλεσμα φαίνεται στο σχήμα 2.26.

Τελευταία εφαρμογή θα είναι ένα απλό πρότυπο χωροχρονικής σκουλικότρυπας (spacetime wormhole). Η χωροχρονική σκουλικότρυπα είναι η απλή χωροχρονική γεωμετρία που συνδέει δύο απομακρυσμένες περιοχές του χώρου οι οποίες είναι ασυμπτωτικά επίπεδες (δηλ. σε αρκετά μεγάλη απόσταση από τα στόμια της σκουλικότρυπας, ο χώρος είναι σχεδόν επίπεδος). Μια τέτοια γεωμετρία φαίνεται στο σχήμα 2.27. Η απόσταση που διανύει κάποιος περνώντας από τα στόμια της τρύπας, μπορεί να είναι πολύ μικρότερη από την απόσταση των στομιών που διανύει κανείς εκτός της σκουλικότρυπας οπότε, θεωρητικά τουλάχιστον, μπορούν να χρησιμοποιηθούν για διαστρικά / διαγαλαξιακά ταξίδια. Ή, όπως φαίνεται και στο σχήμα, να επικοινωνήσουν περιοχές του χώρου που θα ήταν διαφορετικά ασύνδετες μεταξύ τους. Να σημειώσουμε πάντως, ότι παρόλο που τέτοιες γεωμετρίες έχουν εξάψει την φαντασία των φυσικών, αλλά και των συγγραφέων επιστημονικής φαντασίας, η υλοποίησή τους στα πλαίσια της τρέχουσας θεωρίας για τη βαρύτητα, τη Γενική Θεωρία της Σχετικότητας, δεν είναι δυνατή. Για την κατασκευή τους, η εξίσωση του Einstein απαιτεί εξωτικού τύπου ύλη με αρνητική πυκνότητα ενέργειας η οποία δεν έχει παρατηρηθεί στη φύση. Τέτοιες εξωτικές γεωμετρίες είναι δυνατόν να παρουσιάζονται σε μικροσκοπικό επίπεδο ως κβαντικές διακυμάνσεις της γεωμετρίας<sup>29</sup>.

Στην εφαρμογή αυτή θα μελετήσουμε μια πολύ απλή τέτοια γεωμετρία στο επίπεδο, καθώς και την ελεύθερη κίνηση σωματιδίου μέσα σε αυτή<sup>30</sup>. Παίρνουμε το επίπεδο και αφαιρούμε δύο δίσκους ακτίνας  $R$  που τα κέντρα τους απέχουν απόσταση  $d$  όπως φαίνεται στο σχήμα 2.28. Ταυτοποιούμε τα σημεία των κύκλων έτσι ώστε το σημείο 1 στον αριστερό κύκλο να ταυτίζεται με το σημείο 1 στο δεξιό, το σημείο 2 με το σημείο 2 κ.ο.κ. Οι δύο κύκλοι δίνονται από τις παραμετρικές εξισώσεις  $x(\theta) = d/2 + R \cos \theta$ ,  $y(\theta) = R \sin \theta$ ,  $-\pi < \theta \leq \pi$  για το δεξί κύκλο και  $x(\theta) = -d/2 - R \cos \theta$ ,  $y(\theta) = R \sin \theta$ ,  $-\pi < \theta \leq \pi$  για τον αριστερό. Τα σημεία με το ίδιο  $\theta$  στους δύο κύκλους ταυτίζονται. Ένα σωματίο

<sup>29</sup>Διαβάστε το ενδιαφέρον βιβλίο του K.S. Thorne "Black Holes and Time Wraps: Einstein's Outrageous Legacy", W.W. Norton, New York και κοιτάξτε τι είναι ο χωροχρονικός αφρός (spacetime foam) του A. Wheeler.

<sup>30</sup>Η ιδέα αυτή είναι άσκηση στο βιβλίο του J. B. Hartle, "Gravity: An Introduction to Einstein's General Relativity", Addison Wesley 2003, Κεφ. 7, Ασκ. 25.

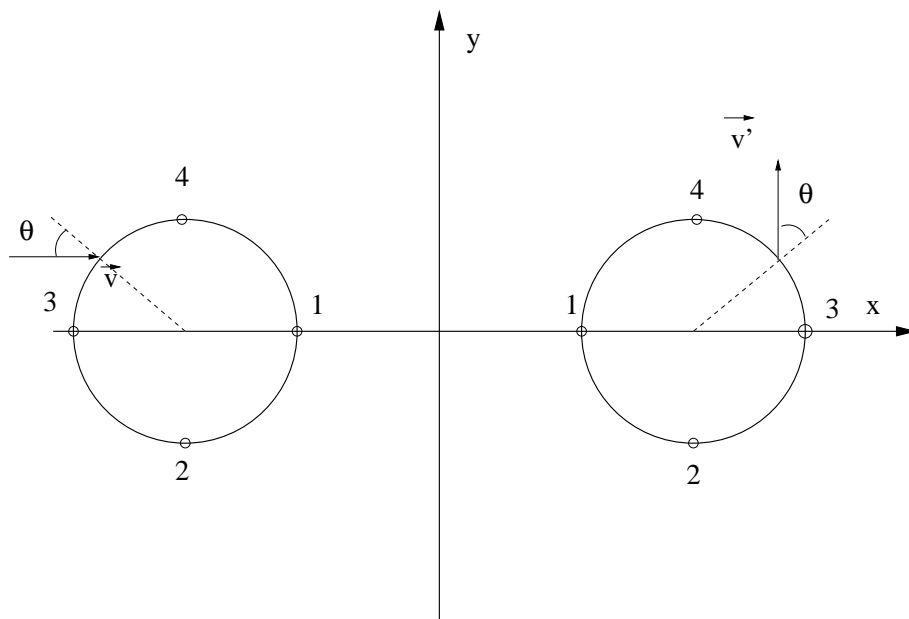


Σχήμα 2.27: Η χωρική γεωμετρία της σκουλικότρυπας. Δύο ασυμπτωτικά επίπεδα μέρη του χώρου συνδέονται με ένα “λαιμό” ό οποίος μπορεί να κανονιστεί να είναι μικρού μήκους.

που βυθίζεται στο αριστερό στόμιο της σκουλικότρυπας με ταχύτητα  $\vec{v}$  αναδύεται άμεσα από το δεξί με ταχύτητα  $\vec{v}'$  όπως φαίνεται στο σχήμα 2.28.

Οι στόχοι μας είναι:

1. Να γράψουμε ένα πρόγραμμα που να υπολογίζει την τροχιά σωματιδίου που κινείται στη γεωμετρία του σχήματος 2.28. Τα όρια της κίνησης είναι  $-L/2 \leq x \leq L/2$  και  $-L/2 \leq y \leq L/2$ . Όταν το σωματίδιο βγει εκτός των παραπάνω ορίων θα πάρουμε περιοδικές συνοριακές συνθήκες, δηλ. θα ταυτίσουμε τα σημεία του ευθύγραμμου τμήματος  $x = -L/2$  με αυτά του  $x = +L/2$  καθώς και αυτά του ευθύγραμμου τμήματος  $y = -L/2$  με αυτά του  $y = +L/2$ . Ο χρήστης θα παρέχει τις παραμέτρους  $R$ ,  $d$  και  $L$  καθώς και τις αρχικές συνθήκες  $(x_0, y_0)$ ,  $(v_0, \phi)$  όπου  $\vec{v}_0 = v_0(\cos \phi \hat{x} + \sin \phi \hat{y})$ . Θα δίνει επίσης τις χρονικές παραμέτρους  $t_f$  και  $dt$  για κίνηση στο χρόνο στο διάστημα  $t \in [t_0 = 0, t_f]$  με βήμα  $dt$ .
2. Στη γεωμετρία με  $L = 20$ ,  $d = 5$ ,  $R = 1$ , να σχεδιάσουμε την τροχιά του σωματιδίου με  $(x_0, y_0) = (0, -1)$ ,  $(v_0, \phi) = (1, 10^\circ)$  με  $t_f = 40$ ,  $dt = 0.05$ .
3. Να βρούμε τροχιά που να είναι κλειστή χωρίς να περνάει από



Σχήμα 2.28: Απλό πρότυπο της χωρικής γεωμετρίας της σκουλικότρυπας που δείχνεται στο σχήμα 2.27. Το σωματίδιο κινείται σε όλο το επίπεδο εκτός από τους δίσκους που έχουμε αφαιρέσει. Ο λαιμός της σκουλικότρυπας προτυποποιείται από τους δύο κύκλους  $x(\theta) = \pm d/2 \pm R \cos \theta$ ,  $y(\theta) = R \sin \theta$ ,  $-\pi < \theta \leq \pi$  και έχει μηδενικό μήκος έχοντας ταυτίσει τα σημεία του χείλους της σκουλικότρυπας. Η ταυτοποίηση γίνεται με συγκεκριμένη φορά ταυτίζοντας τα σημεία με το ίδιο  $\theta$ , έτσι ώστε λ.χ. τα σημεία 1, 2, 3, και 4 να ταυτίζονται (μπορείτε να φανταστείτε τι γίνεται αν διπλώσετε το χαρτί κατά μήκος του άξονα των  $y$  και κολλήσετε τους δύο κύκλους μαζί). Το βύθισμα ενός σωματιδίου μέσα στη σκουλικότρυπα και η ανάδυση από το άλλο χείλος γίνεται όπως φαίνεται για το διάνυσμα  $\vec{v} \rightarrow \vec{v}'$ .

τα άκρα  $|x| = L/2$ ,  $|y| = L/2$  και να εξετάσουμε αν αυτή είναι ευσταθής ως προς μικρές αλλαγές των αρχικών συνθηκών.

4. Να βρούμε άλλες κλειστές τροχιές που να περνάνε μέσα από τα χείλη της σκουλικότρυπας και να εξετάσουμε αν είναι ευσταθείς ως προς μικρές αλλαγές των αρχικών συνθηκών.
5. Να προσθέσουμε στο πρόγραμμα τη δυνατότητα να υπολογίζει την απόσταση που διάνυσε το σωματίδιο. Ένα σωματίο ξεκινάει από τη θέση  $(-x_0, 0)$  και κινείται προς την  $+x$  διεύθυνση μέχρι τη θέση  $(x_0, 0)$  με  $x_0 > R + d/2$ . Εσείς σχεδιάστε στο χαρτί την τροχιά που ακολουθεί και υπολογίστε την απόσταση που διάνυσε. Στη συνέχεια, επιβεβαιώστε τους υπολογισμούς σας με το πρόγραμμα.
6. Σαν άσκηση στο τέλος, να μεταβάλετε το πρόγραμμά έτσι ώστε στα άκρα  $|x| = L/2$ ,  $|y| = L/2$  το σωματίο να ανακλάται ελαστικά. Επανασχεδιάστε τις τροχιές που μελετήσαμε παραπάνω.

Ορίζουμε το δεξί κύκλο  $c_1$  από την παραμετρική σχέση

$$x(\theta) = \frac{d}{2} + R \cos \theta, \quad y(\theta) = R \sin \theta, \quad -\pi < \theta \leq \pi, \quad (2.32)$$

και τον αριστερό κύκλο  $c_2$  από την παραμετρική σχέση

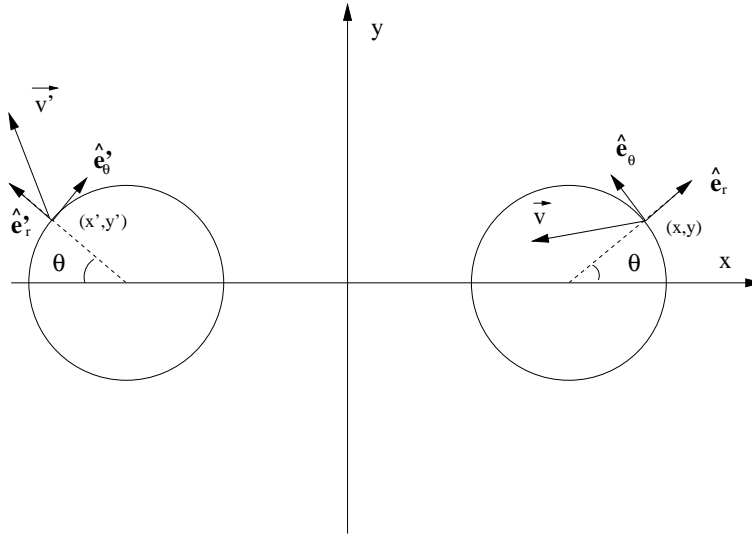
$$x(\theta) = -\frac{d}{2} - R \cos \theta, \quad y(\theta) = R \sin \theta, \quad -\pi < \theta \leq \pi. \quad (2.33)$$

Σε κάθε χρονική στιγμή, προωθούμε το σωματίδιο σύμφωνα με τις σχέσεις

$$\begin{aligned} t_i &= i dt \\ x_i &= x_{i-1} + v_x dt \\ y_i &= y_{i-1} + v_y dt \end{aligned} \quad (2.34)$$

για  $i = 1, 2, \dots$  με δεδομένα  $(x_0, y_0)$ ,  $t_0 = 0$  και όσο  $t_i \leq t_f$ . Αν το σημείο  $(x_i, y_i)$  είναι εκτός των ορίων  $|x| = L/2$ ,  $|y| = L/2$ , το επαναφέρουμε με τις σχέσεις  $x_i \rightarrow x_i \pm L$ ,  $y_i \rightarrow y_i \pm L$  ανάλογα με την περίπτωση. Τα σημεία με το ίδιο  $\theta$  στους δύο κύκλους ταυτίζονται, είναι δηλ. τα ίδια σημεία του χώρου. Αν το σημείο  $(x_i, y_i)$  περάσει μέσα από τον κύκλο  $c_1$  ή  $c_2$ , “βγάζουμε” το σωματίο έξω από τον άλλο κύκλο.





Σχήμα 2.29: Το σωματίδιο περνάει μέσα στο δεξί κύκλο  $c_1$  με ταχύτητα  $\vec{v}$  και αναδύεται από τον  $c_2$  με ταχύτητα  $\vec{v}'$ . Τα ακτινικά / γωνιακά μοναδιαία διανύσματα  $(\hat{e}_r, \hat{e}_\theta)$ ,  $(\hat{e}'_r, \hat{e}'_\theta)$  ταυτίζονται από την παραμετρική ταυτοποίηση των δύο κύκλων  $c_1$  και  $c_2$ .

Το πέρασμα μέσα στον κύκλο  $c_1$  το επισημαίνει η σχέση

$$\left(x_i - \frac{d}{2}\right)^2 + y_i^2 \leq R^2. \quad (2.35)$$

Στην περίπτωση αυτή, υπολογίζουμε τη γωνία  $\theta$  από τη σχέση

$$\theta = \tan^{-1} \left( \frac{y_i}{x_i - \frac{d}{2}} \right), \quad (2.36)$$

και το σημείο  $(x_i, y_i)$  απεικονίζεται στο σημείο  $(x'_i, y'_i)$  όπου

$$x'_i = -\frac{d}{2} - R \cos \theta, \quad y'_i = y_i, \quad (2.37)$$

όπως φαίνεται στο σχήμα 2.29. Για την απεικόνιση της ταχύτητας  $\vec{v} \rightarrow \vec{v}'$  υπολογίζουμε πρώτα τα διανύσματα

$$\left. \begin{aligned} \hat{e}_r &= \cos \theta \hat{x} + \sin \theta \hat{y} \\ \hat{e}_\theta &= -\sin \theta \hat{x} + \cos \theta \hat{y} \end{aligned} \right\} \rightarrow \left\{ \begin{aligned} \hat{e}'_r &= -\cos \theta \hat{x} + \sin \theta \hat{y} \\ \hat{e}'_\theta &= \sin \theta \hat{x} + \cos \theta \hat{y} \end{aligned} \right., \quad (2.38)$$

έτσι ώστε η ταχύτητα

$$\vec{v} = v_r \hat{e}_r + v_\theta \hat{e}_\theta \rightarrow \vec{v}' = -v_r \hat{e}'_r + v_\theta \hat{e}'_\theta, \quad (2.39)$$

όπου οι ακτινικές συνιστώσες  $v_r = \vec{v} \cdot \hat{e}_r$  και  $v_\theta = \vec{v} \cdot \hat{e}_\theta$ . Έτσι, οι τελικές σχέσεις που μας δίνουν την ταχύτητα “ανάδυσης”  $\vec{v}'$  είναι οι:

$$\begin{aligned} v_r &= v_x \cos \theta + v_y \sin \theta \\ v_\theta &= -v_x \sin \theta + v_y \cos \theta \\ v'_x &= v_r \cos \theta + v_\theta \sin \theta \\ v'_y &= -v_r \sin \theta + v_\theta \cos \theta \end{aligned} \quad (2.40)$$

Ανάλογα εργαζόμαστε και για το πέρασμα μέσα στον κύκλο  $c_2$  το οποίο τώρα επισημαίνει η σχέση

$$\left(x_i + \frac{d}{2}\right)^2 + y_i^2 \leq R^2. \quad (2.41)$$

Στην περίπτωση αυτή υπολογίζουμε τη γωνία  $\theta$  από τη σχέση

$$\theta = \pi - \tan^{-1} \left( \frac{y_i}{x_i + \frac{d}{2}} \right), \quad (2.42)$$

και το σημείο  $(x_i, y_i)$  απεικονίζεται στο σημείο  $(x'_i, y'_i)$  όπου

$$x'_i = \frac{d}{2} + R \cos \theta, \quad y'_i = y_i. \quad (2.43)$$

Για την απεικόνιση της ταχύτητας  $\vec{v} \rightarrow \vec{v}'$  υπολογίζουμε τα διανύσματα

$$\left. \begin{aligned} \hat{e}_r &= -\cos \theta \hat{x} + \sin \theta \hat{y} \\ \hat{e}_\theta &= \sin \theta \hat{x} + \cos \theta \hat{y} \end{aligned} \right\} \rightarrow \left\{ \begin{aligned} \hat{e}'_r &= \cos \theta \hat{x} + \sin \theta \hat{y} \\ \hat{e}'_\theta &= -\sin \theta \hat{x} + \cos \theta \hat{y} \end{aligned} \right., \quad (2.44)$$

έτσι ώστε η ταχύτητα

$$\vec{v} = v_r \hat{e}_r + v_\theta \hat{e}_\theta \rightarrow \vec{v}' = -v_r \hat{e}'_r + v_\theta \hat{e}'_\theta. \quad (2.45)$$

Τώρα οι τελικές σχέσεις που μας δίνουν την ταχύτητα “ανάδυσης”  $\vec{v}'$  είναι οι:

$$\begin{aligned} v_r &= -v_x \cos \theta + v_y \sin \theta \\ v_\theta &= v_x \sin \theta + v_y \cos \theta \\ v'_x &= -v_r \cos \theta - v_\theta \sin \theta \\ v'_y &= -v_r \sin \theta + v_\theta \cos \theta \end{aligned} \quad (2.46)$$

Τα συστηματικά σφάλματα προέρχονται μόνο από τα περάσματα μέσα στη σκουλικότρυπα. Δεν υπάρχουν συστηματικά σφάλματα στο πέρασμα των ορίων  $|x| = L/2$ ,  $|y| = L/2$  (γιατί;). Σκεφτείτε τρόπους να τα αντιμετωπίσετε στο πρόγραμμά σας, καθώς και να τα μελετήσετε.

Οι κλειστές τροχιές που ζητούνται προκύπτουν, λ.χ., από τις αρχικές συνθήκες

$$(x_0, y_0, v_0, \phi) = (0, 0, 1, 0) \quad (2.47)$$

που ενώνει τα σημεία 1 του σχήματος 2.28. Είναι ασταθής, και αυτό μπορείτε να το δείτε παίρνοντας  $\phi \rightarrow \phi + \epsilon$ .

Οι κλειστές τροχιές που περνούν τη σκουλικότρυπα, αλλά “περιτυλίγονται” προκύπτουν λ.χ. από τις αρχικές συνθήκες

$$\begin{aligned} (x_0, y_0, v_0, \phi) &= (-9, 0, 1, 0) \\ (x_0, y_0, v_0, \phi) &= (2.5, -3, 1, 90^\circ) \end{aligned}$$

που περνούν από  $3 \rightarrow 3$  και  $2 \rightarrow 2 \rightarrow 4 \rightarrow 4$  αντίστοιχα. Είναι επίσης ασταθείς όπως εύκολα μπορεί να μελετηθεί με το πρόγραμμα που θα γράψετε. Το πρόγραμμα αυτό παρατίθεται παρακάτω για πληρότητα:

```
!=====
program WormHole2D
  implicit none
!
!Declaration of variables
  real(8), parameter :: PI=3.14159265358979324D0
  real(8) :: Lx,Ly,L,R,d
  real(8) :: x0,y0,v0,theta
  real(8) :: t0,tf,dt
  real(8) :: t,x,y,vx,vy
  real(8) :: xc1,yc1,xc2,yc2,r1,r2
  integer :: i
!
!Ask user for input:
  print *, '# Enter L,d,R: '
  read *, L,d,R
  print *, '# L= ',L, ' d= ',d, ' R= ',R
  if( L .le. d+2.0D0*R) stop 'L <= d+2*R'
  if( d .le. 2.0D0*R) stop 'd <= 2*R'
  print *, '# Enter (x0,y0), v0, theta(degrees): '
  read *, x0,y0,v0,theta
  print *, '# x0= ',x0, ' y0 = ',y0
  print *, '# v0= ',v0, ' theta= ',theta, ' degrees'
  if(v0 .le. 0.0D0 ) stop 'illegal value of v0.'
  print *, '# Enter tf, dt: '
  read *, tf,dt
  print *, '# tf= ',tf, ' dt= ',dt
!
!Initialize
  theta = (PI/180.0D0)*theta
```

```

i      = 0
t      = 0.0D0
x      = x0          ; y      = y0
vx     = v0*cos(theta); vy    = v0*sin(theta)
print *, '# x0= ',x, ' y0= ',y, ' v0x= ',vx, ' v0y= ',vy
!Wormhole's centers:
xc1    = 0.5D0*d; yc1    = 0.0D0
xc2    = -0.5D0*d; yc2   = 0.0D0
!Box limits coordinates:
Lx     = 0.5D0*L; Ly     = 0.5D0*L
!Test if already inside cut region:
r1     = sqrt((x-xc1)**2+(y-yc1)**2)
r2     = sqrt((x-xc2)**2+(y-yc2)**2)
if( r1 .le. R ) stop 'r1 <= R'
if( r2 .le. R ) stop 'r2 <= R'
!Test if outside box limits:
if(ABS(x) .ge. Lx) stop '|x| >= Lx'
if(ABS(y) .ge. Ly) stop '|y| >= Ly'
open(unit=11,file='Wormhole.dat')
!
!Compute:
do while( t .lt. tf )
  write(11,*)t,x,y,vx,vy
  i = i+1
  t = i*dt
  x = x + vx*dt; y = y + vy*dt
! Toroidal boundary conditions:
  if( x .gt. Lx) x = x - L
  if( x .lt. -Lx) x = x + L
  if( y .gt. Ly) y = y - L
  if( y .lt. -Ly) y = y + L
! Test if inside the cut disks
  r1 = sqrt((x-xc1)**2+(y-yc1)**2)
  r2 = sqrt((x-xc2)**2+(y-yc2)**2)
  if( r1 .lt. R)then
! Notice: we pass r1 as radius of circle , not R
    call crossC1(x,y,vx,vy,dt,r1,d)
  else if( r2 .lt. R)then
    call crossC2(x,y,vx,vy,dt,r2,d)
  endif
! small chance here that still in C1 or C2, but OK since
! another dt-advance given at the beginning of do-loop
enddo !do while( t .lt. tf )
end program WormHole2D
!=====
subroutine crossC1(x,y,vx,vy,dt,R,d)
  implicit none
  real(8) :: x,y,vx,vy,dt,R,d
  real(8) :: vr,v0 !v0 -> vtheta

```

```

real(8) :: theta,xc,yc
print *,'# Inside C1: (x,y,vx,vy,R)= ',x,y,vx,vy,R
xc      = 0.5D0*d !center of C1
yc      = 0.0D0
theta   = atan2(y-yc,x-xc)
x       = -xc - R*cos(theta) !new x-value, y invariant
!Velocity transformation:
vr      = vx*cos(theta)+vy*sin(theta)
v0      = -vx*sin(theta)+vy*cos(theta)
vx      = vr*cos(theta)+v0*sin(theta)
vy      = -vr*sin(theta)+v0*cos(theta)
!advance x,y, hopefully outside C2:
x       = x + vx*dt
y       = y + vy*dt
print *,'# Exit C2: (x,y,vx,vy )= ',x,y,vx,vy
end subroutine crossC1
=====
subroutine crossC2(x,y,vx,vy,dt,R,d)
implicit none
real(8), parameter :: PI=3.14159265358979324D0
real(8) :: x,y,vx,vy,dt,R,d
real(8) :: vr,v0 !v0 -> vtheta
real(8) :: theta,xc,yc

print *,'# Inside C2: (x,y,vx,vy,R)= ',x,y,vx,vy,R
xc      = -0.5D0*d !center of C2
yc      = 0.0D0
theta   = PI-atan2(y-yc,x-xc)
x       = -xc + R*cos(theta) !new x-value, y invariant
!Velocity transformation:
vr      = -vx*cos(theta)+vy*sin(theta)
v0      = vx*sin(theta)+vy*cos(theta)
vx      = -vr*cos(theta)-v0*sin(theta)
vy      = -vr*sin(theta)+v0*cos(theta)
!advance x,y, hopefully outside C1:
x       = x + vx*dt
y       = y + vy*dt
print *,'# Exit C1: (x,y,vx,vy )= ',x,y,vx,vy
end subroutine crossC2

```

Το πρόγραμμα μεταγλωττίζεται και τρέχει κατά τα γνωστά. Στο συνοδευτικό λογισμικό θα βρείτε τα αρχεία `Wormhole.csh` και `Wormhole_animate.gnu` που μπορούν να σας βοηθήσουν στην απεικόνιση της τροχιάς. Ξεκινήστε το `gnuplot`, θέστε τις επιθυμητές παραμέτρους στο `Wormhole.csh` και δώστε τις εντολές

```
gnuplot> file = "Wormhole.dat"
```

```
gnuplot> R=1;d=5;L=20;  
gnuplot> ! ./Wormhole.csh  
gnuplot> t0=0;dt=0.2;load "Wormhole_animate.gnu"
```

Μπορείτε τώρα να απαντήσετε και τις υπόλοιπες ερωτήσεις που θέσαμε παραπάνω στους στόχους μας...

## 2.5 Ασκήσεις

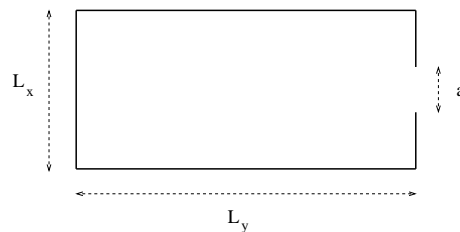
- 2.1 Στο πρόγραμμα `Circle.f90` δώστε τις εντολές που απαιτούνται, ώστε το πρόγραμμα να τυπώνει τον αριθμό των πλήρων κύκλων που διέγραψε το κινητό.
- 2.2 Στο πρόγραμμα `Circle.f90` προσθέστε όλα τα αναγκαία τεστ στις παραμέτρους που εισάγει ο χρήστης, έτσι ώστε το πρόγραμμα να είναι εγγυημένο ότι θα τρέξει ομαλά. Κάνετε το ίδιο και στα άλλα προγράμματα που δίνονται στο κεφάλαιο αυτό.
- 2.3 Ύλικο σημείο κινείται πάνω σε κύκλο με κέντρο την αρχή των αξόνων με σταθερή γωνιακή ταχύτητα  $\omega$ . Τη χρονική στιγμή  $t_0 = 0$  το σωματίο βρίσκεται στο σημείο  $(x_0, y_0)$ . Γράψτε το πρόγραμμα `CircularMotion.f90` που να απεικονίζει την τροχιά. Ο χρήστης θα δίνει στην είσοδο τα  $\omega, x_0, y_0, t_0, t_f, \delta t$ . Στην έξοδο θα παίρνει τα δεδομένα σύμφωνα με το πρόγραμμα `Circle.f90`.
- 2.4 Μετατρέψτε το πρόγραμμα `SimplePendulum.f90`, έτσι ώστε ο χρήστης να μπορεί να δίνει και μη μηδενική ταχύτητα ως αρχική συνθήκη.
- 2.5 Μελετήστε το όριο  $k \rightarrow 0$  στο πρόβλημα της βολής που δίνεται από τις Σχέσεις (2.10). Αναπτύξτε το  $e^{-kt} = 1 - kt + \frac{1}{2!}(kt)^2 + \dots$  και κρατήστε τους όρους που δε μηδενίζονται στο όριο  $k \rightarrow 0$ . Στη συνέχεια, κρατήστε τους όρους με την αμέσως μικρότερη δύναμη του  $k$ . Προγραμματίστε τις εξισώσεις αυτές σε ένα αρχείο `ProjectileSmallAirResistance.f90`. Θεωρήστε τις αρχικές συνθήκες  $\vec{v}_0 = \hat{x} + \hat{y}$  και υπολογίστε αριθμητικά το βεληνεκές της τροχιάς με τα δύο προγράμματα `ProjectileSmallAirResistance.f90`, `ProjectileAirResistance.f90`. Προσδιορίστε τις τιμές του  $k$  για τις οποίες τα αποτελέσματά σας συμφωνούν με ακρίβεια καλύτερη από 5%.
- 2.6 Προγραμματίστε την κίνηση της βολής, όταν η δύναμη της αντίστασης του αέρα έχει μέτρο ανάλογο του τετραγώνου της ταχύτητας. Συγκρίνετε το βεληνεκές της τροχιάς που προκύπτει με αυτό που υπολογίζεται από το πρόγραμμα `ProjectileAirResistance.f90` για τις παραμέτρους του σχήματος 2.10.
- 2.7 Κάνετε τις απαραίτητες μετατροπές στο πρόγραμμα `Lissajous.f90`, ώστε ο χρήστης να μπορεί να δίνει διαφορετικό πλάτος και αρχική

φάση στην κίνηση που γίνεται στη διεύθυνση του άξονα των  $y$ . Μελετήστε τις τροχιές όταν το πλάτος είναι ίσο και η διαφορά φάσης είναι  $\pi/4, \pi/2, \pi, -\pi$  και το ίδιο όταν το πλάτος στη διεύθυνση του άξονα των  $y$  είναι διπλάσιο.

- 2.8 Κάνετε τις απαραίτητες μετατροπές στο πρόγραμμα `ProjectileAirResistance.f90`, ώστε να μπορεί να συμπεριλάβει και τη μελέτη της περίπτωσης  $k = 0$ .
- 2.9 Κάνετε τις απαραίτητες μετατροπές στο πρόγραμμα `ProjectileAirResistance.f90`, ώστε να υπολογίστε την κίνηση του υλικού σημείου στο χώρο. Κάνετε τα διαγράμματα της θέσης/ταχύτητας συναρτήσει του χρόνου και της τρισδιάστατης τροχιάς με την εντολή `splot` στο `gnuplot`. Χρησιμοποιήστε το πρόγραμμα `animate3D.gnu` για να κάνετε animation της τροχιάς.
- 2.10 Κάνετε τις απαραίτητες μετατροπές στο πρόγραμμα `ChargeInB.f90`, ώστε να υπολογίζει τον αριθμό των πλήρων στροφών που έχει κάνει η προβολή της τροχιάς του φορτίου στο επίπεδο  $x - y$ .
- 2.11 Μεταβάλετε το πρόγραμμα `box1D_1.f90`, έτσι ώστε να τυπώνει στο `stdout` τον αριθμό των κρούσεων του υλικού σημείου στο αριστερό τοίχωμα, στο δεξί και το συνολικό.
- 2.12 Επαναλάβετε το ίδιο για το πρόγραμμα `box1D_2.f90`. Συμπληρώστε σε δύο στήλες στον πίνακα της Σελ. 124 το συνολικό αριθμό των κρούσεων που προκύπτουν και σχολιάστε.
- 2.13 Στο πρόγραμμα `box1D_1.f90` επιλέξτε  $L = 10$ ,  $v_0 = 1$ . Ελαττώστε το  $\Delta t$  μέχρι το υλικό σημείο να σταματήσει να κινείται. Για ποια τιμή του  $\Delta t$  συμβαίνει αυτό; Αυξήστε το  $v_0 = 10, 100$ . Μέχρι ποια τιμή του  $\Delta t$  μπορείτε να φτάσετε τώρα; Γιατί;
- 2.14 Στο πρόγραμμα `box1D_1.f90` αλλάξτε τις δηλώσεις `REAL`  $\rightarrow$  `REAL(8)` και στις σταθερές προσθέστε τον εκθέτη `D0` (λ.χ.  $0.0 \rightarrow 0.0D0$ ). Μελετήστε τη διαφορά στα αποτελέσματα που πήρατε στην ανάλυση της παραγράφου 2.3.2. Επαναλάβετε την άσκηση 2.13. Τι παρατηρείτε;
- 2.15 Μεταβάλετε το πρόγραμμα `box1D_1.f90`, ώστε να μελετήσετε την περίπτωση που το κινητό συγκρούεται μη ελαστικά με τα τοιχώματα του κουτιού, έτσι ώστε  $v' = -ev$ ,  $0 < e \leq 1$ .

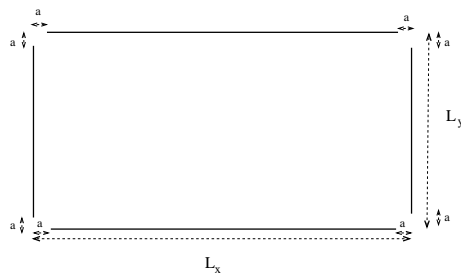


- 2.16 Μεταβάλλετε το πρόγραμμα `box2D_1.f90`, ώστε να μελετήσετε την περίπτωση που το κινητό συγκρούεται μη ελαστικά με τα τοιχώματα του κουτιού, έτσι ώστε  $v'_x = -ev_x$ ,  $v'_y = -ev_y$ ,  $0 < e \leq 1$ .
- 2.17 Χρησιμοποιείτε τη μέθοδο υπολογισμού του χρόνου που βρίσκεται στα προγράμματα `box1D_4.f90` και `box1D_5.f90` και αναπαράγετε τα αποτελέσματα του σχήματος 2.21.
- 2.18 Κινητό πέφτει ελεύθερα στην κατακόρυφη διεύθυνση ξεκινώντας από ηρεμία από ύψος  $h$ . Στο δάπεδο υφίσταται μη ελαστική κρούση, έτσι ώστε μετά την κρούση  $v'_y = -ev_y$  με  $0 < e \leq 1$  παράμετρος. Μελετήστε γράφοντας κατάλληλο πρόγραμμα την κίνηση για  $e = 0.1, 0.5, 0.9, 1.0$ .
- 2.19 Γενικεύστε το προηγούμενο πρόγραμμα για  $\vec{v}_0 = v_{0x} \hat{x}$ . Κάνετε το animation της τροχιάς.
- 2.20 Μελετήστε την κίνηση υλικού σημείου στο κουτί του σχήματος 2.30. Μετρήστε σε κάθε βολή πόσες κρούσεις παίρνετε μέχρι το υλικό σημείο να βγει από το κουτί.

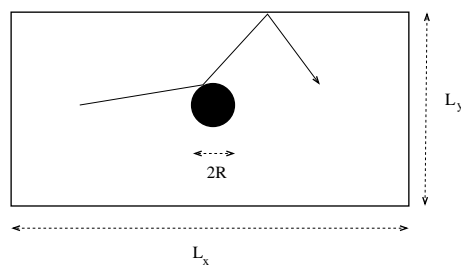


Σχήμα 2.30: Άσκηση 2.20.

- 2.21 Μελετήστε την κίνηση υλικού σημείου στο “μπιλιάρδο” του σχήματος 2.31. Μετρήστε σε κάθε βολή πόσες κρούσεις έχετε μέχρι να πετύχετε μία “τρύπα”. Το πρόγραμμα να σημειώνει σε ποια τρύπα μπήκε η μπίλια.
- 2.22 Προγραμματίστε την κίνηση ενός υλικού σημείου μέσα στο επίπεδο κουτί του σχήματος 2.32. Στο κέντρο του υπάρχει κύκλος πάνω στον οποίο το υλικό σημείο σκεδάζεται ελαστικά (Υποδ.: Χρησιμοποιείτε την υπορουτίνα `reflectVonCircle` του προγράμματος `Cylinder3D.f90`).

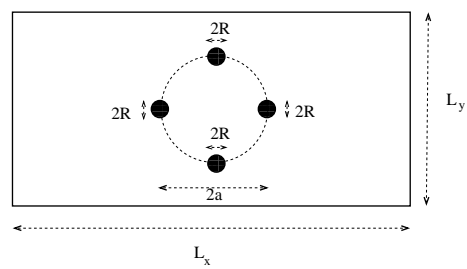


Σχήμα 2.31: Άσκηση 2.21.

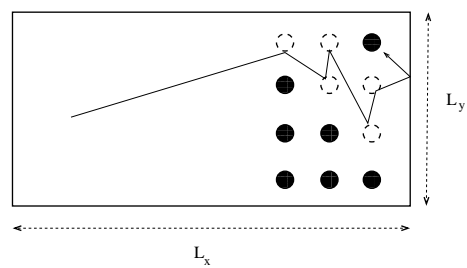


Σχήμα 2.32: Άσκηση 2.22.

- 2.23 Τοποθετήστε στο κουτί της προηγούμενης άσκησης 4 κύκλους πάνω στους οποίους θα ανακλάται το υλικό σημείο (σχήμα 2.33).
- 2.24 Θεωρήστε τη διάταξη του σχήματος 2.34. Κάθε φορά που το υλικό σημείο σκεδάζεται σε ένα κύκλο, ο κύκλος “εξαφανίζεται”. Η κίνηση σταματάει, όταν όλοι οι κύκλοι εξαφανιστούν. Κάθε φορά που το υλικό σημείο χτυπάει τον αριστερό τοίχο, χάνετε ένα πόντο. Προσπαθήστε να διαλέξετε τροχιές που ελαχιστοποιούν τους χαμένους πόντους.



Σχήμα 2.33: Άσκηση 2.23.



Σχήμα 2.34: Άσκηση 2.24.



## ΚΕΦΑΛΑΙΟ 3

### Η Λογιστική Απεικόνιση

Οι μη γραμμικές διαφορικές εξισώσεις περιγράφουν ενδιαφέροντα δυναμικά συστήματα στη φυσική, τη βιολογία και τις οικονομικές επιστήμες. Στο κεφάλαιο αυτό μελετάμε αριθμητικά τη διακριτή λογιστική απεικόνιση ως “ένα απλό μαθηματικό πρότυπο με πολύπλοκες δυναμικές ιδιότητες” [20] οι οποίες συναντώνται σε πολλά από τα προαναφερόμενα συστήματα. Η εξίσωση αυτή παρουσιάζει χαοτική συμπεριφορά για κάποιες τιμές της δυναμικής παραμέτρου της και έτσι θα μας δοθεί η ευκαιρία να αγγίξουμε αυτό το πολύ ενδιαφέρον θέμα με σημαντικές επιπτώσεις στα φυσικά φαινόμενα. Το φαινόμενο αυτό περιορίζει σημαντικά τις δυνατότητες μας για χρήσιμες προβλέψεις στη χρονική εξέλιξη ενός, κατά τ’ άλλα, ντετερμινιστικού δυναμικού συστήματος: τα αποτελέσματα μετρήσεων με διαφορετικές αρχικές συνθήκες ακολουθούν κατανομή που δεν μπορεί να διαχωριστεί από αυτή μιας δειγματοληπτικής συνάρτησης μιας τυχαίας διεργασίας. Το πεδίο αυτό της μελέτης μη γραμμικών δυναμικών συστημάτων είναι τεράστιο και παραπέμπουμε τον αναγνώστη στη βιβλιογραφία για μια πληρέστερη μελέτη [20, 21, 22, 23, 24, 25, 26, 37].

#### 3.1 Εισαγωγή

Η πιο γνωστή εφαρμογή της διακριτής λογιστικής εξίσωσης προέρχεται από τη μελέτη βιολογικών πληθυσμών οι οποίοι αναπαράγονται ανά τακτά χρονικά διαστήματα. Τα απλά πρότυπα προϋποθέτουν οι γενεές να μην επικαλύπτονται.

Το πιο απλό (και αφελές) πρότυπο είναι αυτό που κάνει την εύλογη υπόθεση ότι ο ρυθμός αύξησης  $dP(t)/dt$  ενός πληθυσμού  $P(t)$  είναι

ανάλογος του ήδη υπάρχοντος πληθυσμού:

$$\frac{dP(t)}{dt} = kP(t). \quad (3.1)$$

Η γενική λύση της παραπάνω εξίσωσης είναι  $P(t) = P(0)e^{kt}$  και υποδηλώνει εκθετική αύξηση του πληθυσμού για  $k > 0$  και μείωση για  $k < 0$ . Προφανώς αυτό το μοντέλο μπορεί να ισχύει όσο ο πληθυσμός είναι πολύ μικρός και μπορεί να αγνοηθεί η αλληλεπίδραση με το περιβάλλον (επάρκεια τροφής, ασθένειες, θηρευτές κλπ), αλλά είναι φανερό ότι δεν μπορεί να ισχύει για αρκετά μεγάλους χρόνους. Ο πιο απλός τρόπος για να λάβουμε υπόψη τους παράγοντες αλληλεπίδρασης του πληθυσμού με το περιβάλλον είναι να εισάγουμε έναν μη γραμμικό όρο στην εξίσωση, έτσι ώστε

$$\frac{dP(t)}{dt} = kP(t)(1 - bP(t)). \quad (3.2)$$

Η παράμετρος  $k$  δίνει το μέγιστο ρυθμό ανάπτυξης του πληθυσμού, ενώ το  $b$  ελέγχει τη δυνατότητα του είδους να συντηρήσει ένα συγκεκριμένο επίπεδο πληθυσμού. Η εξίσωση (3.1) μπορεί να διακριτοποιηθεί στο χρόνο υποθέτοντας ότι κάθε γενεά αναπαράγεται, κάθε  $\delta t$  και η  $n$ -οστή γενιά έχει πληθυσμό  $P_n = P(t_n)$  όπου  $t_n = t_0 + (n - 1)\delta t$ . Τότε  $P(t_{n+1}) \approx P(t_n) + \delta t P'(t_n)$  και η εξίσωση (3.1) γίνεται

$$P_{n+1} = rP_n, \quad (3.3)$$

όπου  $r = 1 + k\delta t$ . Οι λύσεις της παραπάνω εξίσωσης θα πρέπει να προσεγγίζουν τις  $P_n \sim P_0 e^{kt_n} \propto e^{(r-1)n}$  και έχουμε εκθετική αύξηση πληθυσμού, όταν  $r > 1$  και μείωση, όταν  $r < 1$ . Η εξίσωση (3.2) μπορεί τότε να διακριτοποιηθεί ως εξής:

$$P_{n+1} = P_n(r - bP_n). \quad (3.4)$$

Με τον ορισμό  $x_n = (b/r)P_n$  παίρνουμε τη λογιστική απεικόνιση

$$x_{n+1} = rx_n(1 - x_n). \quad (3.5)$$

Ορίζουμε τις συναρτήσεις

$$f(x) = rx(1 - x), \quad F(x, r) = rx(1 - x) \quad (3.6)$$

(η μόνη διαφορά τους είναι ότι στην πρώτη το  $r$  θεωρείται δεδομένη παράμετρος), έτσι ώστε

$$x_{n+1} = f(x_n) = f^{(2)}(x_{n-1}) = \dots = f^{(n)}(x_1) = f^{(n+1)}(x_0), \quad (3.7)$$

όπου χρησιμοποιήσαμε το συμβολισμό  $f^{(1)}(x) = f(x)$ ,  $f^{(2)}(x) = f(f(x))$ ,  $f^{(3)}(x) = f(f(f(x)))$ , ... για τη σύνθεση συναρτήσεων. Η παράγωγος της συνάρτησης  $f$  θα μας χρησιμεύσει αρκετά στα παρακάτω:

$$f'(x) = \frac{\partial F(x, r)}{\partial x} = r(1 - 2x). \quad (3.8)$$

Επειδή ερμηνεύουμε την ποσότητα  $x_n$  ως ποσοστό επί του μέγιστου δυνατού πληθυσμού, θα πρέπει  $0 \leq x_n \leq 1$  για κάθε<sup>1</sup>  $n$ . Η συνάρτηση  $f(x)$  παρουσιάζει μέγιστο για  $x = 1/2$  το οποίο είναι ίσο με  $f(1/2) = r/4$ . Άρα, αν  $r > 4$ , τότε  $f(1/2) > 1$  το οποίο με κατάλληλη επιλογή του  $x_0$  θα οδηγήσει σε  $x_{n+1} = f(x_n) > 1$  για κάποιο  $n$ . Οπότε το ενδιαφέρον εύρος των δυνατών τιμών της παραμέτρου  $r$  είναι

$$0 < r \leq 4. \quad (3.9)$$

Η λογιστική απεικόνιση (3.5) μπορεί να θεωρηθεί σαν εξίσωση πεπερασμένων διαφορών, αλλά η τελική της έκφραση είναι μια επαγωγική σχέση ενός βήματος. Έτσι, δεδομένης μιας αρχικής τιμής  $x_0$ , μέσω αυτής παράγεται μια ακολουθία τιμών  $\{x_0, x_1, \dots, x_n, \dots\}$  στην οποία θα αναφερόμαστε<sup>2</sup> ως την τροχιά του  $x_0$ . Στις επόμενες παραγράφους του κεφαλαίου αυτού, θα επικεντρωθούμε στη μελέτη των ιδιοτήτων των τροχιών αυτών ως συνάρτηση της παραμέτρου  $r$ .

Οι λύσεις της λογιστικής απεικόνισης δεν είναι γνωστές παρά μόνο για  $r = 2$  και  $r = 4$ . Για  $r = 2$  έχουμε

$$x_n = \frac{1}{2} (1 - (1 - x_0)^{2^n}), \quad (3.10)$$

ενώ<sup>3</sup> για  $r = 4$

$$x_n = \sin^2(2^n \pi \theta), \quad \theta = \frac{1}{\pi} \sin^{-1} \sqrt{x_0}. \quad (3.11)$$

Για  $r = 2$ ,  $\lim_{n \rightarrow \infty} x_n = 1/2$ , ενώ για  $r = 4$  έχουμε περιοδικές τροχιές για  $x_0$  που δίνουν ρητό  $\theta$  και μη περιοδικές, όταν δίνουν άρρητο  $\theta$ . Για τις υπόλοιπες τιμές του  $r$  θα εργαστούμε αριθμητικά για να μελετήσουμε τις ιδιότητες των τροχιών της λογιστικής εξίσωσης.

<sup>1</sup>Παρατηρήστε ότι αν  $x_n > 1$ , τότε  $x_{n+1} < 0$ , οπότε αν θέλουμε  $x_n \geq 0$  για κάθε  $n$ , τότε θα πρέπει  $x_n \leq 1$  για κάθε  $n$ .

<sup>2</sup>Στη μαθηματική βιβλιογραφία αναφέρεται και ως “splinter of  $x_0$ ” (splinter=ομάδα που δημιουργείται από το διαχωρισμό μιας μεγαλύτερης ομάδας).

<sup>3</sup>E. Schröder, “Über iterierte Funktionen”, Math. Ann. **3** (1870) 296; E. Lorenz, “The problem of deducing the climate from the governing equations”, Tellus **16** (1964) 1

### 3.2 Σταθερά Σημεία και $2^n$ Κύκλοι

Είναι φανερό ότι αν το σημείο  $x^*$  είναι μία λύση της εξίσωσης  $x = f(x)$ , τότε  $x_n = x^* \Rightarrow x_{n+k} = x^*$  για κάθε  $k \geq 0$ . Για τη συγκεκριμένη συνάρτηση  $f(x) = rx(1-x)$  έχουμε δύο λύσεις

$$x_1^* = 0 \quad \text{και} \quad x_2^* = 1 - 1/r. \quad (3.12)$$

Θα δούμε ότι τα σημεία αυτά είναι ελκυστές των τροχιών για κατάλληλες τιμές του  $r$ . Αυτό σημαίνει ότι για ένα εύρος τιμών της αρχικής τιμής  $0 \leq x_0 \leq 1$ , η ακολουθία  $\{x_n\}$  πλησιάζει ασυμπτωτικά κάποιο από αυτά τα σημεία καθώς  $n \rightarrow \infty$ . Προφανώς τα (μέτρου μηδέν) σύνολα αρχικών τιμών  $\{x_0\} = \{x_1^*\}$  και  $\{x_0\} = \{x_2^*\}$  δίνουν τροχιές που ελκύονται από τα  $x_1^*$  και  $x_2^*$  αντίστοιχα. Για να δούμε ποιες από τις δύο τιμές “προτιμούν” οι αρχικές συνθήκες στο διάστημα  $(0, 1)$ , αρκεί να μελετήσουμε την ευστάθεια των δύο σταθερών σημείων  $x_1^*$  και  $x_2^*$ . Ας υποθέσουμε ότι για κάποιο  $n$  η τιμή  $x_n$  είναι απειροστά κοντά στο σταθερό σημείο  $x^*$ , έτσι ώστε

$$\begin{aligned} x_n &= x^* + \epsilon_n \\ x_{n+1} &= x^* + \epsilon_{n+1}. \end{aligned} \quad (3.13)$$

Τότε, επειδή

$$x_{n+1} = f(x_n) = f(x^* + \epsilon_n) \approx f(x^*) + \epsilon_n f'(x^*) = x^* + \epsilon_n f'(x^*), \quad (3.14)$$

όπου χρησιμοποιήσαμε το ανάπτυγμα κατά Taylor της  $f(x^* + \epsilon_n)$  γύρω από το  $x^*$ , καθώς και τη σχέση  $x^* = f(x^*)$ , έχουμε ότι  $\epsilon_{n+1} = \epsilon_n f'(x^*)$ . Οπότε ισχύει ότι

$$\left| \frac{\epsilon_{n+1}}{\epsilon_n} \right| = |f'(x^*)|. \quad (3.15)$$

Άρα, αν  $|f'(x^*)| < 1$ , τότε  $\lim_{n \rightarrow \infty} \epsilon_n = 0$  και το σταθερό σημείο  $x^*$  είναι ευσταθές: η ακολουθία  $\{x_{n+k}\}$  πλησιάζει ασυμπτωτικά κοντά στο  $x^*$ . Αν  $|f'(x^*)| > 1$ , τότε η ακολουθία  $\{x_{n+k}\}$  απομακρύνεται από το  $x^*$  και το σταθερό σημείο είναι ασταθές. Η περίπτωση  $|f'(x^*)| = 1$  είναι οριακή και καταδεικνύει αλλαγή συμπεριφοράς. Πρέπει να μελετηθεί κατά περίπτωση και συνδέεται με την εμφάνιση σημείων διακλάδωσης που θα συζητήσουμε παρακάτω.

Για την ειδική περίπτωση της  $f(x) = rx(1-x)$  με  $f'(x) = r(1-2x)$  έχουμε ότι  $f'(0) = r$  και  $f'(1-1/r) = 2-r$ . Άρα, όταν  $r < 1$ , το σημείο  $x_1^* = 0$  είναι ελκυστής, ενώ το  $x_2^* = 1 - 1/r < 0$  και δεν εμφανίζεται. Όταν  $r > 1$ , το  $x_1^* = 0$  δίνει  $|f'(x_1^*)| = r > 1$ , οπότε το  $x_1^*$  είναι



ασταθές. Οποιαδήποτε αρχική συνθήκη  $x_0$  κοντά στο  $x_1^*$  απομακρύνεται από αυτό. Επειδή για  $1 < r < 3$  έχουμε  $0 \leq |f'(x_2^*)| = |2 - r| < 1$ , το  $x_2^*$  είναι ελκυστής. Οποιαδήποτε αρχική συνθήκη  $x_0 \in (0, 1)$  πλησιάζει ασυμπτωτικά κοντά στην τιμή  $x_2^* = 1 - 1/r$ . Όταν  $r = r_c^{(1)} = 1$ , έχουμε την οριακή κατάσταση  $x_1^* = x_2^* = 0$  και λέμε ότι στην κρίσιμη τιμή  $r_c^{(1)} = 1$  το κρίσιμο σημείο  $x_1^*$  διακλαδώνεται (bifurcates) στα δύο σταθερά σημεία  $x_1^*$  και  $x_2^*$ .

Οι διακλαδώσεις αυτές συνεχίζονται, καθώς το  $r$  αυξάνει. Πράγματι, όταν  $r = r_c^{(2)} = 3$ , έχουμε  $f'(x_2^*) = 2 - r = -1$  και για  $r > r_c^{(2)}$  το  $x_2^*$  γίνεται ασταθές σταθερό σημείο. Θεωρήστε τη λύση της εξίσωσης  $x = f^{(2)}(x)$ . Αν  $0 < x^* < 1$  είναι μία λύση της και για κάποιο  $n$  έχουμε  $x_n = x^*$ , τότε  $x_{n+2} = x_{n+4} = \dots = x_{n+2k} = \dots = x^*$ , καθώς και  $x_{n+1} = x_{n+3} = \dots = x_{n+2k+1} = \dots = f(x^*)$  (άρα και το  $f(x^*)$  είναι λύση της ίδιας εξίσωσης). Αν  $0 < x_3^* < x_4^* < 1$  είναι δύο τέτοιες διαφορετικές λύσεις με  $x_3^* = f(x_4^*)$ ,  $x_4^* = f(x_3^*)$ , τότε το σύστημα εκτελεί μια περιοδική τροχιά με περίοδο 2. Τα  $x_3^*, x_4^*$  είναι τέτοια, ώστε να είναι πραγματικές λύσεις της εξίσωσης

$$f^{(2)}(x) = r^2 x(1-x)(1-rx(1-x)) = x, \quad (3.16)$$

ενώ ταυτόχρονα θα πρέπει να μην είναι οι λύσεις  $x_1^*, x_2^*$  της εξίσωσης  $x = f(x)$ . Επειδή τα σημεία  $x_1^* = 0, x_2^* = 1 - 1/r$  είναι λύσεις<sup>4</sup> και της  $x = f^{(2)}(x)$ , το παραπάνω πολυώνυμο μπορεί να γραφτεί στη μορφή (δείτε [21] για περισσότερες λεπτομέρειες)

$$x \left( x - \left( 1 - \frac{1}{r} \right) \right) (Ax^2 + Bx + C) = 0, \quad (3.17)$$

οπότε αναπτύσσοντας τα πολυώνυμα (3.16), (3.17) και συγκρίνοντας τους συντελεστές τους συμπεραίνουμε ότι  $A = -r^3$ ,  $B = r^2(r+1)$  και  $C = -r(r+1)$ . Οι ρίζες του διωνύμου στην (3.17) καθορίζονται από τη διακρίνουσα  $\Delta = r^2(r+1)(r-3)$ . Για τις τιμές του  $r$  που μας ενδιαφέρουν ( $1 < r \leq 4$ ) η διακρίνουσα γίνεται θετική, όταν  $r > r_c^{(2)} = 3$  και τότε έχουμε δύο διακριτές λύσεις

$$x_\alpha^* = ((r+1) \mp \sqrt{r^2 - 2r - 3})/(2r) \quad \alpha = 3, 4. \quad (3.18)$$

Όταν  $r = r_c^{(2)}$  έχουμε μια διπλή ρίζα, άρα ένα μοναδικό σταθερό σημείο όπως είδαμε και παραπάνω.

Για τη μελέτη της ευστάθειας των λύσεων της  $x = f^{(2)}(x)$  ακολουθούμε την ίδια πορεία που μας οδήγησε στην εξίσωση (3.15) και

<sup>4</sup>Επειδή αν  $x^* = f(x^*) \Rightarrow f^{(2)}(x^*) = f(f(x^*)) = f(x^*) = x^*$  κ.ο.κ., το σημείο  $x^*$  είναι και λύση της  $x^* = f^{(n)}(x^*)$ .

εξετάζουμε αν η  $f^{(2)'}(x)$  έχει απόλυτη τιμή μικρότερη, μεγαλύτερη ή ίση της μονάδας. Παρατηρώντας ότι<sup>5</sup>  $f^{(2)'}(x_3) = f^{(2)'}(x_4) = f'(x_3)f'(x_4) = -r^2 + 2r + 4$ , βλέπουμε ότι για  $r = r_c^{(2)} = 3$ ,  $f^{(2)'}(x_3^*) = f^{(2)'}(x_4^*) = 1$  και για  $r = r_c^{(3)} = 1 + \sqrt{6} \approx 3.4495$ ,  $f^{(2)'}(x_3) = f^{(2)'}(x_4) = -1$ . Στις ενδιάμεσες τιμές  $3 < r < 1 + \sqrt{6}$  οι παράγωγοι  $|f^{(2)'}(x_\alpha^*)| < 1$  για  $\alpha = 3, 4$ . Άρα τα σημεία αυτά είναι ευσταθείς λύσεις της  $x = f^{(2)}(x)$  και έχουμε διακλάδωση των σταθερών σημείων  $x_1^*, x_2^*$  στα  $x_\alpha^*$ ,  $\alpha = 1, 2, 3, 4$  για  $r = r_c^{(2)} = 3$ . Σχεδόν όλες οι τροχιές με αρχικές συνθήκες στο διάστημα  $[0, 1]$  ελκύνονται ασυμπτωτικά από την περιοδική τροχιά με περίοδο 2, τον “2-κύκλο”  $\{x_3^*, x_4^*\}$ .

Με τον ίδιο τρόπο μπορούμε να βρούμε πως στο σημείο  $r = r_c^{(3)} = 1 + \sqrt{6}$  έχουμε διακλάδωση των σταθερών σημείων  $x_\alpha^*$ ,  $\alpha = 1, 2, 3, 4$  σε οκτώ σημεία  $x_\alpha^*$ ,  $\alpha = 1, \dots, 8$ . Αυτά είναι πραγματικές λύσεις της εξίσωσης του 4-κύκλου της λογιστικής εξίσωσης  $x = f^{(4)}(x)$ . Για  $r_c^{(3)} < r < r_c^{(4)} \approx 3.5441$  τα τελευταία σημεία  $x_\alpha^*$ ,  $\alpha = 5, \dots, 8$  αποτελούν ελκυστή που είναι ένας 4-κύκλος μιας γενικής τροχιάς της λογιστικής εξίσωσης<sup>6</sup>. Παρόμοια για  $r_c^{(4)} < r < r_c^{(5)}$  έχουμε διακλάδωση σε 16 σταθερά σημεία της  $x = f^{(8)}(x)$  που οδηγούν σε 8-κύκλο και για  $r_c^{(5)} < r < r_c^{(6)}$  σε 16-κύκλο κ.ο.κ<sup>7</sup>. Το φαινόμενο αυτό λέγεται διπλασιασμός περιόδου και συνεχίζει επ’ άπειρον. Τα σημεία  $r_c^{(n)}$  πλησιάζουν διαρκώς μεταξύ τους έτσι, ώστε  $\lim_{n \rightarrow \infty} r_c^{(n)} = r_c \approx 3.56994567$ . Όπως θα δούμε, η τιμή  $r_c$  σηματοδοτεί την έναρξη μη περιοδικής, χαοτικής συμπεριφοράς των τροχιών της λογιστικής απεικόνισης.

Οι αλγεβρικοί υπολογισμοί των σημείων διακλάδωσης είναι πολύπλοκοι και έτσι αναγκαζόμαστε να καταφύγουμε σε αριθμητικές μεθόδους υπολογισμού τους. Αρχικά θα γράψουμε ένα πρόγραμμα που να μας δίνει τροχιές της λογιστικής απεικόνισης για επιλεγμένες τιμές του  $r$  και  $x_0$ . Το πρόγραμμα θα το βρείτε στο αρχείο `logistic.f90` και παρατίθεται παρακάτω.

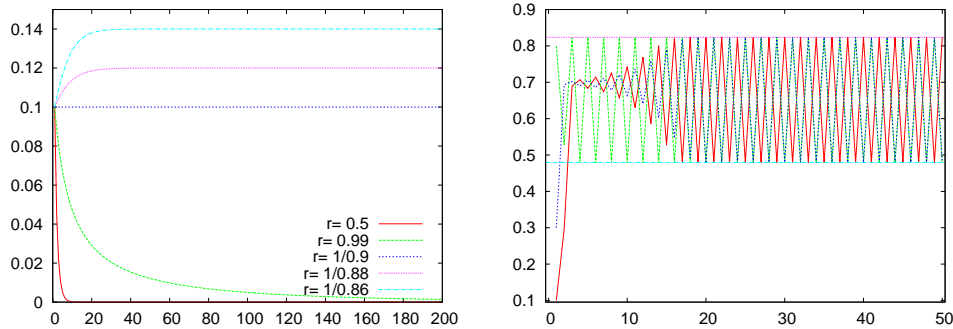
!=====

!Discrete Logistic Map

<sup>5</sup>Από τον κανόνα της αλυσίδας  $dh(g(x))/dx = h'(g(x))g'(x)$  έχουμε ότι  $f^{(2)'}(x_3^*) = df(f(x_3^*))/dx = f'(f(x_3^*))f'(x_3^*) = f'(x_4^*)f'(x_3^*)$  και ανάλογα για την  $f^{(2)'}(x_4^*)$ . Γενικεύοντας μπορούμε να αποδείξουμε ότι για τις  $n$  λύσεις  $x_{n+1}^*, x_{n+2}^*, \dots, x_{2n}^*$  που ανήκουν στον  $n$ -κύκλο της εξίσωσης  $x = f^{(n)}(x)$  έχουμε ότι  $f^{(n)'}(x_{n+i}) = f'(x_{n+1})f'(x_{n+2}) \dots f'(x_{2n})$  για κάθε  $i = 1, \dots, n$ .

<sup>6</sup>Τα  $x_\alpha^*$ ,  $\alpha = 1, \dots, 4$  είναι ασταθή σταθερά σημεία και 2-κύκλος.

<sup>7</sup>Γενικά, για  $r_c^{(n)} < r < r_c^{(n+1)} < r_c \approx 3.56994567$  έχουμε  $2^n$  σταθερά σημεία της εξίσωσης  $x = f^{(2^{n-1})}(x)$  και εμφάνιση  $2^{n-1}$ -κύκλων που είναι ελκυστές.



Σχήμα 3.1: Στο αριστερό σχήμα φαίνονται τροχιές της λογιστικής εξίσωσης για  $x_0 = 0.1$ . Διακρίνεται η πρώτη διακλάδωση για  $r_c^{(1)} = 1$  από το  $x_1^* = 0$  στο  $x_2^* = 1 - 1/r$ . Στο δεξί σχήμα χρησιμοποιούμε  $r = 3.5$ , τέτοιο ώστε  $r_c^{(2)} < r < r_c^{(3)}$ . Οι τρεις καμπύλες είναι για τρία διαφορετικά αρχικά σημεία. Το σύστημα, μετά από μια μεταβατική φάση που εξαρτάται από το αρχικό σημείο, ακολουθεί τροχιά που είναι 2-κύκλος. Οι οριζόντιες γραμμές είναι οι αναμενόμενες τιμές  $x_{3,4}^* = ((r+1) \mp \sqrt{r^2 - 2r - 3})/(2r)$  που αναφέρονται στο κείμενο.

```

!=====
program logistic_map
  implicit none
  integer :: NSTEPS,i
  real(8) :: r,x0,x1
  ! ----- Input:
  print *, '# Enter NSTEPS, r, x0: '
  read *, NSTEPS, r, x0
  print *, '# NSTEPS = ', NSTEPS
  print *, '# r      = ', r
  print *, '# x0     = ', x0
  ! ----- Initialize:
  open(unit=33,file='log.dat')
  write(33,*) 0,x0
  ! ----- Calculate:
  do i=1,NSTEPS
    x1 = r * x0 * (1.0D0-x0)
    write(33,*) i, x1
    x0 = x1
  enddo
  close(33)
end program logistic_map

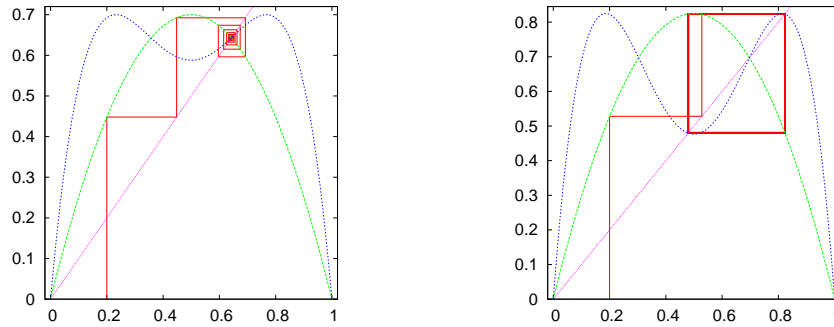
```

Το πρόγραμμα μεταγλωττίζεται και τρέχει κατά τα γνωστά:

```
> gfortran logistic.f90 -o 1
```

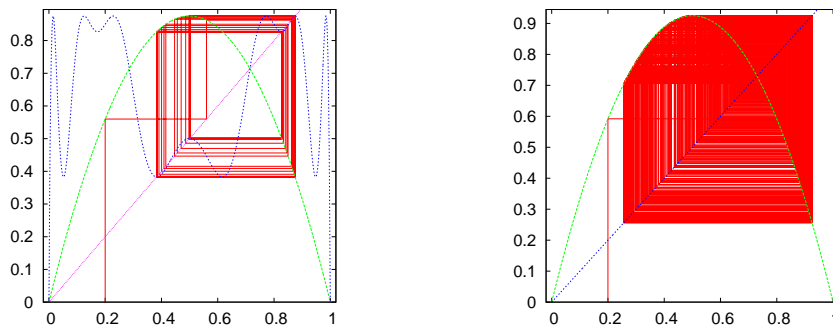
```
> echo "100 0.5 0.1" | ./1
```

όπου με την εντολή `echo` εκτυπώνουμε στο `stdout` τις τιμές των παραμέτρων  $NSTEPS=100$ ,  $r=0.5$  και  $x_0=0.1$ , και τις διοχετεύουμε μέσω `pipe` στο `stdin` του προγράμματος που τρέχει με την εντολή `./1`. Τα αποτελέσματα θα τα βρούμε σε δύο στήλες στο αρχείο `log.dat` και μπορούμε να τα δούμε με το `gnuplot`. Με τον τρόπο αυτό φτιάχνουμε τα σχήματα 3.1. Σε αυτά μπορούμε να δούμε τις δύο πρώτες διακλαδώσεις, καθώς αυξάνεται το  $r$  και ξεπερνάει τις τιμές  $r_c^{(1)}$  και  $r_c^{(2)}$ . Με τον ίδιο τρόπο μπορούμε να μελετήσουμε τροχιές που βρίσκονται ασυμπτωτικά κοντά σε  $2^n$ -κύκλους κάθε φορά που το  $r$  ξεπερνάει την τιμή  $r_c^{(n-1)}$ .



Σχήμα 3.2: Επίπεδη απεικόνιση των τροχιών της λογιστικής εξίσωσης (cobweb plot) για  $r = 2.8$  και  $3.3$ . Στο αριστερό σχήμα βλέπουμε παράδειγμα σταθερού σημείου  $x^* = f(x^*)$ . Απεικονίζεται η συνάρτηση  $f(x)$  (πράσινη καμπύλη), η συνάρτηση  $f^{(2)}(x)$  (μπλε καμπύλη), καθώς και η διαγώνιος. Η τροχιά καταλήγει στη μοναδική μη μηδενική τομή των καμπύλων της διαγωνίου και της  $f(x)$ , το σταθερό σημείο  $x_2^* = 1 - 1/r$ . Η τροχιά τέμνει την  $f^{(2)}(x)$  στο ίδιο σημείο. Δεν υπάρχουν άλλα σημεία τομής της  $f^{(2)}(x)$  με τη διαγώνιο. Δεξιά έχουμε παράδειγμα 2-κύκλου. Η  $f^{(2)}(x)$  τέμνει τη διαγώνιο και σε δύο διαφορετικά σημεία που είναι τα  $x_3^*$  και  $x_4^*$ . Διακρίνουμε το 2-κύκλο στον οποίο καταλήγει ασυμπτωτικά η τροχιά που είναι το ορθογώνιο  $(x_3^*, x_3^*)$ ,  $(x_4^*, x_3^*)$ ,  $(x_4^*, x_4^*)$ ,  $(x_3^*, x_4^*)$ .

Ένας άλλος τρόπος να μελετήσουμε και να καταλάβουμε γραφικά τους  $2^n$ -κύκλους είναι το διάγραμμα που κατασκευάζεται με τον ακόλουθο τρόπο (cobweb plot): Στο επίπεδο επιλέγουμε αρχικά το σημείο  $(x_0, 0)$ . Στη συνέχεια, υπολογίζουμε το σημείο  $(x_0, x_1)$ , όπου  $x_1 = f(x_0)$  και το οποίο είναι πάνω στην καμπύλη της γραφικής παράστασης της  $f(x)$ . Το  $(x_0, x_1)$  το “προβάλλουμε” πάνω στη διαγώνιο  $y = x$  παίρνοντας το σημείο  $(x_1, x_1)$ . Επαναλαμβάνουμε αυτή τη διαδικασία παίρνοντας τα σημεία  $(x_n, x_{n+1})$  και  $(x_{n+1}, x_{n+1})$  πάνω στην καμπύλη  $y = f(x)$  και  $y = x$  αντίστοιχα. Τα σταθερά σημεία  $x^* = f(x^*)$  είναι η τομή των



Σχήμα 3.3: Αριστερά βλέπουμε τον 4-κύκλο τον οποίο πλησιάζει ασυμπτωτικά η τροχιά για  $r = 3.5$ . Η μπλε καμπύλη είναι τώρα η συνάρτηση  $f^{(4)}(x)$  η οποία τέμνει τη διαγώνιο σε τέσσερα σημεία  $x_\alpha$ ,  $\alpha = 5, 6, 7, 8$  και ο 4-κύκλος περνάει από τα σημεία αυτά. Τέλος, δεξιά παρατηρούμε μια μη περιοδική τροχιά που προκύπτει για  $r = 3.7$ , όταν το σύστημα παρουσιάζει χαοτική συμπεριφορά.

καμπύλων αυτών και αν η τροχιά ελκύεται από αυτά, τότε η ακολουθία των παραπάνω σημείων θα συγκλίνει σε αυτά. Αν είμαστε σε  $2^n$ -κύκλο, τότε θα παρατηρήσουμε την περιοδική τροχιά που περνά από τα σημεία που είναι λύσεις της  $x = f^{(2^n)}(x)$ . Για να υλοποιήσουμε την παραπάνω άσκηση, γράφουμε το ακόλουθο πρόγραμμα το οποίο θα βρείτε στο αρχείο `logistic1.f90`:

```
!=====
! Discrete Logistic Map
! Map the trajectory in 2d space (plane)
!=====
program logistic_map
  implicit none
  integer :: NSTEPS,i
  real(8) :: r,x0,x1
  ! ----- Input:
  print *, '# Enter NSTEPS, r, x0:'
  read *, NSTEPS, r, x0
  print *, '# NSTEPS = ', NSTEPS
  print *, '# r = ', r
  print *, '# x0 = ', x0
  ! ----- Initialize:
  open(unit=33, file='trj.dat')
  ! ----- Calculate:
  write(33,*) 0, x0, 0
  do i=1, NSTEPS
    x1 = r * x0 * (1.0D0-x0)
    write(33,*) 2*i-3, x0, x1
```

```

write(33,*) 2*i-2,x1,x1
x0 = x1
enddo
close(33)
end program logistic_map

```

Η μεταγλώττιση και το τρέξιμο γίνεται ακριβώς όπως και για το πρόγραμμα `logistic.f90`. Τα αποτελέσματα μπορούμε να τα δούμε με το `gnuplot`. Για παράδειγμα, η πάνω αριστερά γραφική παράσταση στο σχήμα 3.2 μπορεί να γίνει με τις εντολές:

```

gnuplot> set size square
gnuplot> f(x) = r*x*(1.0-x)
gnuplot> r = 3.3
gnuplot> plot "<echo 50 3.3 0.2|./l;cat trj.dat" using 2:3 w l
gnuplot> replot f(x) ,f(f(x)),x

```

Με την εντολή `plot` τρέχουμε το πρόγραμμα όπως από τη γραμμή εντολών και στη συνέχεια κάνουμε τη γραφική παράσταση από το αρχείο `trj.dat`. Στην επόμενη γραμμή προσθέτουμε τις γραφικές παραστάσεις της  $f(x)$ ,  $f^{(2)}(x) = f(f(x))$  και της διαγωνίου  $y = x$ . Στα σχήματα 3.2 και 3.3 βλέπουμε παραδείγματα από ελκυστές σταθερών σημείων, 2-κύκλων και 4-κύκλων. Επίσης, δίνεται και το παράδειγμα μιας μη περιοδικής τροχιάς που παρουσιάζεται, όταν το σύστημα παρουσιάζει χαοτική συμπεριφορά για  $r > r_c \approx 3.56994567$ .

### 3.3 Διάγραμμα διακλάδωσης

Οι διακλαδώσεις της λογιστικής εξίσωσης μπορούν να απεικονιστούν γραφικά στο “διάγραμμα διακλάδωσης”. Θυμίζουμε ότι οι πρώτες διακλαδώσεις συμβαίνουν για κρίσιμες τιμές του  $r$

$$r_c^{(1)} < r_c^{(2)} < r_c^{(3)} < \dots < r_c^{(n)} < \dots < r_c, \quad (3.19)$$

όπου  $r_c^{(1)} = 1$ ,  $r_c^{(2)} = 3$ ,  $r_c^{(3)} = 1 + \sqrt{6}$  και  $r_c = \lim_{n \rightarrow \infty} r_c^{(n)} \approx 3.56994567$ . Για  $r_c^{(n)} < r < r_c^{(n+1)}$  έχουμε  $2^n$  σταθερά σημεία  $x_\alpha^*$ ,  $\alpha = 1, 2, \dots, 2^n$  της  $x = f^{(2^n)}(x)$ . Απεικονίζοντας αυτά τα σημεία  $x_\alpha^*(r)$  ως συνάρτηση του  $r$  φτιάχνουμε το διάγραμμα διακλάδωσης. Αυτά θα τα υπολογίσουμε αριθμητικά με το πρόγραμμα `bifurcate.f90`. Στο πρόγραμμα αυτό επιλέγουμε τις τιμές του  $r$  που θέλουμε να μελετήσουμε και για κάθε μία από αυτές καταγράφουμε τα σημεία των  $2^{n-1}$ -κύκλων<sup>8</sup>  $x_\alpha^*(r)$ ,

<sup>8</sup>Για να είμαστε ακριβείς, με τον τρόπο αυτό δεν φτιάχνουμε το διάγραμμα διακλάδωσης (bifurcation diagram) το οποίο έχει και τα ασταθή σημεία, αλλά το διάγραμμα

$\alpha = 2^{n-1} + 1, 2^{n-1} + 2, \dots, 2^n$ . Αυτό μπορεί να γίνει εύκολα αν εφαρμόσουμε επαγωγικά τη λογιστική εξίσωση αρκετές φορές, έτσι ώστε οι τροχιές να φτάσουν να εκτελούν πρακτικά μόνο τους  $2^{n-1}$ -κύκλους. Στο πρόγραμμα, η παράμετρος που καθορίζει τα σημεία της τροχιάς που “πετάμε” είναι η NTRANS. Μετά καταγράφουμε NSTEPS σημεία της τροχιάς τα οποία υποθέτουμε ότι το NTRANS είναι αρκετά μεγάλο, έτσι ώστε να καταγράψουμε μόνο τα σημεία των  $2^{n-1}$ -κύκλων. Το πρόγραμμα δίνεται παρακάτω:

```

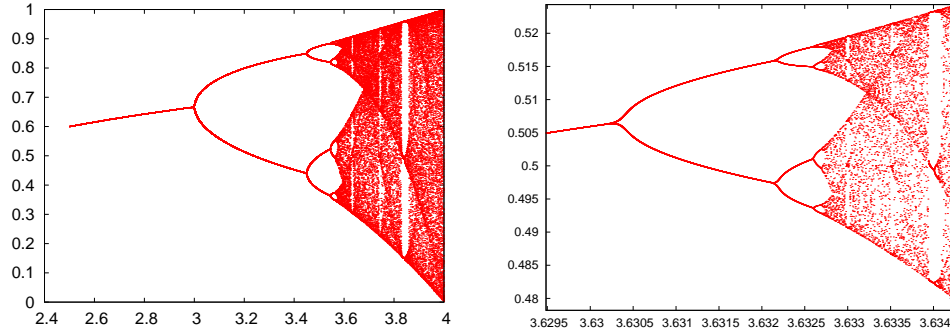
=====
! Bifurcation Diagram of the Logistic Map
=====
program bifurcation_diagram
  implicit none
  real(8),parameter :: rmin  = 2.5D0
  real(8),parameter :: rmax  = 4.0D0
  integer,parameter :: NTRANS = 500  !Number of discarded steps
  integer,parameter :: NSTEPS = 100  !Number of recorded steps
  integer,parameter :: RSTEPS = 2000 !Number of values of r
  integer           :: i
  real(8)           :: r,dr,x0,x1
! ----- Initialize:
  open(unit=33,file='bif.dat')
  dr      = (rmax-rmin)/RSTEPS !Increment in r
! ----- Calculate:
  r = rmin
  do while ( r .le. rmax)
    x0 = 0.5D0
! ----- Transient steps: skip
    do i=1,NTRANS
      x1 = r * x0 * (1.0D0-x0)
      x0 = x1
    enddo
    do i=1,NSTEPS
      x1 = r * x0 * (1.0D0-x0)
      write(33,*) r,x1
      x0 = x1
    enddo
    r = r + dr
  enddo ! do while
  close(33)
end program bifurcation_diagram

```

Το πρόγραμμα μεταγλωττίζεται και τρέχει πολύ απλά με τις εντολές

---

τροχιών (orbit diagram). Για απλότητα χρησιμοποιούμε το ίδιο όνομα.



Σχήμα 3.4: Αριστερά δείχνεται το διάγραμμα διακλάδωσης όπως προκύπτει από το πρόγραμμα `bifurcate.f90` για  $2.5 < r < 4$ . Παρατηρούμε τις πρώτες διακλαδώσεις οι οποίες ακολουθούνται από διαδοχικές περιοχές χαοτικών, μη περιοδικών, τροχιών με “διαλείμματα” περιοδικών τροχιών. Οι χαοτικές τροχιές παίρνουν τιμές σε υποδιαστήματα του διαστήματος  $(0, 1)$  μέχρι για  $r = 4$  όπου παίρνουν τιμές μέσα σε ολόκληρο το διάστημα. Παρατηρούμε για  $r = 1 + \sqrt{8} \approx 3.8284$  τον 3-κύκλο ο οποίος στη συνέχεια διακλαδώνεται σε  $3 \cdot 2^n$ -κύκλους. Δεξιά μεγεθύνουμε το διάγραμμα σε μια περιοχή τιμών του  $r$  που καταδεικνύει την αυτομοιότητα (self-similarity) του διαγράμματος σε όλες τις κλίμακες.

```
> gfortran bifurcate.f90 -o b
> ./b;
```

και μπορούμε να κατασκευάσουμε το αριστερό σχήμα 3.4 με το `gnuplot`

```
gnuplot> plot "bif.dat" with dots
```

Παρατηρούμε τα ευσταθή σταθερά σημεία και τους  $2^n$ -κύκλους για  $r < r_c$ . Μόλις το  $r$  ξεπεράσει την τιμή  $r_c$ , οι τροχιές είναι μη-περιοδικές και το σύστημα παρουσιάζει χαοτική συμπεριφορά. Θα συζητήσουμε την έννοια της χαοτικής συμπεριφοράς με περισσότερη λεπτομέρεια σε επόμενη παράγραφο. Για την ώρα, απλά επισημαίνουμε ότι από το διάγραμμα μπορούμε να μετρήσουμε την απόσταση μεταξύ των κρίσιμων σημείων  $\Delta r^{(n)} = r_c^{(n+1)} - r_c^{(n)}$  η οποία διαρκώς μειώνεται σχεδόν γεωμετρικά με το  $n$ , έτσι ώστε

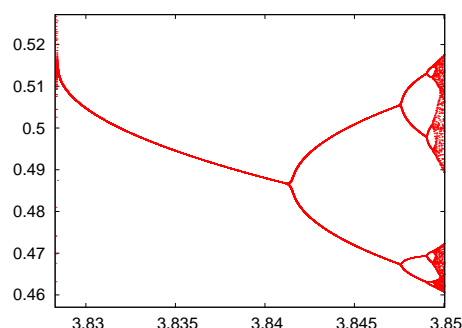
$$\lim_{n \rightarrow \infty} \frac{\Delta r^{(n)}}{\Delta r^{(n+1)}} = \delta \approx 4.669\,201\,609, \quad (3.20)$$

όπου  $\delta$  είναι η σταθερά του Feigenbaum. Επίσης, παρατηρείται ότι τα πλάτη των διακλαδώσεων  $\Delta w_n$  επίσης μειώνονται, έτσι ώστε

$$\lim_{n \rightarrow \infty} \frac{\Delta w_n}{\Delta w_{n+1}} = \alpha \approx 2.502\,907\,875. \quad (3.21)$$



Είναι ενδιαφέρον, επίσης να επισημάνουμε την εμφάνιση σταθερού 3-κύκλου για  $r = 1 + \sqrt{8} \approx 3.8284 > r_c$ ! Χρησιμοποιώντας το θεώρημα Sharkovskii, οι Li and Yorke<sup>9</sup> δείχνανε πως οποιοδήποτε μονοδιάστατο σύστημα έχει 3-κύκλους, τότε θα έχει και κύκλους οποιουδήποτε μήκους και χαοτικές τροχιές. Ο 3-κύκλος και η ευστάθειά του μπορεί να υπολογιστεί μελετώντας τις λύσεις της εξίσωσης  $x = f^{(3)}(x)$  όπως κάναμε με τις σχέσεις (3.16) και (3.17) (δείτε [21] για λεπτομέρειες). Μεγεθύνουμε ένα κλάδο αυτού του 3-κύκλου στο σχήμα 3.5. Κύκλους όμοιους με τον 3-κύκλο μπορούμε να δούμε να επαναλαμβάνονται σε διαφορετικές κλίμακες όπως φαίνεται και στο δεξί σχήμα 3.4. Στο σχήμα 3.4 παρατη-



Σχήμα 3.5: Μεγέθυνση μιας από τις τρεις διακλαδώσεις του 3-κύκλου για  $r > 1 + \sqrt{8}$ . Αριστερά παρατηρούμε την προσωρινή παύση της χαοτικής συμπεριφοράς των τροχιών η οποία επανέρχεται δεξιά μετά από ένα διάλειμμα περιοδικών τροχιών.

ρούμε ότι ανάμεσα στις χαοτικές τροχιές της λογιστικής εξίσωσης παρουσιάζονται “παράθυρα” περιοδικών τροχιών. Αυτά είναι άπειρα στον αριθμό, αλλά αριθμήσιμα. Ακόμα πιο ενδιαφέρον είναι το γεγονός ότι, αν πάρουμε ένα κλάδο μέσα σε αυτά τα παράθυρα και τον μεγεθύνουμε, αυτός είναι όμοιος με ολόκληρο το διάγραμμα! Ένα τέτοιο παράδειγμα μπορούμε να δούμε στο δεξί σχήμα 3.4. Λέμε ότι το διάγραμμα διακλάδωσης παρουσιάζει ιδιότητες αυτομοιότητας (self-similarity). Υπάρχουν και άλλες ενδιαφέρουσες ιδιότητες του διαγράμματος διακλάδωσης για τις οποίες παραπέμπουμε τον αναγνώστη στη βιβλιογραφία.

Κλείνοντας, να σημειώσουμε ότι οι ποιοτικές ιδιότητες του διαγράμματος διακλάδωσης είναι όμοιες για μια ολόκληρη κλάση από συναρτήσεις. Ο Feigenbaum ανακάλυψε ότι εκτός από τη λογιστική απεικόνιση, κάθε συνάρτηση που είναι κοίλη με μοναδικό μέγιστο έχει τις

<sup>9</sup>T.Y. Li, J.A. Yorke, “Period Three Implies Chaos”, American Mathematical Monthly 82 (1975) 985.

ίδιες ιδιότητες, όπως για παράδειγμα οι συναρτήσεις<sup>10</sup>  $g(x) = xe^{r(1-x)}$ ,  $u(x) = r \sin(\pi x)$  και  $w(x) = b - x^2$ . Οι σταθερές  $\delta$  και  $\alpha$  των σχέσεων (3.20) και (3.21) είναι οι ίδιες για όλες αυτές τις συναρτήσεις. Οι απεικονίσεις που παράγουν χαοτική συμπεριφορά μελετώνται με πολύ ενδιαφέρον και μπορείτε να βρείτε μια λίστα από αυτές στην αναφορά [27].

### 3.4 Μέθοδος Newton-Raphson

Για τον εντοπισμό των σημείων διακλάδωσης θα χρειαστεί να λύσουμε τις, μη γραμμικές πολυωνυμικές αλγεβρικές εξισώσεις  $x = f^{(n)}(x)$  και  $f^{(n)'}(x) = -1$ . Για το λόγο αυτό πρέπει να καταφύγουμε σε προσεγγιστικό, αριθμητικό υπολογισμό, και η απλή μέθοδος Newton-Raphson θα αποδειχθεί μια καλή επιλογή.

Η μέθοδος Newton-Raphson για την εύρεση των ριζών μιας εξίσωσης  $g(x) = 0$  είναι μια επαγωγική μέθοδος που χρησιμοποιεί ένα αρχικό σημείο  $x_0$  και υπολογίζει με τη βοήθεια της παραγώγου  $g'(x)$  διαδοχικά σημεία  $x_1, x_2, \dots, x_n, x_{n+1}, \dots$  τα οποία είναι ολοένα και καλύτερες προσεγγίσεις μιας από τις ρίζες της εξίσωσης. Η επαγωγή σταματάει όταν αποφασίσουμε ότι πετύχαμε το επιθυμητό επίπεδο σύγκλισης. Υποθέτοντας ότι η συνάρτηση  $g(x)$  είναι αναλυτική, από το ανάπτυγμά της κατά Taylor γύρω από το σημείο  $x_n$  παίρνουμε

$$g(x_{n+1}) = g(x_n) + (x_{n+1} - x_n)g'(x) + \dots \quad (3.22)$$

Αν θέλουμε να έχουμε  $g(x_{n+1}) \approx 0$ , τότε πρέπει να πάρουμε

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}. \quad (3.23)$$

Η παραπάνω εξίσωση δίνει και τον αλγόριθμο της μεθόδου ο οποίος είναι μια επαγωγική σχέση ενός βήματος.

Διαφορετικά σημεία  $x_0$  θα οδηγήσουν σε διαφορετικές ρίζες της εξίσωσης, όταν η σύγκλιση είναι επιτυχής. Η σύγκλιση της μεθόδου είναι τετραγωνική με τον αριθμό των επαναλήψεων, όταν οι παράγωγοι  $g'(x)$ ,  $g''(x)$  είναι μη μηδενικές στη ρίζα και η τρίτη παράγωγος είναι φραγμένη: Υπάρχει μια περιοχή της ρίζας, έστω  $\alpha$ , τέτοια ώστε η απόσταση  $\Delta x_{n+1} = x_{n+1} - \alpha$  να είναι  $\Delta x_{n+1} \propto (\Delta x_n)^2$ . Αν η ρίζα  $\alpha$  είναι πολλαπλή,

<sup>10</sup>Η συνάρτηση  $x \exp(r(1-x))$  έχει χρησιμοποιηθεί ως μοντέλο πληθυσμών που η μεγάλη τους πυκνότητα περιορίζεται από επιδημίες. Οι πληθυσμοί είναι πάντα θετικοί ανεξάρτητα από τις (θετικές) αρχικές συνθήκες και την τιμή του  $r$ .

τότε η σύγκλιση είναι πιο αργή. Οι αποδείξεις είναι απλές και μπορούν να βρεθούν στην [28].

Η μέθοδος αυτή είναι απλή στον προγραμματισμό και τις περισσότερες φορές αρκετή για τη λύση αρκετών προβλημάτων. Αλλά στη γενική περίπτωση δουλεύει καλά μόνο κοντά στην περιοχή των ριζών. Θα πρέπει επίσης να έχουμε στο νου μας ότι η μέθοδος αυτή αποτυγχάνει σε απλές περιπτώσεις: Όταν  $g'(x_n) = 0$  για κάποιο  $n$ , η μέθοδος σταματάει. Σε συναρτήσεις που τείνουν στο 0 όταν  $x \rightarrow \pm\infty$  είναι εύκολο να κάνουμε κακή επιλογή του  $x_0$  που να μη συγκλίνει προς κάποια ρίζα. Για να αποφύγουμε την κακή επιλογή του  $x_0$  μπορούμε να συνδυάσουμε τη μέθοδο Newton-Raphson με τη μέθοδο της διχοτόμησης (bisection method). Όταν η παράγωγος  $g'(x)$  αποκλίνει στη ρίζα, μπορεί να έχουμε προβλήματα. Για παράδειγμα, η εξίσωση  $|x|^\nu = 0$  με  $0 < \nu < 1/2$  δεν δίνει συγκλίνουσα ακολουθία. Σε ορισμένες περιπτώσεις, η μέθοδος μπορεί να οδηγήσει και σε μη-συγκλίνοντες κύκλους [8]. Σε ποιο ομαλές συναρτήσεις η περιοχή ελκυσμού (basin of attraction) μιας ρίζας μπορεί να είναι πάρα πολύ μικρή. Αυτές είναι οι περιοχές στον πραγματικό άξονα που αν επιλεγούν ως  $x_0$  θα οδηγήσουν στη συγκεκριμένη ρίζα. Δείτε την άσκηση 13.

Για να προγραμματίσουμε τη μέθοδο Newton-Raphson ας πάρουμε το παράδειγμα της εξίσωσης

$$\epsilon \tan \epsilon = \sqrt{\rho^2 - \epsilon^2} \quad (3.24)$$

η οποία προκύπτει από την επίλυση της εξίσωσης Schrödinger για την εύρεση του ενεργειακού φάσματος ενός σωματιδίου μάζας  $m$  σε ένα μονοδιάστατο πηγάδι δυναμικού βάθους  $V_0$  και πλάτους  $L$ . Οι σταθερές  $\epsilon = \sqrt{mL^2 E}/(2\hbar)$  και  $\rho = \sqrt{mL^2 V_0}/(2\hbar)$ . Επιλύοντας ως προς  $\epsilon$  δεδομένου του  $\rho$  μπορούμε να υπολογίσουμε τις δυνατές τιμές της ενέργειας  $E$ . Η συνάρτηση  $g(x)$  της οποίας αναζητούμε τις ρίζες και η παράγωγός της  $g'(x)$  είναι

$$\begin{aligned} g(x) &= x \tan x - \sqrt{\rho^2 - x^2} \\ g'(x) &= \frac{x}{\sqrt{\rho^2 - x^2}} + \frac{x}{\cos^2 x} + \tan x. \end{aligned} \quad (3.25)$$

Το πρόγραμμα το οποίο υλοποιεί τη μέθοδο για την παραπάνω συνάρτηση θα το βρείτε στο αρχείο nr.f90:

```
!=====
!Newton Raphson for a function of one variable
!=====
```

```

program NewtonRaphson
  implicit none
  real(8), parameter :: rho = 15.0D0
  real(8), parameter :: eps = 1D-6
  integer, parameter :: NMAX = 1000
  real(8) :: x0, x1, err, g, gp
  integer :: i
  print *, 'Enter x0: '
  read *, x0
  err = 1.0D0
  print *, 'iter          x                      error          '
  print *, '-----'
  print *, 0,x0,err
  do i=1,NMAX
    !value of function g(x):
    g = x0*tan(x0)-sqrt(rho*rho-x0*x0)
    !value of the derivative g'(x):
    gp = x0/sqrt(rho*rho-x0*x0)+x0/(cos(x0)**2)+tan(x0)
    x1 = x0 - g/gp
    err = ABS(x1-x0)
    print *,i,x1,err
    if(err .lt. eps) exit
    x0 = x1
  enddo
end program NewtonRaphson

```

Στο παραπάνω πρόγραμμα ζητάμε από το χρήστη μόνο το αρχικό σημείο  $x_0 = x0$ , ενώ θέτουμε την παράμετρο  $\rho = \text{rho} = 15$ . Για να λύσουμε το πρόβλημα, βοηθάει να κάνουμε τη γραφική παράσταση της (3.24). Με το gnuplot παριστάνουμε γραφικά τα δύο μέλη της (3.24) και από τα σημεία τομής τους μπορούμε να εκτιμήσουμε πού βρίσκονται οι ρίζες που αναζητάμε έτσι ώστε να επιλέξουμε κατάλληλα αρχικά σημεία  $x_0$ :

```

gnuplot> g1(x) = x*tan(x)
gnuplot> g2(x) = sqrt(rho*rho-x*x)
gnuplot> plot [0:20][0:20] g1(x), g2(x)

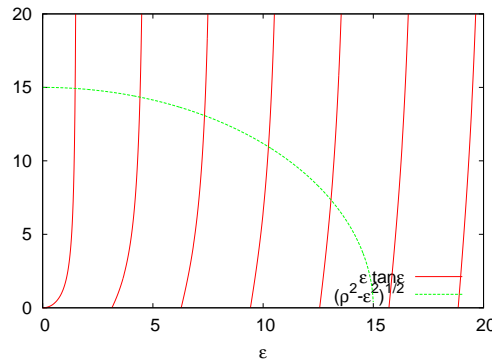
```

Μπορούμε τώρα να μεταγλωττίσουμε και να τρέξουμε το πρόγραμμα:

```

> gfortran nr.f90 -o n
> echo "1.4"|./n
Enter x0:
iter x                      error
-----
0      1.3999999999999999    1.0000000000000000
1      1.5254292024457967    0.12542920244579681

```



Σχήμα 3.6: Γραφική παράσταση των συναρτήσεων που βρίσκονται στα δύο μέλη της (3.24). Οι τομές των καμπύλων δίνουν προσεγγιστικά τις λύσεις της εξίσωσης και μπορούν να χρησιμοποιηθούν ως αρχικά σημεία στη μέθοδο Newton-Raphson.

2	1.5009739120496131	2.4455290396183660E-002
3	1.4807207017202200	2.0253210329393090E-002
4	1.4731630533073483	7.5576484128716537E-003
5	1.4724779331237687	6.8512018357957949E-004
6	1.4724731072313519	4.8258924167932093E-006
7	1.4724731069952235	2.3612845012621619E-010

Συμπεραίνουμε πως μια ρίζα της εξίσωσης είναι η  $\epsilon \approx 1.472473107$ . Με ανάλογο τρόπο υπολογίζουμε και τις υπόλοιπες ρίζες, κάτι το οποίο το αφήνουμε σαν άσκηση στον αναγνώστη.

Η παραπάνω μέθοδος μπορεί να γενικευτεί και για ένα σύστημα από δύο αλγεβρικές εξισώσεις. Έστω ότι έχουμε να λύσουμε τις εξισώσεις  $g_1(x_1, x_2) = 0$  και  $g_2(x_1, x_2) = 0$ . Για να βρούμε το ανάλογο επαγωγικό σχήμα που θα οδηγήσει σε μια συγκλίνουσα προς μια ρίζα ακολουθία  $(x_{10}, x_{20}), (x_{11}, x_{21}), \dots, (x_{1n}, x_{2n}), (x_{1(n+1)}, x_{2(n+1)}), \dots$ , αναπτύσσουμε κατά Taylor τις δύο συναρτήσεις γύρω από το σημείο  $(x_{1n}, x_{2n})$

$$\begin{aligned}
 g_1(x_{1(n+1)}, x_{2(n+1)}) &= g_1(x_{1n}, x_{2n}) + (x_{1(n+1)} - x_{1n}) \frac{\partial g_1(x_{1n}, x_{2n})}{\partial x_1} \\
 &\quad + (x_{2(n+1)} - x_{2n}) \frac{\partial g_1(x_{1n}, x_{2n})}{\partial x_2} + \dots \\
 g_2(x_{1(n+1)}, x_{2(n+1)}) &= g_2(x_{1n}, x_{2n}) + (x_{1(n+1)} - x_{1n}) \frac{\partial g_2(x_{1n}, x_{2n})}{\partial x_1} \\
 &\quad + (x_{2(n+1)} - x_{2n}) \frac{\partial g_2(x_{1n}, x_{2n})}{\partial x_2} + \dots \quad (3.26)
 \end{aligned}$$

Ορίζοντας  $\delta x_1 = (x_{1(n+1)} - x_{1n})$  και  $\delta x_2 = (x_{2(n+1)} - x_{2n})$  και θέτοντας

$g_1(x_{1(n+1)}, x_{2(n+1)}) \approx 0$ ,  $g_2(x_{1(n+1)}, x_{2(n+1)}) \approx 0$ , παίρνουμε

$$\begin{aligned} \delta x_1 \frac{\partial g_1}{\partial x_1} + \delta x_2 \frac{\partial g_1}{\partial x_2} &= -g_1 \\ \delta x_1 \frac{\partial g_2}{\partial x_1} + \delta x_2 \frac{\partial g_2}{\partial x_2} &= -g_2. \end{aligned} \quad (3.27)$$

Αυτό είναι ένα γραμμικό  $2 \times 2$  σύστημα

$$\begin{aligned} A_{11}\delta x_1 + A_{12}\delta x_2 &= b_1 \\ A_{21}\delta x_1 + A_{22}\delta x_2 &= b_2, \end{aligned} \quad (3.28)$$

όπου  $A_{ij} = \partial g_i / \partial x_j$  και  $b_i = -g_i$  με  $i, j = 1, 2$ . Λύνοντας ως προς  $\delta x_i$  παίρνουμε την επαγωγική σχέση

$$\begin{aligned} x_{1(n+1)} &= x_{1n} + \delta x_1 \\ x_{2(n+1)} &= x_{2n} + \delta x_2. \end{aligned} \quad (3.29)$$

Ο αλγόριθμος σταματάει όταν τα  $\delta x_i$  γίνουν αρκετά μικρά.

Για παράδειγμα θα λύσουμε τις εξισώσεις με  $g_1(x) = 2x^2 - 3xy + y - 2$ ,  $g_2(x) = 3x + xy + y - 1$ . Έχουμε  $A_{11} = 4x - 3y$ ,  $A_{12} = 1 - 3x$ ,  $A_{21} = 3 + y$ ,  $A_{22} = 1 + x$ . Γράφουμε το πρόγραμμα στο αρχείο nr2.f90:

```
!=====
!Newton Raphson of two functions of two variables
!=====
program NewtonRaphson2
  implicit none
  real(8), parameter :: eps = 1D-6
  integer, parameter :: NMAX = 1000
  real(8) :: A(2,2), b(2), dx(2)
  real(8) :: x, y, err
  integer :: i
  print *, 'Enter x0,y0: '
  read *, x, y
  err = 1.0D0
  print *, 'iter      x      y      error      '
  print *, '-----'
  print *, 0, x, y, err
  do i=1, NMAX
    b(1) = -(2.0D0*x*x-3.0D0*x*y+y-2.0D0) ! -g1(x,y)
    b(2) = -(3.0D0*x + x*y + y - 1.0D0) ! -g2(x,y)
    ! dg1/dx      dg1/dy
    A(1,1) = 4.0D0*x-3.0D0*y; A(1,2) = 1.0D0-3.0D0*x
    ! dg2/dx      dg2/dy
```

```

A(2,1) = 3.0D0+y          ; A(2,2) = 1.0D0+x
call solve2x2(A,B,dx)
x = x + dx(1)
y = y + dx(2)
err = 0.5D0*SQRT(dx(1)**2+dx(2)**2)
print *,i,x,y,err
if(err .lt. eps) exit
enddo
end program NewtonRaphson2
!=====
subroutine solve2x2(A,b,dx)
implicit none
real(8) :: A(2,2),b(2),dx(2)
real(8) :: num1,num2,det
num1 = A(2,2)*b(1)-A(1,2)*b(2)
num2 = A(1,1)*b(2)-A(2,1)*b(1)
det = A(1,1)*A(2,2)-A(1,2)*A(2,1)
if(det .eq. 0.0D0) stop 'solve2x2: det=0'
dx(1)= num1/det
dx(2)= num2/det
end subroutine solve2x2

```

Για να πάρουμε μια ιδέα πού περίπου είναι οι πραγματικές ρίζες του συστήματος, κάνουμε ένα τρισδιάστατο γράφημα με το gnuplot:

```

gnuplot> set isosamples 20
gnuplot> set hidden3d
gnuplot> splot 2*x**2-3*x*y+y-2,3*x+y*x+y-1,0

```

Εκτός από τις  $g_i(x, y)$  κάνουμε και τη γραφική παράσταση του  $x = 0$  επιπέδου και από την τομή των τριών επιφανειών εντοπίζουμε τις ρίζες. Η μεταγλώττιση και το τρέξιμο του προγράμματος δίνει:

```

> gfortran nr2.f90 -o n
> echo 2.2 1.5 |./n
Enter x0,y0:
iter      x          y          error
-----
0  2.20000000  1.50000000  1.0000
1  0.76427104  0.26899383  0.9456
2  0.73939531 -0.68668275  0.4780
3  0.74744506 -0.71105605  1.2834E-002
4  0.74735933 -0.71083147  1.2019E-004
5  0.74735932 -0.71083145  1.2029E-008
> echo 0 1 |./n
.....
5 -0.10899022  1.48928857  4.3461E-012

```

```
> echo -5 0 | ./n
6 -6.13836909 -3.77845711 3.2165E-013
```

Βρήκαμε έτσι τρεις ρίζες  $(0.74735932, -0.71083145), (-0.10899022, 1.48928857), (-6.13836909, -3.77845711)$ .

Η μέθοδος Newton-Raphson για πολλές μεταβλητές γίνεται ακριβή: έχουμε εκτός από τον υπολογισμό της συνάρτησης και τον υπολογισμό των παραγώγων που μπορεί να γίνει απαγορευτικά ακριβός σε ορισμένα προβλήματα. Επίσης, γίνεται προβληματικός ο εντοπισμός των λύσεων αφού η μέθοδος συγκλίνει γρήγορα μόνο κοντά στην περιοχή των ριζών. Παραπέμπουμε τον αναγνώστη στην [8] για περισσότερες πληροφορίες για το πώς αντιμετωπίζονται αυτά τα προβλήματα.

### 3.5 Υπολογισμός Σημείων Διακλάδωσης

Για τον εντοπισμό των σημείων διακλάδωσης για  $r < r_c$  θα λύσουμε τις αλγεβρικές εξισώσεις  $x = f^{(k)}(x)$  και  $f^{(k)'}(x) = -1$ . Στα σημεία αυτά, θα έχουμε τους  $k$ -κύκλους να γίνονται ασταθείς και να γίνονται σταθεροί οι  $2k$ -κύκλοι. Αυτό γίνεται όταν  $r = r_c^{(n)}$  με  $k = 2^{n-2}$ . Θα αναζητήσουμε ως λύσεις τα σημεία  $(x_\alpha^*, r_c^{(n)})$  για  $\alpha = k+1, k+2, \dots, 2k$

Ορίζουμε τις συναρτήσεις  $F(x, r) = f(x) - rx(1-x)$  και  $F^{(k)}(x, r) = f^{(k)}(x) - rx(1-x)$  όπως κάναμε στην (3.6). Για να εντοπίσουμε τα σημεία διακλάδωσης θα λύσουμε τις αλγεβρικές εξισώσεις:

$$\begin{aligned} g_1(x, r) &= x - F^{(k)}(x, r) = 0 \\ g_2(x, r) &= \frac{\partial F^{(k)}(x, r)}{\partial x} + 1 = 0. \end{aligned} \quad (3.30)$$

Σύμφωνα με την προηγούμενη παράγραφο, για τον υπολογισμό των ριζών των παραπάνω εξισώσεων θα λύσουμε το γραμμικό σύστημα (3.28)



με τους συντελεστές

$$\begin{aligned}
 b_1 &= -g_1(x, r) = -x + F^{(k)}(x, r) \\
 b_2 &= -g_2(x, r) = -\frac{\partial F^{(k)}(x, r)}{\partial x} - 1 \\
 A_{11} &= \frac{\partial g_1(x, r)}{\partial x} = 1 - \frac{\partial F^{(k)}(x, r)}{\partial x} \\
 A_{12} &= \frac{\partial g_1(x, r)}{\partial r} = -\frac{\partial F^{(k)}(x, r)}{\partial r} \\
 A_{21} &= \frac{\partial g_2(x, r)}{\partial x} = \frac{\partial^2 F^{(k)}(x, r)}{\partial x^2} \\
 A_{22} &= \frac{\partial g_2(x, r)}{\partial r} = \frac{\partial^2 F^{(k)}(x, r)}{\partial x \partial r}.
 \end{aligned} \tag{3.31}$$

Τις παραγώγους δεν μπορούμε να τις υπολογίσουμε αναλυτικά, οπότε θα καταφύγουμε στην προσέγγιση

$$\begin{aligned}
 \frac{\partial F^{(k)}(x, r)}{\partial x} &\approx \frac{F^{(k)}(x + \epsilon, r) - F^{(k)}(x - \epsilon, r)}{2\epsilon} \\
 \frac{\partial F^{(k)}(x, r)}{\partial r} &\approx \frac{F^{(k)}(x, r + \epsilon) - F^{(k)}(x, r - \epsilon)}{2\epsilon},
 \end{aligned} \tag{3.32}$$

και για τις δεύτερες παραγώγους

$$\begin{aligned}
 \frac{\partial^2 F^{(k)}(x, r)}{\partial x^2} &\approx \frac{\frac{\partial F^{(k)}(x + \frac{\epsilon}{2}, r)}{\partial x} - \frac{\partial F^{(k)}(x - \frac{\epsilon}{2}, r)}{\partial x}}{2\frac{\epsilon}{2}} \\
 &= \frac{1}{\epsilon} \left\{ \frac{F^{(k)}(x + \epsilon, r) - F^{(k)}(x, r)}{\epsilon} - \frac{F^{(k)}(x, r) - F^{(k)}(x - \epsilon, r)}{\epsilon} \right\} \\
 &= \frac{1}{\epsilon^2} \{ F^{(k)}(x + \epsilon, r) - 2F^{(k)}(x, r) + F^{(k)}(x - \epsilon, r) \} \\
 \frac{\partial^2 F^{(k)}(x, r)}{\partial x \partial r} &\approx \frac{\frac{\partial F^{(k)}(x + \epsilon_x, r)}{\partial r} - \frac{\partial F^{(k)}(x - \epsilon_x, r)}{\partial r}}{2\epsilon_x} \\
 &= \frac{1}{2\epsilon_x} \left\{ \frac{F^{(k)}(x + \epsilon_x, r + \epsilon_r) - F^{(k)}(x + \epsilon_x, r - \epsilon_r)}{2\epsilon_r} \right. \\
 &\quad \left. - \frac{F^{(k)}(x - \epsilon_x, r + \epsilon_r) - F^{(k)}(x - \epsilon_x, r - \epsilon_r)}{2\epsilon_r} \right\} \\
 &= \frac{1}{4\epsilon_x \epsilon_r} \{ F^{(k)}(x + \epsilon_x, r + \epsilon_r) - F^{(k)}(x + \epsilon_x, r - \epsilon_r) \\
 &\quad - F^{(k)}(x - \epsilon_x, r + \epsilon_r) + F^{(k)}(x - \epsilon_x, r - \epsilon_r) \}
 \end{aligned} \tag{3.33}$$

Μπορούμε να γράψουμε τώρα το πρόγραμμα για τη Newton-Raphson όπως και στην προηγούμενη παράγραφο. Η μόνη διαφορά θα είναι ο υπολογισμός των παραγώγων από τις παραπάνω προσεγγιστικές σχέσεις και ο υπολογισμός της  $F^{(k)}(x, r)$  από συνάρτηση που θα συνθέτει την  $f(x)$   $k$ -φορές. Το πρόγραμμα θα το βρείτε στο αρχείο `bifurcationPoints.f90`:

```
!=====
!           bifurcationPoints.f
! Calculate bifurcation points of the discrete logistic map
! at period k by solving the condition
!  $g1(x, r) = x - F(k, x, r) = 0$ 
!  $g2(x, r) = dF(k, x, r)/dx + 1 = 0$ 
! determining when the Floquet multiplier becomes 1
!  $F(k, x, r)$  iterates  $F(x, r) = r * x * (x - 1)$  k times
! The equations are solved by using a Newton-Raphson method
!=====
program bifurcationPoints
  implicit none
  real(8), parameter :: tol=1.0D-10
  integer :: k, iter
  real(8) :: r0, x0
  real(8) :: A(2,2), B(2), dX(2)
  real(8) :: error
  real(8) :: F, dFdx, dFdr, d2Fdx2, d2Fdxdx
! ----- Input:
  print *, '# Enter k, r0, x0: '
  read *, k, r0, x0
  print *, '# Period k= ', k
  print *, '# r0= ', r0, ' x0= ', x0
! ----- Initialize
  error = 1.0D0 !initial large value of error>tol
  iter = 0
  do while(error .gt. tol)
! ----- Calculate jacobian matrix
    A(1,1) = 1.0D0 - dFdx(k, x0, r0)
    A(1,2) = -dFdr(k, x0, r0)
    A(2,1) = d2Fdx2(k, x0, r0)
    A(2,2) = d2Fdxdx(k, x0, r0)
    B(1) = -x0 + F(k, x0, r0)
    B(2) = -dFdx(k, x0, r0) - 1.0D0
! ----- Solve a 2x2 linear system:
    call solve2x2(A, B, dX)
    x0 = x0 + dX(1)
    r0 = r0 + dX(2)
    error = 0.5D0 * sqrt(dX(1)**2 + dX(2)**2)
    iter = iter + 1
  end do
end program
```

```

    print*,iter,'x0= ',x0,' r0= ',r0,' err=',error
enddo !do while(error .gt. tol)
end program bifurcationPoints
!=====
!Function F(k,x,r) and its derivatives
real(8) function F(k,x,r)
    implicit none
    real(8) :: x,r,x0
    integer k,i

    x0 = x
    do i=1,k
        x0 = r*x0*(1.0D0-x0)
    enddo
    F = x0

end function F
!-----
real(8) function dFdx(k,x,r)
    implicit none
    real(8) :: x,r,eps
    real(8) :: F
    integer k

    eps = 1.0D-6*x
    dFdx = (F(k,x+eps,r)-F(k,x-eps,r))/(2.0D0*eps)
end function dFdx
!-----
real(8) function dFdr(k,x,r)
    implicit none
    real(8) :: x,r,eps
    real(8) :: F
    integer k

    eps = 1.0D-6*r
    dFdr = (F(k,x,r+eps)-F(k,x,r-eps))/(2.0D0*eps)
end function dFdr
!-----
real(8) function d2Fdx2(k,x,r)
    implicit none
    real(8) :: x,r,eps
    real(8) :: F
    integer k

    eps = 1.0D-6*x
    d2Fdx2 = (F(k,x+eps,r)-2.0D0*F(k,x,r)+F(k,x-eps,r))/(eps*eps)
end function d2Fdx2
!-----
real(8) function d2Fdrdx(k,x,r)

```

```

implicit none
real(8) :: x,r,epsx,epsr
real(8) :: F
integer k

epsx    = 1.0D-6*x
epsr    = 1.0D-6*r
d2Fdrdx = (F(k,x+epsx,r+epsr)-F(k,x+epsx,r-epsr) &
           -F(k,x-epsx,r+epsr)+F(k,x-epsx,r-epsr)) &
           /(4.0D0*epsx*epsr)
end function d2Fdrdx
!=====
subroutine solve2x2(A,b,dx)
implicit none
real(8) :: A(2,2),b(2),dx(2)
real(8) :: num1,num2,det
num1 = A(2,2)*b(1) - A(1,2)*b(2)
num2 = A(1,1)*b(2) - A(2,1)*b(1)
det   = A(1,1)*A(2,2)- A(1,2)*A(2,1)
if(det .eq. 0.0D0) stop 'solve2x2: det = 0'
dx(1) = num1/det
dx(2) = num2/det
end subroutine solve2x2

```

Η μεταγλώττιση του προγράμματος και η εκτέλεσή του γίνεται με τις εντολές:

```

> gfortran bifurcationPoints.f90 -o b
> echo 2 3.5 0.5 |./b
# Enter k,r0,x0:
# Period k= 2
# r0= 3.500000000000000 x0= 0.500000000000
1 x0= 0.4455758353187 r0= 3.38523275827 err= 6.35088E-002
2 x0= 0.4396562547624 r0= 3.45290970406 err= 3.39676E-002
3 x0= 0.4399593001407 r0= 3.44949859951 err= 1.71226E-003
4 x0= 0.4399601690333 r0= 3.44948974267 err= 4.44967E-006
5 x0= 0.4399601689937 r0= 3.44948974281 err= 7.22160E-011
> echo 2 3.5 0.85 |./b
.....
4 x0= 0.8499377795512 r0= 3.44948974275 err= 1.85082E-011
> echo 4 3.5 0.5 |./b
.....
5 x0= 0.5235947861540 r0= 3.54409035953 err= 1.86318E-011
> echo 4 3.5 0.35 |./b
.....
5 x0= 0.3632903374118 r0= 3.54409035955 err= 5.91653E-013

```

Παραπάνω δείχνουμε τα σημεία του 2-κύκλου και μερικά από τα ση-

μεία του 4-κύκλου. Επίσης μπορούμε να συγκρίνουμε την τιμή του  $r_c^{(3)} = 3.449490132$  που υπολογίζουμε με την αναμενόμενη  $r_c^{(3)} = 1 + \sqrt{6} \approx 3.449489742$ . Αφήνουμε ως άσκηση στον αναγνώστη να βελτιώσει την ακρίβεια μειώνοντας τα συστηματικά σφάλματα του υπολογισμού, καθώς και να πετύχει μεγαλύτερη ακρίβεια στην τιμή του  $r_c^{(4)}$ .

### 3.6 Εκθέτες Liapunov

Όπως είδαμε, όταν  $r > r_c \approx 3.56994567$  οι τροχιές της λογιστικής εξίσωσης παρουσιάζουν χαοτική συμπεριφορά. Αυτό κυρίως σημαίνει ότι οι τροχιές είναι ευαίσθητες στην επιλογή των αρχικών συνθηκών, έτσι ώστε τροχιές που ξεκινάνε από απειροστά κοντά αρχικά σημεία, σε μικρό αριθμό βημάτων διαφέρουν σημαντικά. Αυτό συνεπάγεται, επίσης, πως υπάρχει σύνολο από αρχικές συνθήκες που καλύπτουν πυκνά κάποια υποδιαστήματα του  $(0, 1)$  των οποίων οι τροχιές δεν πλησιάζουν οριακά κανένα κύκλο οποιουδήποτε μήκους.

Πιο συγκεκριμένα, έστω δύο τροχιές οι οποίες έχουν ως αρχικά σημεία τα  $x_0, \tilde{x}_0$  αντίστοιχα και  $\Delta x_0 = x_0 - \tilde{x}_0$ . Όταν για αρκετά μικρά  $n$  τα επόμενα σημεία  $x_n, \tilde{x}_n$  απέχουν απόσταση  $\Delta x_n = \tilde{x}_n - x_n$  που αυξάνει εκθετικά με το  $n$  (“χρόνο”), δηλαδή

$$\Delta x_n \sim \Delta x_0 e^{\lambda n}, \quad \lambda > 0, \quad (3.34)$$

αυτό αποτελεί ένδειξη χαοτικού συστήματος<sup>11</sup>. Ο εκθέτης  $\lambda$  λέγεται εκθέτης Liapunov. Για τον υπολογισμό του εκθέτη είναι χρήσιμο να χρησιμοποιήσουμε την ακόλουθη σχέση:

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \ln |f'(x_k)|. \quad (3.35)$$

Η ισχύς αυτής της σχέσης αποδεικνύεται θεωρώντας απειροστό  $\epsilon \equiv$

<sup>11</sup>Από μόνη της η ευαισθησία στις αρχικές συνθήκες δε συνεπάγεται αναγκαστικά χάος. Χρειάζεται να υπάρχει και τοπολογική ανάμιξη (topological mixing) και πυκνές περιοδικές τροχιές (density of periodic orbits). Τοπολογική ανάμιξη σημαίνει ότι οποιοδήποτε ανοικτό σύνολο του χώρου των φάσεων, αν εξελιχθεί αρκετά με το χρόνο, θα έχει επικάλυψη με οποιοδήποτε άλλο ανοικτό σύνολο. Πυκνότητα περιοδικών τροχιών σημαίνει πως οποιοδήποτε σημείο του χώρου των φάσεων πλησιάζεται αυθαίρετα κοντά από κάποια περιοδική τροχιά.

$|\Delta x_0|$  έτσι ώστε  $\lambda = \lim_{n \rightarrow \infty} \lim_{\epsilon \rightarrow 0} \frac{1}{n} \ln |\Delta x_n|/\epsilon$ . Έτσι, έχουμε

$$\begin{aligned}
 \tilde{x}_1 &= f(\tilde{x}_0) = f(x_0 + \epsilon) \approx f(x_0) + \epsilon f'(x_0) \\
 &= x_1 + \epsilon f'(x_0) \Rightarrow \\
 \frac{\Delta x_1}{\epsilon} &= \frac{\tilde{x}_1 - x_1}{\epsilon} \approx f'(x_0) \\
 \tilde{x}_2 &= f(\tilde{x}_1) = f(x_1 + \epsilon f'(x_0)) \approx f(x_1) + (\epsilon f'(x_0)) f'(x_1) \\
 &= x_2 + \epsilon f'(x_0) f'(x_1) \Rightarrow \\
 \frac{\Delta x_2}{\epsilon} &= \frac{\tilde{x}_2 - x_2}{\epsilon} \approx f'(x_0) f'(x_1) \\
 \tilde{x}_3 &= f(\tilde{x}_2) = f(x_2 + \epsilon f'(x_0) f'(x_1)) \approx f(x_2) + (\epsilon f'(x_0) f'(x_1)) f'(x_2) \\
 &= x_3 + \epsilon f'(x_0) f'(x_1) f'(x_2) \Rightarrow \\
 \frac{\Delta x_3}{\epsilon} &= \frac{\tilde{x}_3 - x_3}{\epsilon} \approx f'(x_0) f'(x_1) f'(x_2). \tag{3.36}
 \end{aligned}$$

Οπότε επαγωγικά δείχνουμε ότι  $|\Delta x_n|/\epsilon \approx f'(x_0) f'(x_1) f'(x_2) \dots f'(x_{n-1})$  και, παίρνοντας το λογάριθμο και τα όρια, αποδεικνύεται η (3.35).

Αρχικά, μπορούμε να υπολογίσουμε τους εκθέτες Liapunov χρησιμοποιώντας τον ορισμό (3.34). Μεταβάλλουμε κατάλληλα το πρόγραμμα `logistic.f90`, έτσι ώστε να υπολογίζει δύο τροχιές που αρχικά απέχουν απόσταση  $\epsilon = \text{epsilon}$ :

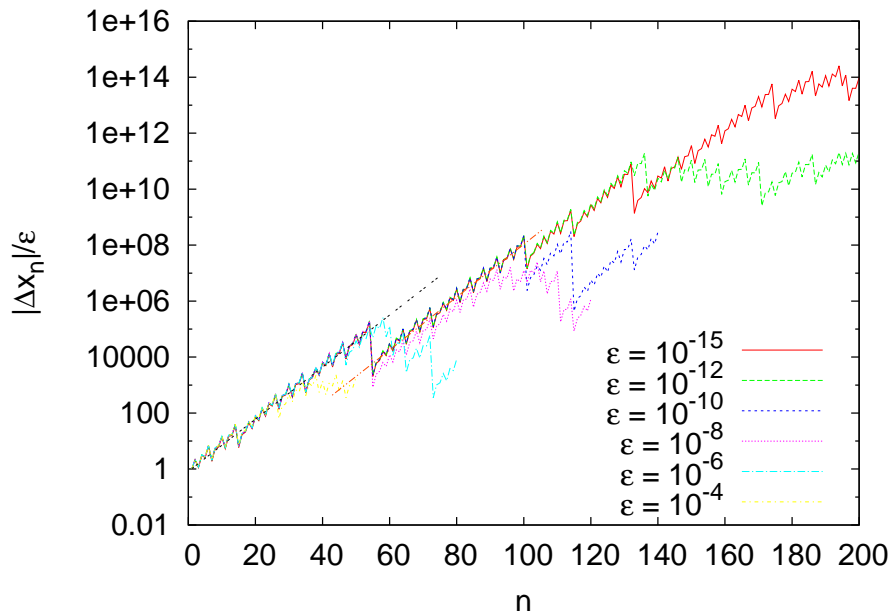
```

!=====
!Discrete Logistic Map:
!Two trajectories with close initial conditions.
!=====
program logistic_map
  implicit none
  integer :: NSTEPS, i
  real(8) :: r, x0, x1, x0t, x1t, epsilon

  ! ----- Input:
  print *, '# Enter NSTEPS, r, x0, epsilon:'
  read *, NSTEPS, r, x0, epsilon
  print *, '# NSTEPS = ', NSTEPS
  print *, '# r = ', r
  print *, '# x0 = ', x0
  print *, '# epsilon = ', epsilon

  x0t = x0 + epsilon
  ! ----- Initialize:

```



Σχήμα 3.7: Γραφική παράσταση της  $|\Delta x_n|/\epsilon$  για τη λογιστική εξίσωση με  $r = 3.6$ ,  $x_0 = 0.2$ . Παρατηρούμε τη σύγκλιση των καμπύλων στο όριο  $\epsilon \rightarrow 0$  και την προσεγγιστική εκθετική συμπεριφορά της γραφικής παράστασης στο όριο αυτό. Τα δύο ευθύγραμμα τμήματα είναι προσαρμογή στην εξίσωση (3.34) και δίνουν  $\lambda = 0.213(4)$  και  $\lambda = 0.217(6)$  αντίστοιχα.

```

open(unit=33,file='lia.dat')
write(33,*) 1,x0,x0t,ABS(x0t-x0)/epsilon
! ----- Calculate:
do i=2,NSTEPS
  x1 = r * x0 * (1.0D0-x0)
  x1t = r * x0t * (1.0D0-x0t)
  write(33,*)i,x1,x1t,ABS(x1t-x1)/epsilon
  x0 = x1; x0t = x1t
enddo
close(33)
end program logistic_map

```

Αφού το τρέξουμε βρίσκουμε στην τέταρτη στήλη του αρχείου lia.dat την τιμή  $|\Delta x_n|/\epsilon$ . Μπορούμε έτσι να φτιάξουμε καμπύλες όπως στο σχήμα 3.7 με τις εντολές:

```

> gfortran liapunov1.f90 -o l
> gnuplot
gnuplot> set logscale y

```

```
gnuplot> plot \
    "<echo 200 3.6 0.2 1e-15 1./1;cat lia.dat" u 1:4 w l
```

Στην τελευταία γραμμή κάνουμε γραφική παράσταση από το `stdout` της εντολής μέσα στα εισαγωγικά. Η πρώτη εντολή τρέχει το πρόγραμμα `./1` διαβάζοντας από το `stdin` τις παραμέτρους  $NSTEPS = 200$ ,  $r = 3.6$ ,  $x_0 = 0.2$  και  $\epsilon = 10^{-15}$ . Η δεύτερη τυπώνει στο `stdout` τα περιεχόμενα του αρχείου `lia.dat` το οποίο διαβάζεται από το `gnuplot` που κάνει τη γραφική παράσταση. Με την εντολή `set logscale y` στο `gnuplot`, κάνουμε λογαριθμική την κλίμακα μόνο στον  $y$ -άξονα. Έτσι μια εκθετική συνάρτηση παριστάνεται γραφικά από μία ευθεία. Στο σχήμα 3.7 παρατηρούμε τέτοια ευθεία διάταξη των σημείων  $|\Delta x_n|/\epsilon$  η οποία γίνεται ολοένα και καλύτερη όσο το  $\epsilon$  μικραίνει. Η κλίση των ευθυγράμμων αυτών τμημάτων μας δίνει τον εκθέτη  $\lambda$ . Επίσης, παρατηρούμε παράλληλα ευθύγραμμα τμήματα με την ίδια περίπου κλίση που αντιστοιχούν στο ίδιο  $\lambda$ . Η διάταξη των σημείων δεν είναι ακριβώς ευθεία οπότε ο υπολογισμός του  $\lambda$  ενέχει σφάλματα, όπως σημειώνουμε και στο σχήμα 3.7. Αλλάζοντας την αρχική συνθήκη παίρνουμε ελαφρά διαφορετική τιμή για το  $\lambda$  και έτσι μπορούμε να πάρουμε για την τιμή του  $\lambda$  τη μέση τιμή των μετρήσεων μας και να εκτιμήσουμε το σφάλμα ως το σφάλμα της μέσης τιμής. Αυτό αφήνεται για άσκηση στον αναγνώστη.

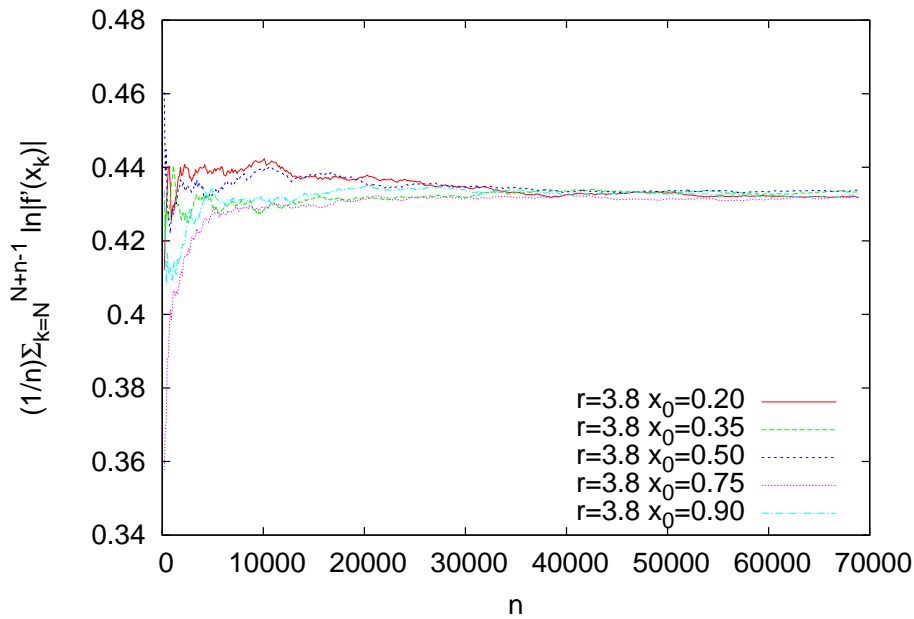
Ας πούμε περιληπτικά πώς μπορεί κανείς να υπολογίσει το  $\lambda$  με προσρμογή της εκθετικής συνάρτησης στα σημεία  $|\Delta x_n|/\epsilon$ . Επειδή  $|\Delta x_n|/\epsilon \sim C \exp(\lambda n) \Rightarrow \ln(|\Delta x_n|/\epsilon) = \lambda n + c$ , οπότε μπορούμε να κάνουμε προσρμογή σε ευθεία γραμμή. Αυτό γίνεται εύκολα με τη χρήση του `gnuplot`:

```
gnuplot> fit [5:53] a*x+b \
    "<echo 500 3.6 0.2 1e-15 1./1;cat lia.dat" \
    using 1:(log($4)) via a,b
gnuplot> replot exp(a*x+b)
```

Με την παραπάνω εντολή προσαρμόζουμε στη συνάρτηση  $a*x+b$  τα δεδομένα της 1ης και του λογαρίθμου της 4ης στήλης (`using 1:(log($4))`) του `stdout` της εντολής που παράγει τα δεδομένα όπως και στη γραφική παράσταση που κάναμε παραπάνω. Τα δεδομένα που επιλέγουμε είναι για  $5 \leq n \leq 53$  (`[5:53]`) και οι παράμετροι που προσαρμόζονται οι  $a, b$  (`via a,b`). Στη δεύτερη γραμμή προσθέτουμε τη γραφική παράσταση του εκθετικού της ευθείας που υπολογίζεται.

Θα χρησιμοποιήσουμε τώρα τη σχέση (3.35) για τον υπολογισμό του  $\lambda$ . Η σχέση αυτή είναι προσεγγιστικά σωστή, όταν (α) έχουμε ήδη μπει στη μόνιμη κατάσταση, και (β) στο όριο μεγάλου  $n$ . Για το λόγο αυτό





Σχήμα 3.8: Γραφική παράσταση του αθροίσματος  $(1/n) \sum_{k=N}^{N+n-1} \ln |f'(x_k)|$  ως συνάρτηση του  $n$  για τη λογιστική εξίσωση με  $r = 3.8$ ,  $N = 2000$  για διαφορετικές αρχικές συνθήκες  $x_0 = 0.20, 0.35, 0.50, 0.75, 0.90$ . Παρατηρούμε τη σύγκλιση των καμπύλων στο όριο  $n \rightarrow \infty$  η οποίες μας δίνουν  $\lambda = 0.4325(10)$ .

θα πρέπει να μελετήσουμε αν η σχέση συγκλίνει ικανοποιητικά, καθώς (α) “πετάξουμε” ένα αριθμό από NTRANS βήματα, (β) υπολογίσουμε το άθροισμα (3.35) με ολοένα αυξανόμενο NSTEPS=  $n$  και (γ) υπολογίσουμε το άθροισμα (3.35) ξεκινώντας τη λογιστική εξίσωση από διαφορετικό αρχικό σημείο  $x_0$ . Θα πρέπει να κάνουμε τη μελέτη αυτή προσεκτικά για κάθε τιμή του  $r$  που θα θεωρήσουμε, καθώς σε κάθε περίπτωση η συνεισφορά κάθε παράγοντα στη σύγκλιση του  $\lambda$  θα είναι διαφορετική: Σε περιοχές έντονης χαοτικής συμπεριφοράς (μεγάλο  $\lambda$ ) η σύγκλιση γίνεται με πιο αργό ρυθμό. Το πρόγραμμα θα το βρείτε στο αρχείο `liapunov2.f90`:

```
!=====
! Discrete Logistic Map:
! Liapunov exponent from sum_i ln|f'(x_i)|
! NTRANS: number of discarded iteration in order to discard
!          transient behaviour
! NSTEPS: number of terms in the sum
!=====
```

```

program logistic_map
  implicit none
  integer :: NTRANS, NSTEPS, i
  real(8) :: r, x0, x1, sum

  ! ----- Input:
  print *, '# Enter NTRANS, NSTEPS, r, x0:'
  read *, NTRANS, NSTEPS, r, x0
  print *, '# NTRANS = ', NTRANS
  print *, '# NSTEPS = ', NSTEPS
  print *, '# r = ', r
  print *, '# x0 = ', x0

  do i=1, NTRANS
    x1 = r * x0 * (1.0D0 - x0)
    x0 = x1
  enddo
  sum = log(ABS(r*(1.0D0 - 2.0D0*x0)))
  ! ----- Initialize:
  open(unit=33, file='lia.dat')
  write(33,*) 1, x0, sum
  ! ----- Calculate:
  do i=2, NSTEPS
    x1 = r * x0 * (1.0D0 - x0)
    sum = sum + log(ABS(r*(1.0D0 - 2.0D0*x1)))
    write(33,*) i, x1, sum/i
    x0 = x1
  enddo
  close(33)
end program logistic_map

```

Το πρόγραμμα, αφού εκτελέσει  $NTRANS$  βήματα  $x_{n+1} = f(x_n)$ , υπολογίζει  $NSTEPS$  φορές το άθροισμα των όρων  $\ln |f'(x_k)| = \ln |r(1 - 2x_k)|$  το οποίο και αποθηκεύει στη μεταβλητή `sum`. Σε κάθε βήμα τυπώνεται στο αρχείο `lia.dat` το άθροισμα `sum` διαιρεμένο με τον αριθμό των βημάτων `i`. Στο σχήμα 3.6 δείχνουμε τα αποτελέσματα των μετρήσεών μας για  $r = 3.8$ . Είναι σημείο όπου το σύστημα έχει ισχυρή χαοτική συμπεριφορά και για να βεβαιωθούμε για τη σύγκλιση του αθροίσματος πρέπει να υπολογίσουμε ένα μεγάλο αριθμό από όρους στο άθροισμα. Χρησιμοποιώντας  $NTRANS = 2000$  και  $NSTEPS \approx 70\,000$  πετυχαίνουμε ακρίβεια περίπου 0.2% με  $\lambda = 0.4325 \pm 0.0010 \equiv 0.4325(10)$ . Η κύρια συνεισφορά στο σφάλμα προέρχεται από τις διαφορετικές “διαδρομές” σύγκλισης για κάθε αρχικό σημείο που επιλέξαμε. Το σχήμα (3.6) μπορεί να γίνει με τις εντολές:

```
> gfortran liapunov2.f90 -o 1
```

```
> gnuplot
gnuplot> plot \
  "<echo 2000 70000 3.8 0.20 |./1;cat lia.dat" u 1:3 w l,\
  "<echo 2000 70000 3.8 0.35 |./1;cat lia.dat" u 1:3 w l,\
  "<echo 2000 70000 3.8 0.50 |./1;cat lia.dat" u 1:3 w l,\
  "<echo 2000 70000 3.8 0.75 |./1;cat lia.dat" u 1:3 w l,\
  "<echo 2000 70000 3.8 0.90 |./1;cat lia.dat" u 1:3 w l
```

Μέσα στην εντολή `plot` τρέχουμε το πρόγραμμα `./1` με παραμέτρους  $NTRANS = 2000$ ,  $NSTEPS = 70000$ ,  $r = 3.8$  και  $x_0 = 0.20, 0.35, 0.50, 0.75, 0.90$ . Στη συνέχεια, το `gnuplot` διαβάζει από το `stdout` της εντολής `cat lia.dat` τα περιεχόμενα των δεδομένων που έχουν καταχωρηθεί από το πρόγραμμα `./1`.

Για να εντοπίσουμε τις περιοχές χαοτικής συμπεριφοράς θα εξετάσουμε την εξάρτηση του εκθέτη `Liapunov` από την παράμετρο  $r$ . Επωφελούμενοι από την εμπειρία μας από τον προσεκτικό προσδιορισμό του  $\lambda$ , θα τρέξουμε το προηγούμενο πρόγραμμα για διαφορετικές τιμές του  $r$  έχοντας επιλέξει σταθερές παραμέτρους  $NTRANS = 2000$ ,  $NSTEPS = 60000$  και από δεδομένο αρχικό σημείο  $x_0 = 0.2$ . Ο υπολογισμός θα μας δώσει αποτελέσματα με ακρίβεια της τάξης του 1%. Για προσεκτικά υπολογισμένες τιμές του  $\lambda$  με προσδιορισμό του σφάλματος στη μέτρηση, θα πρέπει να ακολουθήσουμε τη μέθοδο που περιγράψαμε στα σχήματα 3.7 και 3.8. Το πρόγραμμα θα το βρείτε στο αρχείο `liapunov3.f90` και είναι απλή παραλλαγή του προηγούμενου προγράμματος, έτσι ώστε να εκτελείται για πολλές τιμές της παραμέτρου  $r$ .

```
!=====
! Discrete Logistic Map:
! Liapunov exponent from sum_i ln|f'(x_i)|
! Calculation for r in [rmin,rmax] with RSTEPS steps
! RSTEPS: values of r studied: r=rmin+(rmax-rmin)/RSTEPS
! NTRANS: number of discarded iteration in order to discard
!         transient behaviour
! NSTEPS: number of terms in the sum
! xstart: value of initial x0 for every r
!=====
program logistic_map
  implicit none
  real(8),parameter :: rmin  = 2.5D0
  real(8),parameter :: rmax  = 4.0D0
  real(8),parameter :: xstart = 0.2D0
  integer,parameter :: RSTEPS = 1000
  integer,parameter :: NSTEPS = 60000
  integer,parameter :: NTRANS = 2000
  integer :: i,ir
```

```

real(8) :: r,x0,x1,sum,dr

open(unit=33,file='lia.dat')
dr = (rmax-rmin)/(RSTEPS-1)
do ir=0,RSTEPS-1
  r = rmin+ir*dr
  x0= xstart
  do i=1,NTRANS
    x1 = r * x0 * (1.0D0-x0 )
    x0 = x1
  enddo
  sum = log(ABS(r*(1.0D0-2.0D0*x0)))
! ----- Calculate:
  do i=2,NSTEPS
    x1 = r * x0 * (1.0D0-x0 )
    sum = sum + log(ABS(r*(1.0D0-2.0D0*x1)))
    x0 = x1
  enddo
  write(33,*)r,sum/NSTEPS
enddo !do ir=0,RSTEPS-1
close(33)
end program logistic_map

```

Το πρόγραμμα μεταγλωττίζεται και εκτελείται με τις εντολές

```

> gfortran liapunov3.f90 -o l
> ./l &

```

όπου η χρήση του & γίνεται, έτσι ώστε το πρόγραμμα ./l να εκτελεστεί στο υπόβαθρο (background).

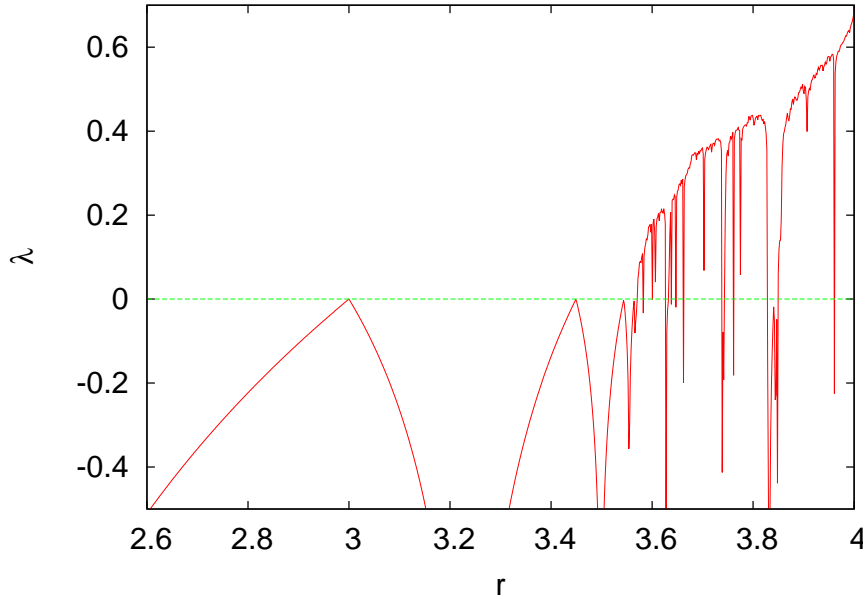
Τα δεδομένα αποθηκεύονται στο αρχείο lia.dat και μπορούμε να φτιάξουμε και να μελετήσουμε το σχήμα 3.7 με το gnuplot:

```

gnuplot> plot "lia.dat" with lines notitle ,0 notitle

```

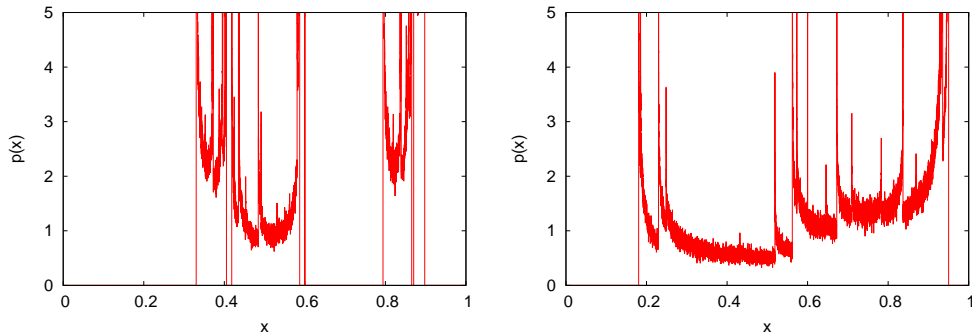
Το σχήμα 3.9 μπορούμε τώρα να το συγκρίνουμε με το διάγραμμα διακλάδωσης του σχήματος 3.4. Οι περιοχές  $\lambda < 0$  αντιστοιχούν στις περιοχές με σταθερούς  $k$ -κύκλους, δηλαδή σταθερές περιοδικές τροχιές. Οι περιοχές  $\lambda > 0$  αντιστοιχούν στις χαοτικές περιοχές, τις περιοχές όπου εμφανίζεται ισχυρό χάος (strong chaos). Οι περιοχές αυτές χωρίζονται από κρίσιμα σημεία με  $\lambda = 0$ . Στα σημεία αυτά έχουμε ασθενές χάος (weak chaos) όπου οι γειτονικές τροχιές αποκλίνουν μεταξύ τους όχι εκθετικά, αλλά σαν μια δύναμη του  $n$ . Έχουμε δηλαδή  $|\Delta x_n| \sim |\Delta x_0|n^\omega$ , όπου  $\omega = 1/(1-q)$  είναι ένας θετικός εκθέτης που πρέπει να υπολογιστεί. Στη βιβλιογραφία χρησιμοποιείται συνήθως ο εκθέτης  $q$  και η



Σχήμα 3.9: Ο εκθέτης Liapunov  $\lambda$  της λογιστικής εξίσωσης ο οποίος υπολογίζεται από τη σχέση (3.35). Παρατηρήστε την εμφάνιση χαοτικής συμπεριφοράς στις περιοχές όπου  $\lambda > 0$  καθώς και την εμφάνιση των “παραθύρων”  $\lambda < 0$  με σταθερούς  $k$ -κύκλους όπως είχαμε δει και στο διάγραμμα διακλάδωσης, σχήμα 3.4. Στα κρίσιμα σημεία με  $\lambda = 0$  έχουμε την εκδήλωση χάους (onset of chaos, edge of chaos) με ασθενή χαοτική συμπεριφορά (weak chaos,  $|\Delta x_n| \sim |\Delta x_0|n^\omega$ ) όπου συμβαίνει μετάβαση από περιοχή με σταθερούς  $k$ -κύκλους σε περιοχή με ισχυρή χαοτική συμπεριφορά (strong chaos). Παρατηρούμε την εκδήλωση χάους όταν  $r = r_c \approx 3.5699$  όπου  $\lambda = 0$  για πρώτη φορά (για μικρότερα  $r$  η γραφική παράσταση φαίνεται να αγγίζει τη γραμμή  $\lambda = 0$ , αλλά στην πραγματικότητα το  $\lambda$  παίρνει πολύ μικρές κατά απόλυτη τιμή αρνητικές τιμές).

εκθετική συμπεριφορά (ισχυρό χάος) παίρνεται στο όριο  $q \rightarrow 1$ . Παρατηρούμε τα “διαλείμματα χάους” που είχαμε συζητήσει, όταν μελετούσαμε το διάγραμμα διακλάδωσης. Οι κρίσιμες τιμές του  $r$  στις οποίες συμβαίνουν αυτές οι μεταβάσεις μπορούν να υπολογιστούν με ακρίβεια εντοπίζοντας τον υπολογισμό στο κατάλληλο διάστημα τιμών του  $r$  σε αρκετά μικρή γειτονιά του κρίσιμου σημείου. Με το παραπάνω πρόγραμμα, αυτό γίνεται εύκολα αλλάζοντας τις τιμές των παραμέτρων  $r_{\min}$  και  $r_{\max}$ .

Ένας άλλος τρόπος να μελετήσουμε τις χαοτικές ιδιότητες των τροχιών της λογιστικής εξίσωσης είναι να υπολογίσουμε την κατανομή των τιμών του  $x$  στο διάστημα  $(0, 1)$ . Στις περιοδικές τροχιές, αφού περάσουμε τη μεταβατική περίοδο, η κατανομή θα συγκεντρώνεται στα σημεία των  $k$ -κύκλων, ενώ στις χαοτικές, μη περιοδικές, τροχιές θα κα-



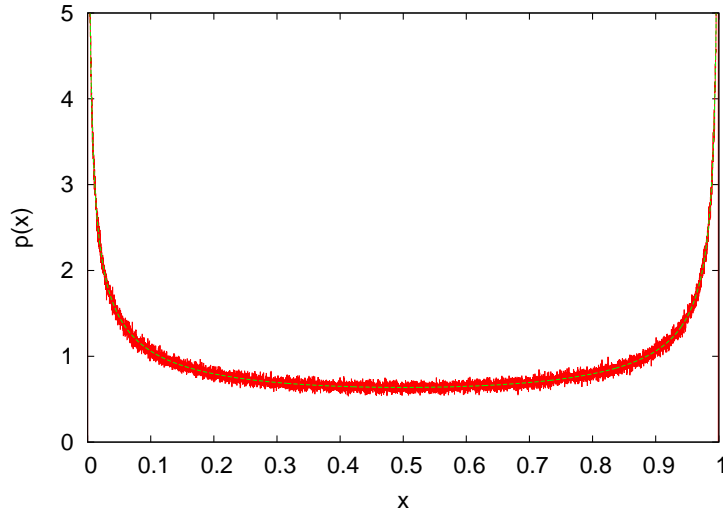
Σχήμα 3.10: Κατανομές των τιμών  $p(x)$  του  $x$  στη λογιστική εξίσωση για  $r = 3.59$  (αριστερά) και  $3.8$  (δεξιά). Αριστερά η χαοτική συμπεριφορά φαίνεται να είναι πιο ασθενής από δεξιά, κάτι το οποίο θα αντικατοπτριστεί στη χαμηλότερη τιμή της εντροπίας. Υπάρχουν ολόκληρες περιοχές του  $x$  με  $p(x) = 0$ . Η κατανομή αυτή είναι πολύ δύσκολο να διακριθεί από μια πραγματικά τυχαία κατανομή πιθανότητας.

τανέμεται πάνω σε υποδιαστήματα του  $(0, 1)$ . Η πυκνότητα πιθανότητας  $p(x)$  να βρούμε μια τιμή  $x$  για δεδομένο  $r$  είναι ανεξάρτητη του αρχικού σημείου της τροχιάς. Αντίστροφα, αν μας δοθεί ένας μεγάλος αριθμός από σημεία που παράγει η λογιστική εξίσωση, θα ήταν πρακτικά αδύνατο να καταλάβουμε πως προέρχονται από ένα ντετερμινιστικό κανόνα. Για το λόγο αυτό, χαοτικά συστήματα μπορούν να χρησιμοποιηθούν για την παραγωγή ψευδοτυχαίων αριθμών, όπως θα δούμε στο κεφάλαιο 11. Για να μετρήσουμε την “τυχειότητα” της κατανομής, μπορούμε να χρησιμοποιήσουμε την εντροπία η οποία είναι ένα μέτρο της αταξίας ενός συστήματος. Όπως θα εξηγήσουμε λεπτομερέστερα στο κεφάλαιο 12, αυτή δίνεται από τη σχέση

$$S = - \sum_k p_k \ln p_k, \quad (3.37)$$

όπου  $p_k$  είναι η πιθανότητα εμφάνισης της κατάστασης  $k$ . Στην περίπτωση μας, θα υπολογίσουμε την κατανομή  $p_k$  χωρίζοντας το διάστημα  $(0, 1)$  σε υποδιαστήματα πλάτους  $\Delta x$ . Για δεδομένο  $r$ , θα πάρουμε ένα μεγάλο αριθμό  $M$  από μετρήσεις των τιμών  $x_n$  της λογιστικής εξίσωσης και θα υπολογίσουμε το ιστόγραμμα  $h_k$  της κατανομής τους στα διαστήματα  $(x_k, x_k + \Delta x)$  που επιλέξαμε. Η πυκνότητα κατανομής αντιστοιχεί στο όριο της ποσότητας  $p_k = h_k / (M \Delta x)$  για μεγάλα  $M$  και μικρό  $\Delta x$  (μεγάλο  $N$ ). Πράγματι, τότε  $\sum_{k=1}^N p_k \Delta x = 1$  το οποίο συγκλίνει στο  $\int_0^1 p(x) dx = 1$ . Θα ορίσουμε τότε  $S = - \sum_{k=1}^N p_k \ln p_k \Delta x$ .

Το πρόγραμμα που θα γράψουμε θα υπολογίζει για τις επιλεγμένες τιμές του  $r$  την κατανομή  $p_k$  και από αυτή την εντροπία  $S$  σύμφωνα



Σχήμα 3.11: Κατανομές των τιμών  $p(x)$  του  $x$  στη λογιστική εξίσωση για  $r = 4$ . Έχουμε ισχυρή χαοτική συμπεριφορά και η  $p(x)$  είναι μη-μηδενική σε ολόκληρο το διάστημα  $(0,1)$ . Ευνοούνται οι τιμές του  $x$  που βρίσκονται στα άκρα του διαστήματος και η εντροπία είναι μεγάλη. Η καμπύλη είναι η γραφική παράσταση της γνωστής [29] για  $r = 4$  κατανομής  $p(x) = \pi^{-1} x^{-1/2} (1-x)^{-1/2}$  (κατανομή beta για  $a = 1/2$ ,  $b = 1/2$ ).

με την (3.37). Αυτό θα προκύψει μεταβάλλοντας το πρόγραμμα του αρχείου `liapunov3.f90` προσθέτοντας την παράμετρο `NHIST` που είναι ο αριθμός των διαστημάτων  $N$  για τα ιστογράμματα και αποθηκεύοντας τις πυκνότητες πιθανότητας στο array `p(NHIST)`. Το πρόγραμμα θα το βρείτε στο αρχείο `entropy.f90`:

```
!=====
! Discrete Logistic Map:
! Entropy calculation from  $S = -\sum_i p_i \ln p_i$ 
! Calculation for  $r$  in  $[rmin, rmax]$  with RSTEPS steps
! RSTEPS: values of  $r$  studied:  $r = rmin + (rmax - rmin) / RSTEPS$ 
! NHIST : number of histogram bins for calculation of  $p_i$ 
! NSTEPS: number of values of  $x$  in the histograms
! NTRANS: number of discarded iteration in order to discard
!         transient behaviour
! xstart: value of initial  $x_0$  for every  $r$ 
!=====
program logistic_map
  implicit none
  real(8), parameter :: rmin  = 2.5D0
  real(8), parameter :: rmax  = 4.0D0
  real(8), parameter :: xstart = 0.2D0
```

```

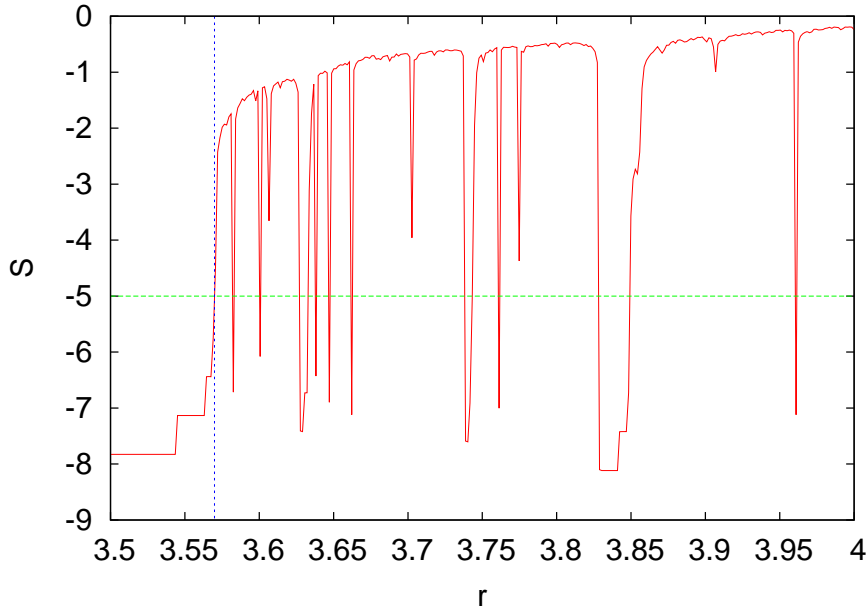
integer ,parameter :: RSTEPS = 1000
integer ,parameter :: NHIST  = 10000
integer ,parameter :: NTRANS = 2000
integer ,parameter :: NSTEPS = 5000000
real(8) ,parameter :: xmin=0.0D0 ,xmax=1.0D0
integer :: i,ir,isum,n
real(8) :: r,x0,x1,sum,dr,dx
real(8) :: p(NHIST),S

open(unit=33,file='entropy.dat')
p = 0.0D0
dr = (rmax-rmin)/(RSTEPS-1)
dx = (xmax-xmin)/(NHIST -1)
do ir=0,RSTEPS-1
  r = rmin+ir*dr
  x0= xstart
  do i=1,NTRANS
    x1 = r * x0 * (1.0D0-x0 )
    x0 = x1
  enddo
!make histogram:
  n=INT(x0/dx)+1;p(n)=p(n)+1.0D0
  do i=2,NSTEPS
    x1 = r * x0 * (1.0D0-x0 )
    n = INT(x1/dx)+1
    p(n)=p(n)+1.0D0
    x0 = x1
  enddo
!p(k) is now histogram of x-values.
!Normalize so that sum_k p(k)*dx=1
!to get probability distribution:
  p = p/NSTEPS/dx
!sum all non zero terms: p(n)*log(p(n))*dx
  S = -SUM(p*log(p),MASK=p.gt.0.0D0)*dx
  write(33,*)r,S
enddo !do ir=0,RSTEPS-1
close(33)
!print the last probability distribution:
open(unit=34,file='entropy_hist.dat')
do n=1,NHIST
  x0 = xmin +(n-1)*dx + 0.5D0*dx
  write(34,*) r,x0,p(n)
enddo
close(34)
end program logistic_map

```

Για τον υπολογισμό των κατανομών και της εντροπίας θα πρέπει να επιλέξουμε τις παραμέτρους με τις οποίες θα ελέγξουμε τα συστηματικά σφάλματα στους υπολογισμούς μας. Η παράμετρος NTRANS θα





Σχήμα 3.12: Η εντροπία  $S = -\sum_k p_k \ln p_k \Delta x$  για τη λογιστική εξίσωση ως συνάρτηση του  $r$ . Η κάθετη γραμμή είναι η  $r_c \approx 3.56994567$  που σηματοδοτεί την αρχή του χάους και η οριζόντια η αντίστοιχη εντροπία. Παρατηρούμε αριστερά τη χαμηλή εντροπία των αρχικών  $2^n$  ευσταθών κύκλων η οποία αυξάνεται σε κάθε διακλάδωση. Η εντροπία είναι μεγάλη στις περιοχές του χάους με απότομη ελάττωση στα “διαλείμματα” χάους. Φαίνεται καθαρά το διάλειμμα του 3-κύκλου για  $r = 1 + \sqrt{8} \approx 3.8284$  και οι ακόλουθες διακλαδώσεις του που είδαμε και στα διαγράμματα διακλάδωσης (σχήμα 3.4) και εκθέτη Liapunov (σχήμα 3.9).

πρέπει να είναι αρκετά μεγάλη, έτσι ώστε η μεταβατική περίοδος να μη “μολύνει” τα αποτελέσματα. Οι μετρήσεις μας θα πρέπει να ελεγχθούν ως προς τη σταθερότητά τους μεταβάλλοντας την τιμή της. Το ίδιο και η τιμή της  $x_{start}$  που δίνει το αρχικό σημείο για τη λογιστική εξίσωση για κάθε  $r$ . Η NHIST ελέγχει τη διαμέριση του διαστήματος  $(0, 1)$  και το μέγεθος του  $\Delta x$  και θα πρέπει να είναι αρκετά μεγάλη. Η NSTEPS είναι ο αριθμός των “μετρήσεων” για κάθε τιμή του  $r$  και θα πρέπει να είναι αρκετός για να μειώνεται ο “θόρυβος” στην τιμή του  $p_k$ . Εννοείται ότι όσο μικρότερο είναι το  $\Delta x$  τόσο μεγαλύτερο πρέπει να είναι το NSTEPS. Με κατάλληλες επιλογές παίρνουμε τις κατανομές  $p_k$  όπως φαίνεται στα σχήματα 3.10 και 3.11 για  $r = 3.59, 3.58$  και 4. Παρατηρούμε ότι ισχυρότερο χάος συνδέεται με πλατύτερη κατανομή των τιμών του  $x$ .

Ο υπολογισμός της εντροπίας φαίνεται στο σχήμα 3.12. Οι περιοδικές τροχιές έχουν σχετικά χαμηλή εντροπία ενώ οι χαοτικές υψηλότερη.

Η εντροπία αυξάνει ξαφνικά με την έναρξη του χάους για  $r = r_c$  και αυξάνει καθώς το χάος γίνεται ισχυρότερο. Στα “διαλείμματα” χάους η εντροπία πέφτει απότομα και δραστικά. Μπορούμε να διακρίνουμε πως τα διαστήματα αυτά είναι τα ίδια με αυτά που είδαμε στο διάγραμμα διακλάδωσης (σχήμα 3.4) και στον εκθέτη Liapunov (σχήμα 3.9). Η εντροπία αυξάνεται μέχρι την τιμή  $r = 4$ , αλλά αυτό δε γίνεται ομαλά. Μεγεθύνοντας περιοχές του διαγράμματος βλέπουμε άπειρο αριθμό από απότομες πτώσεις της εντροπίας σε διαστήματα του  $r$  που είναι διαρκώς μικρότερα.

## 3.7 Ασκήσεις

Πολλά από τα προγράμματα που ζητούνται στις ασκήσεις, θα τα βρείτε μέσα στον υποκατάλογο Problems του συνοδευτικού λογισμικού του κεφαλαίου.

3.1 Να δείξετε ότι οι τροχιές της λογιστικής απεικόνισης για  $r < 1$  μειώνονται εκθετικά με το  $n$  για αρκετά μεγάλο  $n$ . Να το επιβεβαιώσετε:

- (α') Επιλέξτε  $r = 0.5$  και κάνετε τις γραφικές παραστάσεις των τροχιών για  $x_0 = 0.1 - 0.9$  με βήμα 0.1 για  $n = 1, \dots, 1000$ . Στις γραφικές παραστάσεις να πάρετε τον άξονα  $y$  σε λογαριθμική κλίμακα. Τι σχήμα έχουν οι καμπύλες που παίρνετε για αρκετά μεγάλο  $n$ ;
- (β') Προσαρμόστε τα σημεία  $x_n$  για  $n > 20$  στη συνάρτηση  $ce^{-ax}$  και υπολογίστε τις σταθερές  $a$  και  $c$ . Πώς εξαρτώνται αυτές από το αρχικό σημείο  $x_0$ ; Με το gnuplot αυτό μπορεί να γίνει λ.χ. με τις εντολές:

```
gnuplot> !gfortran logistic.f90 -o l
gnuplot> a=0.7;c=0.4;
gnuplot> fit [10:] c*exp(-a*x) \
"<echo 1000 0.5 0.5|./l;cat log.dat" via a,c
gnuplot> plot c*exp(-a*x),\
"<echo 1000 0.5 0.5|./l;cat log.dat" w l
```

Παραπάνω θέσαμε NSTEPS = 1000,  $r = 0.5$ ,  $x_0 = 0.5$ . Θέτοντας τα όρια [10:] στην εντολή fit, προσαρμόζουμε μόνο στα σημεία  $x_n \geq 10$ , αποφεύγοντας έτσι τη μεταβατική περίοδο και την απόκλιση από την εκθετική συμπεριφορά για μικρά  $n$ .

- (γ') Επαναλάβετε για  $r = 0.3 - 0.9$  με βήμα 0.1, καθώς και για  $r = 0.99, 0.999$ . Προσοχή, καθώς το  $r$  πλησιάζει τη μονάδα θα χρειαστεί να πετάξετε περισσότερα αρχικά σημεία για μια καλή προσαρμογή και πιθανώς να αυξήσετε το NSTEPS. Ελέγχετε πάντα γραφικά αν η εκθετική συνάρτηση που υπολογίζετε ταιριάζει με τα σημεία  $x_n$  για μεγάλα  $n$ . Φτιάξτε ένα πίνακα με τον εκθέτη  $a$  ως συνάρτηση του  $r$ .

Οι λύσεις της εξίσωσης (3.3) είναι  $e^{(r-1)x}$ . Τι συμπεραίνετε από τις τιμές που υπολογίσατε στον πίνακα που φτιάξατε;

- 3.2 Μελετήστε τη λογιστική απεικόνιση για  $r = 2$ . Για NSTEPS=100, υπολογίστε τις τροχιές που παίρνετε για  $x_0=0.2, 0.3, 0.5, 0.7, 0.9$  και κάντε τις γραφικές τους παραστάσεις στο ίδιο γράφημα. Υπολογίστε το σταθερό σημείο  $x_2^*$  και συγκρίνετε με την αναμενόμενη τιμή  $1 - 1/r$ . Επαναλάβετε για  $x_0 = 10^{-\alpha}$  για  $\alpha = -1, -2, -5, -10, -20, -25$ . Τι συμπεραίνετε για το σημείο  $x_1^* = 0$ ;
- 3.3 Μελετήστε τη λογιστική απεικόνιση για  $r = 2.9, 2.99, 2.999$ . Υπολογίστε το σταθερό σημείο  $x_2^*$  και συγκρίνετε με την αναμενόμενη τιμή  $1 - 1/r$ . Πόσο μεγάλο NSTEPS πρέπει να πάρετε κάθε φορά. Μπορείτε να επιλέξετε  $x_0=0.3$ .
- 3.4 Μελετήστε τη λογιστική απεικόνιση για  $r = 3.2$ . Πάρτε  $x_0=0.3, 0.5, 0.9$  και NSTEPS=300 και κάντε τις γραφικές παραστάσεις των τροχιών που προκύπτουν. Υπολογίστε τα σταθερά σημεία  $x_3^*$  και  $x_4^*$  με την εντολή `tail log.dat`. Αυξήστε το NSTEPS και επαναλάβετε για να βεβαιωθείτε πως η τροχιά έχει συγκλίνει στον 2-κύκλο. Συγκρίνετε τις τιμές τους με τη σχέση (3.18). Για  $x_0=0.3$  κάντε τη γραφική παράσταση και με τους εξής τρόπους:

```
gnuplot> plot \
" <echo 300 3.2 0.3 | ./1;awk 'NR%2==0' log.dat" w l
gnuplot> replot \
" <echo 300 3.2 0.3 | ./1;awk 'NR%2==1' log.dat" w l
```

Τι προκύπτει;

- 3.5 Επαναλάβετε την προηγούμενη άσκηση για  $r = 3.4494$ . Πόσο μεγάλο NSTEPS πρέπει να πάρετε, ώστε να υπολογίσετε τα  $x_3^*$  και  $x_4^*$  με ακρίβεια 6 δεκαδικών ψηφίων;
- 3.6 Επαναλάβετε την προηγούμενη άσκηση για  $r = 3.5$  και 3.55. Επιλέξτε NSTEPS=1000,  $x_0=0.5$ . Δείξτε ότι οι τροχιές ακολουθούν 4-κύκλο και 8-κύκλο αντίστοιχα. Υπολογίστε τα σημεία  $x_5^*-x_8^*$  και  $x_9^*-x_{16}^*$ .
- 3.7 Κάνετε τη γραφική παράσταση των συναρτήσεων  $f(x)$ ,  $f^{(2)}(x)$ ,  $f^{(4)}(x)$ ,  $x$  για τη λογιστική απεικόνιση για δεδομένο  $r$  στο ίδιο σχήμα. Χρησιμοποιείστε τις εντολές:

```
gnuplot> set samples 1000
gnuplot> f(x) = r*x*(1-x)
gnuplot> r=1; plot [0:1] x, f(x), f(f(x)), f(f(f(x)))
```

Η εντολή  $r=1$  καθορίζει την τιμή του  $r$ . Πάρτε διαδοχικά  $r = 2.5, 3, 3.2, 1 + \sqrt{6}, 3.5$ . Από τις τομές των καμπύλων με τη διαγώνιο, προσδιορίστε τα σταθερά σημεία και τα σημεία των  $k$ -κύκλων.

- 3.8 Να φτιάξετε τα cobweb plots των σχημάτων 3.2 και 3.4 για  $r = 2.8, 3.3$  και 3.5. Να επαναλάβετε απορρίπτοντας διαδοχικά από το σχήμα όλο και περισσότερα από τα πρώτα σημεία των τροχιών μέχρι να προκύψουν μόνο τα σημεία των  $k$ -κύκλων. Κάνετε το ίδιο για  $r = 3.55$ .
- 3.9 Να φτιάξετε και τα δύο διαγράμματα διακλάδωσης του σχήματος 3.4.
- 3.10 Να φτιάξετε το διάγραμμα διακλάδωσης για  $3.840 < r < 3.851$  και για  $0.458 < x < 0.523$ . Με μεγέθυνσή της κατάλληλης περιοχής του διαγράμματος, προσδιορίστε τα τέσσερα πρώτα σημεία διακλάδωσης με ακρίβεια 5 σημαντικών ψηφίων. Να πάρετε  $NTRANS=15000$ .
- 3.11 Να φτιάξετε το διάγραμμα διακλάδωσης για  $2.9 < r < 3.57$ . Υπολογίστε γραφικά τα σημεία διακλάδωσης  $r_c^{(n)}$  για  $n = 2, 3, 4, 5, 6, 7, 8$ . Βεβαιωθείτε ότι τα αποτελέσματά σας είναι σταθερά ως προς τη μεταβολή των παραμέτρων  $NTRANS$ ,  $NSTEPS$  καθώς και από την επιλογή του κλάδου που επιλέγετε να μεγεθύνετε. Από τις τιμές των  $r_c^{(n)}$  που γνωρίζετε για  $n = 2, 3$ , και από την ευαισθησία των τιμών από τις παραπάνω επιλογές, εκτιμήστε την ακρίβεια που πετυχαίνετε με τη γραφική αυτή μέθοδο. Υπολογίστε τους λόγους  $(r_c^{(n)} - r_c^{(n-1)}) / (r_c^{(n+1)} - r_c^{(n)})$  και συγκρίνετε τα αποτελέσματά σας με τη σχέση (3.20).
- 3.12 Επιλέξτε τιμές του  $\rho$  στην (3.24), έτσι ώστε να έχετε μόνο ένα ενεργειακό επίπεδο, για το οποίο και να προσδιορίσετε την τιμή του  $\epsilon$ . Πότε έχουμε τρία ενεργειακά επίπεδα;
- 3.13 Δίνεται το πολυώνυμο  $g(x) = x^3 - 2x^2 - 11x + 12$ . Να βρείτε τις ρίζες του πολυωνύμου που παίρνετε με τη μέθοδο Newton-Raphson αν επιλέξετε  $x_0 = 2.35287527, 2.35284172, 2.35283735, 2.352836327, 2.352836323$ . Τι συμπεραίνετε σχετικά με τις περιοχές ελκυσμού των ριζών του πολυωνύμου; Αφού κάνετε τη γραφική παράσταση του πολυωνύμου, δοκιμάστε και άλλα αρχικά σημεία που θα σας δώσουν κάθε μία από τις ρίζες.
- 3.14 Υπολογίστε με τη μέθοδο Newton-Raphson τα σημεία του 4-κύκλου  $x_5^*, \dots, x_8^*$  της λογιστικής απεικόνισης. Χρησιμοποιήστε το διάγραμμα

$n$	$r_c^{(n)}$	$n$	$r_c^{(n)}$
2	3.0000000000	10	3.56994317604
3	3.4494897429	11	3.569945137342
4	3.544090360	12	3.5699455573912
5	3.564407266	13	3.569945647353
6	3.5687594195	14	3.5699456666199
7	3.5696916098	15	3.5699456707464
8	3.56989125938	16	3.56994567163008
9	3.56993401837	17	3.5699456718193
$r_c = 3.56994567 \dots$			

Πίνακας 3.1: Οι τιμές  $r_c^{(n)}$  για τη λογιστική απεικόνιση όπως υπολογίζονται στην άσκηση 17. Την τιμή  $r_c^{(\infty)} \equiv r_c$  την παίρνουμε από τη βιβλιογραφία.

διακλάδωσης μεγεθύνοντας τις κατάλληλες περιοχές έτσι ώστε να επιλέξετε κατάλληλα αρχικά σημεία για τις επαναλήψεις της μεθόδου. Να ελέγξετε αν οι τιμές του  $r_c^{(4)}$  που θα υπολογιστούν είναι ίδιες για κάθε  $x_\alpha^*$ . Ρυθμίστε τις παραμέτρους στο πρόγραμμα, έτσι ώστε να βελτιώσετε την ακρίβεια στις μετρήσεις σας.

3.15 Να επαναλάβετε την προηγούμενη άσκηση για τα σημεία του 8-κύκλου  $x_9^*, \dots, x_{16}^*$  και την τιμή  $r_c^{(5)}$  της λογιστικής απεικόνισης.

3.16 Να επαναλάβετε την προηγούμενη άσκηση για τα σημεία του 16-κύκλου  $x_{17}^*, \dots, x_{32}^*$  και την τιμή  $r_c^{(6)}$  της λογιστικής απεικόνισης.

3.17 Να υπολογίσετε τα κρίσιμα σημεία  $r_c^{(n)}$  για  $n = 3, \dots, 17$  της λογιστικής απεικόνισης με τη μέθοδο Newton-Raphson. Για να το κάνετε αυτό θα πρέπει να εντοπίσετε προσεκτικά από το διάγραμμα διακλάδωσης τα σημεία διακλάδωσης και να επιλέξετε τα αρχικά σημεία στη μέθοδο Newton-Raphson από τα σημεία που θα υπολογίσετε γραφικά. Θα πρέπει το πρόγραμμα `bifurcationPoints.f90` να διαβάζει τις παραμέτρους `eps`, `epsx`, `epsr` από το `stdin`, ώστε να μπορέσετε να τις ρυθμίζετε, καθώς αυξάνεται το  $n$ . Αν αυτές οι παράμετροι είναι πολύ μικρές, θα έχετε αστάθεια, ενώ αν είναι μεγάλες, θα έχετε συστηματικά σφάλματα. Με αυτόν τον τρόπο προσπαθήστε να αναπαράγετε τον πίνακα 3.1

3.18 Από τα αποτελέσματα που δίνονται στον πίνακα 3.1 υπολογίστε τους λόγους  $\Delta r_c^{(n)} / \Delta r_c^{(n+1)}$  που δίνονται στη σχέση 3.20 και

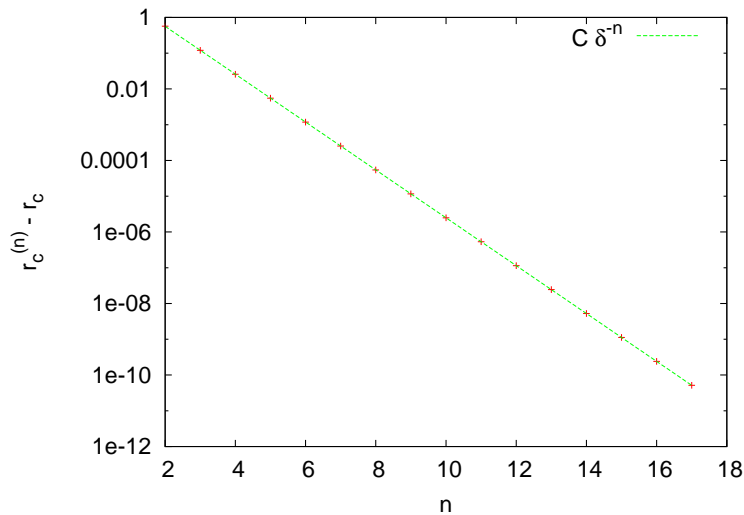
από αυτούς δείτε με πόση ακρίβεια μπορείτε να υπολογίσετε τον αριθμό του Feigenbaum.

- 3.19 Εκτιμήστε τον αριθμό του Feigenbaum  $\delta$  και την κρίσιμη τιμή  $r_c$  υποθέτοντας από τη σχέση (3.20) ότι για αρκετά μεγάλο  $n$   $r_c^{(n)} \approx r_c - C\delta^{-n}$ . Προσαρμόστε τα δεδομένα του πίνακα 3.1 στη συνάρτηση αυτή και υπολογίστε έτσι τα  $\delta$  και  $r_c$ . Στο σχήμα 3.13 επιβεβαιώνεται η υπόθεση μας και φαίνεται η εκθετική σύγκλιση των  $r_c^{(n)}$  στην τιμή  $r_c$ . Να φτιάξετε το ίδιο σχήμα με τις παραμέτρους που θα υπολογίσετε.

Υπόδειξη: Μπορείτε από το gnuplot να δώσετε τις εντολές:

```
gnuplot> nmin=2;nmax=17
gnuplot> r(x)= rc-c*d**(-x)
gnuplot> fit [nmin:nmax] r(x) "rcrit" u 1:2 via rc,c,d
gnuplot> plot "rcrit", r(x)
gnuplot> print rc,d
```

όπου στο αρχείο rcrit έχετε αποθηκεύσει τις τιμές του πίνακα 3.1. Μεταβάλλετε τις τιμές των nmin, nmax και επαναλάβετε μέχρι να πάρετε σταθερές τιμές και ικανοποιητική προσαρμογή στα δεδομένα.



Σχήμα 3.13: Επιβεβαίωση της σχέσης  $r_c^{(n)} \approx r_c - C\delta^{-n}$  που εξετάζεται στην άσκηση 17. Οι τιμές των παραμέτρων στο σχήμα είναι περίπου  $r_c = 3.5699457$ ,  $\delta = 4.669196$  και  $C = 12.292$ .

3.20 Να υπολογίσετε τα 3 πρώτα σημεία διακλάδωσης μετά την εμφάνιση του 3-κύκλου για  $r = 1 + \sqrt{8}$  με τη μέθοδο Newton-Raphson. Να επιλέξετε ένα σημείο διακλάδωσης του 3-κύκλου, ένα του 6-κύκλου και ένα του 12-κύκλου και να μεγεθύνετε το διάγραμμα διακλάδωσης στην περιοχή τους.

3.21 Θεωρήστε την απεικόνιση που περιγράφει την εξέλιξη ενός πληθυσμού

$$x_{n+1} = p(x_n) = x_n e^{r(1-x_n)}. \quad (3.38)$$

- (α') Να κάνετε τη γραφική παράσταση των συναρτήσεων  $x$ ,  $p(x)$ ,  $p^{(2)}(x)$ ,  $p^{(4)}(x)$  για  $r = 1.8, 2, 2.6, 2.67, 2.689$  για  $0 < x < 8$ . Για ποιες τιμές του  $r$  περιμένετε να πάρετε σταθερούς  $k$ -κύκλους;
- (β') Για τις ίδιες τιμές του  $r$  παραστήστε γραφικά τροχιές με αρχικά σημεία  $x_0 = 0.2, 0.5, 0.7$ . Για κάθε  $r$  να φτιάξετε μία γραφική παράσταση.
- (γ') Με τη μέθοδο Newton-Raphson να εντοπίσετε τα σημεία  $r_c(n)$  για  $n = 3, 4, 5$  καθώς και τα δύο πρώτα σημεία διακλάδωσης του 3-κύκλου.
- (δ') Να φτιάξετε το διάγραμμα διακλάδωσης για  $1.8 < r < 4$ . Να εντοπίσετε το σημείο έναρξης χάους καθώς και το σημείο έναρξης του 3-κύκλου και να τα μεγεθύνετε γύρω από ένα κλάδο που θα επιλέξετε.
- (ε') Να εκτιμήσετε τον αριθμό  $\delta$  του Feigenbaum όπως στην άσκηση 17. Είναι συμβατό το αποτέλεσμα σας με την παγκοσμιότητα του αριθμού αυτού; Είναι η τιμή του  $r_c$  ίδια με αυτή της λογιστικής εξίσωσης;

3.22 Θεωρήστε την απεικόνιση (sine map):

$$x_{n+1} = s(x_n) = r \sin(\pi x_n). \quad (3.39)$$

- (α') Να κάνετε τη γραφική παράσταση των συναρτήσεων  $x$ ,  $s(x)$ ,  $s^{(2)}(x)$ ,  $s^{(4)}(x)$ ,  $s^{(8)}(x)$  για  $r = 0.65, 0.75, 0.84, 0.86, 0.88$ . Για ποιες τιμές του  $r$  περιμένετε να πάρετε σταθερούς  $k$ -κύκλους;
- (β') Για τις ίδιες τιμές του  $r$  να παραστήστε γραφικά τροχιές με αρχικά σημεία  $x_0 = 0.2, 0.5, 0.7$ . Για κάθε  $r$  να φτιάξετε μία γραφική παράσταση.
- (γ') Με τη μέθοδο Newton-Raphson να εντοπίσετε τα σημεία  $r_c^{(n)}$  για  $n = 3, 4, 5$ , καθώς και τα δύο πρώτα σημεία διακλάδωσης του 3-κύκλου.



(δ') Να φτιάξετε το διάγραμμα διακλάδωσης για  $0.6 < r < 1$ . Μέσα σε ποια όρια κινούνται οι τιμές του  $x$ ; Επαναλάβετε για  $0.6 < r < 2$ . Τι παρατηρείτε; Να εντοπίσετε το σημείο έναρξης χάους, καθώς και το σημείο έναρξης του 3-κύκλου και να τα μεγεθύνετε γύρω από ένα κλάδο που θα επιλέξετε.

3.23 Θεωρήστε την απεικόνιση:

$$x_{n+1} = 1 - rx_n^2. \quad (3.40)$$

(α') Να φτιάξετε το διάγραμμα διακλάδωσης για  $0 < r < 2$ . Μέσα σε ποια όρια κινούνται οι τιμές του  $x$ ; Να εντοπίσετε το σημείο έναρξης χάους, καθώς και το σημείο έναρξης του 3-κύκλου και να τα μεγεθύνετε γύρω από ένα κλάδο που θα επιλέξετε.

(β') Με τη μέθοδο Newton-Raphson να εντοπίσετε τα σημεία  $r_c^{(n)}$  για  $n = 3, 4, 5$ , καθώς και τα δύο πρώτα σημεία διακλάδωσης του 3-κύκλου.

3.24 Θεωρήστε την απεικόνιση (tent map):

$$x_{n+1} = r \min\{x_n, 1 - x_n\} = \begin{cases} rx_n & 0 \leq x_n \leq \frac{1}{2} \\ r(1 - x_n) & \frac{1}{2} < x_n \leq 1 \end{cases}. \quad (3.41)$$

Να φτιάξετε το διάγραμμα διακλάδωσης για  $0 < r < 2$ . Μέσα σε ποια όρια κινούνται οι τιμές του  $x$ ; Στο ίδιο διάγραμμα να παραστήσετε γραφικά τις συναρτήσεις  $r/2$ ,  $r - r^2/2$ .

Μεγεθύνετε την περιοχή του διαγράμματος για  $1.407 < r < 1.416$  και  $0.580 < x < 0.588$ . Σε ποιο σημείο τα δύο μη συνεκτικά διαστήματα μέσα στα οποία παίρνουν τιμές τα  $x_n$  συγχωνεύονται σε ένα; Στη συνέχεια μεγεθύνετε τα διαστήματα  $1.0 < r < 1.1$ ,  $0.4998 < x < 0.5004$  και  $1.00 < r < 1.03$ ,  $0.4999998 < x < 0.5000003$  και εντοπίστε τα σημεία συγχώνευσης δύο μη συνεκτικών διαστημάτων μέσα στα οποία παίρνουν τιμές τα  $x_n$ .

3.25 Θεωρήστε την απεικόνιση (gauss map ή mouse map):

$$x_{n+1} = e^{-rx_n^2} + q. \quad (3.42)$$

Να φτιάξετε το διάγραμμα διακλάδωσης για  $-1 < q < 1$  και  $r = 4.5, 4.9, 7.5$ . Να κάνετε το πρόγραμμα υπολογισμού του διαγράμματος να παίρνει ως αρχικό σημείο της νέας τροχιάς το τελευταίο

της προηγούμενης και να επιλέξετε  $x_0 = 0$  για  $q = -1$ . Επαναλάβετε με  $x_0 = 0.7, 0.5, -0.7$ . Τι παρατηρείτε; Παρατηρήστε ότι, καθώς αυξάνει το  $q$ , έχουμε διακλαδώσεις και “αντι-διακλαδώσεις”.

3.26 Θεωρήστε την απεικόνιση (circle map):

$$x_{n+1} = [x_n + r - q \sin(2\pi x_n)] \mod 1. \quad (3.43)$$

(Προσοχή στο πρόγραμμα, οι τιμές του  $0 \leq x_n < 1$ ). Να φτιάξετε το διάγραμμα διακλάδωσης για  $0 < q < 2$  και  $r = 1/3$ .

3.27 Χρησιμοποιήστε το πρόγραμμα `liapunov.f90` για να υπολογίσετε την απόσταση δύο τροχιών της λογιστικής απεικόνισης με  $r = 3.6$  οι οποίες αρχικά απέχουν  $\Delta x_0 = 10^{-15}$ . Επιλέξτε  $x_0 = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.999$  και υπολογίστε τον εκθέτη Liapunov με προσαρμογή ευθείας γραμμής στα κατάλληλα διαστήματα. Υπολογίστε τη μέση τιμή τους και το σφάλμα της μέσης τιμής.

3.28 Υπολογίστε τον εκθέτη Liapunov για  $r = 3.58, 3.60, 3.65, 3.70, 3.80$  και με τους δύο τρόπους που αναπτύσσονται στο κείμενο. Χρησιμοποιήστε τουλάχιστον 5 διαφορετικά αρχικά σημεία και υπολογίστε τη μέση τιμή τους και το σφάλμα της μέσης τιμής. Συγκρίνετε τις τιμές που παίρνετε για το  $\lambda$  και σχολιάστε.

3.29 Υπολογίστε αριθμητικά την κρίσιμη τιμή  $r_c$  που είναι το όριο  $\lim_{n \rightarrow \infty} r_c^{(n)}$  για τη λογιστική εξίσωση με ακρίβεια 9 δεκαδικών ψηφίων. Χρησιμοποιήστε τον υπολογισμό του εκθέτη Liapunov  $\lambda$  που δίνεται από τη σχέση (3.35).

3.30 Υπολογίστε αριθμητικά τις κρίσιμες τιμές του  $r$  της λογιστικής εξίσωσης στις οποίες (α) εισερχόμαστε στον σταθερό 3-κύκλο, (β) επανέρχεται η χαοτική συμπεριφορά μετά από τις διακλαδώσεις του 3-κύκλου. Κάνετε τον υπολογισμό με τη βοήθεια του εκθέτη Liapunov και επιβεβαιώστε τα αποτελέσματά σας συγκρίνοντάς με το διάγραμμα διακλάδωσης.

3.31 Υπολογίστε τον εκθέτη Liapunov με τη βοήθεια της σχέσης (3.35)

για τις ακόλουθες απεικονίσεις:

$$\begin{aligned}
 x_{n+1} &= x_n e^{r(1-x_n)}, & 1.8 < r < 4 \\
 x_{n+1} &= r \sin(\pi x_n), & 0.6 < r < 1 \\
 x_{n+1} &= 1 - rx_n^2, & 0 < r < 2 \\
 x_{n+1} &= e^{-rx_n^2} + q, & r = 7.5, -1 < q < 1 \\
 x_{n+1} &= \left[ x_n + \frac{1}{3} - q \sin(2\pi x_n) \right] \bmod 1, & 0 < q < 2, (3.44)
 \end{aligned}$$

και να φτιάξετε τα ανάλογα σχήματα με το 3.9. Να αντιπαραθέσετε τα σχήματα που θα φτιάξετε με τα αντίστοιχα διαγράμματα διακλάδωσης (μπορείτε να βάλετε και τα δύο στο ίδιο σχήμα). Στη gauss map ( $x_{n+1} = e^{-rx_n^2} + q$ ) να κάνετε τον υπολογισμό για δύο αρχικά σημεία  $x_0 = 0, 0.2$  και να παρατηρήσετε τη διαφορά. Στη circle map ( $x_{n+1} = [x_n + 1/3 - q \sin(2\pi x_n)] \bmod 1$ ) να μελετήσετε ιδιαίτερα την περιοχή  $0 < q < 0.15$ .

3.32 Αφού αναπαράγετε τα σχήματα 3.10, 3.11 και 3.12, να υπολογίσετε τη συνάρτηση  $p(x)$  για  $r = 3.68, 3.80, 3.93$  και  $3.98$ . Να συζητήσετε σε ποια σημεία έχετε ισχυρότερη χαοτική συμπεριφορά από τις  $p(x)$  και από τις αντίστοιχες τιμές της εντροπίας. Στη συνέχεια, να υπολογίσετε την εντροπία μέσα στο διάστημα  $r \in (3.95, 4.00)$  παίρνοντας RSTEPS=2000 και να εκτιμήσετε τις τιμές του  $r$  μέσα στο διάστημα αυτό που έχουμε είσοδο και έξοδο από το χάος. Συγκρίνετε τα αποτελέσματά σας με τον αντίστοιχο υπολογισμό που θα κάνετε για τον εκθέτη Liapunov.

3.33 Θεωρήστε την απεικόνιση (Hénon map):

$$\begin{aligned}
 x_{n+1} &= y_n + 1 - ax_n^2 \\
 y_{n+1} &= bx_n
 \end{aligned} \tag{3.45}$$

(α') Φτιάξτε τα δύο διαγράμματα διακλάδωσης για τις τιμές  $x_n, y_n$  για  $b = 0.3, 1.0 < a < 1.5$ . Ελέγξτε αν οι τιμές  $a = 1.01, 1.4$  που θα χρησιμοποιήσουμε παρακάτω αντιστοιχούν σε σταθερές περιοδικές τροχιές ή χαοτική συμπεριφορά.

(β') Γράψτε ένα πρόγραμμα `attractor.f90` που θα παίρνει NINIT = NL × NL αρχικές συνθήκες  $(x_0(i), y_0(i))$   $i = 1, \dots, \text{NL}$  πάνω σε ένα NL × NL πλέγμα τιμών στο τετράγωνο  $x_m \leq x_0 \leq x_M, y_m \leq y_0 \leq y_M$ . Κάθε ένα από τα σημεία  $(x_0(i), y_0(i))$  θα εξελίσσεται σύμφωνα με τη σχέση (3.45) για  $n = \text{NSTEPS}$  βήματα.

Το πρόγραμμα τότε θα εκτυπώνει τα σημεία  $(x_n(i), y_n(i))$  στο stdout. Επιλέξτε  $x_m = y_m = 0.6$ ,  $x_M = y_M = 0.8$ , NL= 200.

- (γ') Επιλέξτε  $a = 1.01$ ,  $b = 0.3$  και φτιάξτε στο ίδιο σχήμα τη γραφική παράσταση των σημείων  $(x_n(i), y_n(i))$  για  $n = 0, 1, 2, 3, 10, 20, 30, 40, 60, 1000$ .
- (δ') Επιλέξτε  $a = 1.4$ ,  $b = 0.3$  και φτιάξτε στο ίδιο σχήμα τη γραφική παράσταση των σημείων  $(x_n(i), y_n(i))$  για  $n = 0, \dots, 7$ .
- (ε') Επιλέξτε  $a = 1.4$ ,  $b = 0.3$  και φτιάξτε στο ίδιο σχήμα τη γραφική παράσταση των σημείων  $(x_n(i), y_n(i))$  για  $n = 999$ . Παρατηρήστε τον παράξενο ελκυστή Hénon (Hénon strange attractor) και τις fractal ιδιότητές του. Αυτή χαρακτηρίζεται από διάσταση Hausdorff<sup>12</sup>  $d_H = 1.261 \pm 0.003$ . Μεγεθύνετε τις περιοχές

$$\begin{aligned} \{(x, y) | & -1.290 < x < -1.270, \quad 0.378 < y < 0.384\}, \\ \{(x, y) | & 1.150 < x < -1.130, \quad 0.366 < y < 0.372\}, \\ \{(x, y) | & 0.108 < x < 0.114, \quad 0.238 < y < 0.241\}, \\ \{(x, y) | & 0.300 < x < 0.320, \quad 0.204 < y < 0.213\}, \\ \{(x, y) | & 1.076 < x < 1.084, \quad 0.090 < y < 0.096\}, \\ \{(x, y) | & 1.216 < x < 1.226, \quad 0.032 < y < 0.034\}. \end{aligned}$$

### 3.34 Θεωρήστε την απεικόνιση (Duffing map):

$$\begin{aligned} x_{n+1} &= y_n \\ y_{n+1} &= -bx_n + ay_n - y_n^3. \end{aligned} \quad (3.46)$$

- (α') Φτιάξτε τα δύο διαγράμματα διακλάδωσης για τις τιμές  $x_n, y_n$  για  $b = 0.3$ ,  $0 < a < 2.78$ . Επιλέξτε τέσσερις αρχικές συνθήκες  $(x_0, y_0) = (\pm 1/\sqrt{2}, \pm 1/\sqrt{2})$ . Τι παρατηρείτε;
- (β') Χρησιμοποιήστε κατάλληλα το πρόγραμμα attractor.f90 της άσκησης 33 για να μελετήσετε τον ελκυστή της απεικόνισης για  $b = 0.3$ ,  $a = 2.75$ .

### 3.35 Θεωρήστε την απεικόνιση (Tinkerbell map):

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + ax_n + by_n \\ y_{n+1} &= 2x_n y_n + cx_n + dy_n. \end{aligned} \quad (3.47)$$

<sup>12</sup>D.A. Russel, J.D. Hanson, and E. Ott, “Dimension of strange attractors”, Phys. Rev. Lett. 45 (1980) 1175. Δείτε και λήμμα “Hausdorff dimension” στη Wikipedia.

- (α') Επιλέξτε  $a = 0.9$ ,  $b = -0.6013$ ,  $c = 2.0$ ,  $d = 0.50$ . Απεικονίστε μια τροχιά στο επίπεδο κάνοντας τη γραφική παράσταση των σημείων  $(x_n, y_n)$  για  $n = 0, \dots, 10\,000$  με  $(x_0, y_0) = (-0.72, -0.64)$ .
- (β') Χρησιμοποιήστε κατάλληλα το πρόγραμμα `attractor.f90` της άσκησης 33 για να μελετήσετε τον ελκυστή της απεικόνισης για τις τιμές των παραμέτρων  $a$ ,  $b$ ,  $c$ ,  $d$  που δίνονται παραπάνω. Επιλέξτε  $x_m = -0.68$ ,  $x_M = -0.76$ ,  $y_m = -0.60$ ,  $y_M = -0.68$ ,  $n = 10\,000$ .
- (γ') Επαναλάβετε το προηγούμενο υποερώτημα αλλάζοντας το  $d = 0.27$ .



## ΚΕΦΑΛΑΙΟ 4

### Κίνηση Σωματιδίου

Στο κεφάλαιο αυτό μελετάται αριθμητικά η επίλυση των κλασικών εξισώσεων κίνησης μονοδιάστατων μηχανικών συστημάτων, όπως λ.χ. αυτή του σημειακού σωματιδίου σε μια ευθεία, του απλού εκκρεμούς κλπ. Γίνεται εισαγωγή σε μεθόδους αριθμητικής ολοκλήρωσης διαφορικών εξισώσεων με αρχικές συνθήκες και ιδιαίτερα στη μέθοδο Runge–Kutta 4ης τάξης. Τέλος, μελετώνται τα συστήματα του αρμονικού ταλαντωτή και του απλού εκκρεμούς με απόσβεση και οδήγηση από εξωτερική χρονοεξαρτημένη δύναμη. Το τελευταίο σύστημα είναι μη γραμμικό και γίνεται μια εισαγωγή στις χαοτικές ιδιότητές του.

#### 4.1 Αριθμητική Ολοκλήρωση Εξισώσεων Νεύτωνα

Θεωρούμε το πρόβλημα της λύσης των εξισώσεων κίνησης σωματιδίου υπό την επίδραση δυνάμεων που δίνονται από το νόμο του Νεύτωνα. Οι εξισώσεις αυτές μπορούν να γραφτούν υπό τη μορφή

$$\frac{d^2\vec{x}}{dt^2} = \vec{a}(t, \vec{x}, \vec{v}), \quad (4.1)$$

όπου

$$\vec{a}(t, \vec{x}, \vec{v}) \equiv \frac{\vec{F}}{m} \quad \vec{v} = \frac{d\vec{x}}{dt}. \quad (4.2)$$

Η κλάση των προβλημάτων που θα συζητήσουμε σε επίπεδο αριθμητικής ανάλυσης είναι προβλήματα αρχικών τιμών, δηλ. επίλυση διαφορικών εξισώσεων για τις οποίες δίνονται οι αρχικές συνθήκες

$$\vec{x}(t_0) = \vec{x}_0 \quad \vec{v}(t_0) = \vec{v}_0, \quad (4.3)$$

οι οποίες προσδιορίζουν μία μοναδική λύση  $\vec{x}(t)$ . Οι διαφορικές εξισώσεις (4.1) είναι δεύτερης τάξης ως προς τις συναρτήσεις  $\vec{x}(t)$ . Για την αριθμητική λύση τους είναι βολικό να ανάγουμε τις εξισώσεις αυτές σε ένα σύστημα από διπλάσιο αριθμό εξισώσεων πρώτου βαθμού:

$$\frac{d\vec{x}}{dt} = \vec{v} \quad \frac{d\vec{v}}{dt} = \vec{a}(t, \vec{x}, \vec{v}). \quad (4.4)$$

Ειδικά θα ενδιαφερθούμε για την κίνηση σωματιδίου πάνω στην ευθεία (1 διάσταση), οπότε το σύστημα των εξισώσεων γίνεται

$$\begin{aligned} \frac{dx}{dt} &= v & \frac{dv}{dt} &= a(t, x, v) & \text{1-διάσταση} \\ x(t_0) &= x_0 & v(t_0) &= v_0, \end{aligned} \quad (4.5)$$

καθώς και στο επίπεδο (2 διαστάσεις), οπότε το σύστημα των εξισώσεων γίνεται

$$\begin{aligned} \frac{dx}{dt} &= v_x & \frac{dv_x}{dt} &= a_x(t, x, v_x, y, v_y) & \text{2-διαστάσεις} \\ \frac{dy}{dt} &= v_y & \frac{dv_y}{dt} &= a_y(t, x, v_x, y, v_y) \\ x(t_0) &= x_0 & v_x(t_0) &= v_{0x} \\ y(t_0) &= y_0 & v_y(t_0) &= v_{0y}, \end{aligned} \quad (4.6)$$

## 4.2 Πρελούδιο: Μέθοδοι Euler

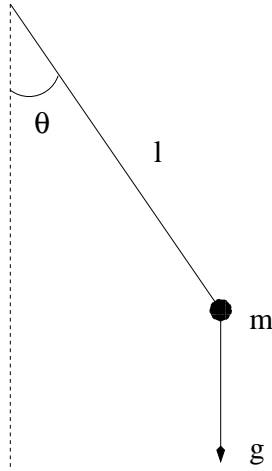
Για να πάρουμε μια πρώτη γεύση του προβλήματος θα μελετήσουμε το πρόβλημα του εκκρεμούς μήκους  $l$  μέσα σε ομογενές πεδίο βαρύτητας  $g$  (σχήμα 4.1). Οι εξισώσεις κίνησης δίνονται από το σύστημα των διαφορικών εξισώσεων

$$\begin{aligned} \frac{d^2\theta}{dt^2} &= -\frac{g}{l} \sin \theta \\ \frac{d\theta}{dt} &= \omega, \end{aligned} \quad (4.7)$$

που εύκολα ανάγεται στο σύστημα πρώτης τάξης

$$\begin{aligned} \frac{d\theta}{dt} &= \omega \\ \frac{d\omega}{dt} &= -\frac{g}{l} \sin \theta, \end{aligned} \quad (4.8)$$





Σχήμα 4.1: Το εκκρεμές μήκους  $l$  μέσα σε ομογενές πεδίο βαρύτητας  $g$ .

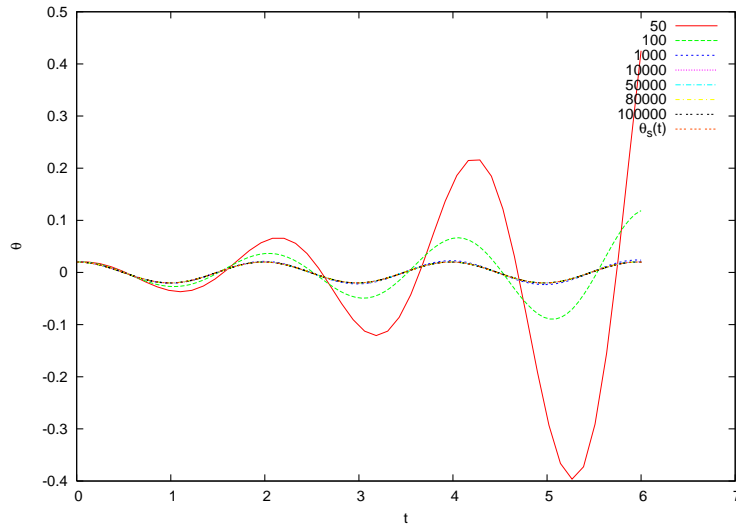
Το παραπάνω σύστημα πρέπει να γραφτεί σε διακριτή μορφή, έτσι ώστε να επιτευχθεί η αριθμητική του επίλυση με τη βοήθεια υπολογιστή. Ο πιο απλός τρόπος είναι να θεωρήσουμε την ολοκλήρωση του συστήματος από τον αρχικό χρόνο  $t_i = 0$  μέχρι τελικό χρόνο  $t_f$  χωρίζοντας το χρονικό διάστημα  $t_f - t_i$  σε  $N - 1$  ίσα διαστήματα πλάτους<sup>1</sup>  $\Delta t \equiv h$ , όπου  $h = (t_f - t_i)/(N - 1)$  και προσεγγίζοντας τις παραγώγους από τις σχέσεις  $(x_{n+1} - x_n)/\Delta t \approx x'_n$ :

$$\begin{aligned}\omega_{n+1} &= \omega_n + \alpha_n \Delta t \\ \theta_{n+1} &= \theta_n + \omega_n \Delta t.\end{aligned}\tag{4.9}$$

όπου  $\alpha = -(g/l) \sin \theta$  η γωνιακή επιτάχυνση. Η μέθοδος αυτή ακούει στο όνομα “μέθοδος Euler”. Το σφάλμα που εισάγεται από τη μέθοδο σε κάθε βήμα είναι της τάξης του  $(\Delta t)^2$ . Πράγματι, αυτό προκύπτει από απλή ανάπτυξη κατά Taylor γύρω από το σημείο  $t_n$  αγνοώντας όλους τους όρους από τη δεύτερη παράγωγο και πάνω<sup>2</sup>. Κατά την ολοκλήρωση από  $t_i$  σε  $t_f$ , το συνολικό σφάλμα είναι τάξης  $\Delta t$ ! Ο λόγος είναι ότι τα σφάλματα αυτά προστίθενται σε κάθε βήμα και αφού ο αριθμός των βημάτων  $N \propto 1/\Delta t$ , το συνολικό σφάλμα είναι  $\propto (\Delta t)^2 \times (1/\Delta t) = \Delta t$ . Η μέθοδος Euler λέμε ότι είναι μια μέθοδος πρώτης τάξης και ως εκ τούτου έχει περιορισμένη ακρίβεια, ειδικά σε προβλήματα που παρουσιάζουν περιοδικότητα. Η μέθοδος είναι ασύμμετρη, γιατί χρησιμοποιεί για την προώθηση της λύσης πληροφορία για τη συνάρτηση μόνο στην αρχή του

<sup>1</sup>Έχουμε  $N$  διακριτούς χρόνους  $t_i \equiv t_1, \dots, t_{N-2}, t_{N-1} \equiv t_f$

<sup>2</sup>Δείτε το Παράρτημα 4.7 για τις λεπτομέρειες.



Σχήμα 4.2: Σύγκλιση της μεθόδου Euler για το απλό εκκρεμές με περίοδο  $T \approx 1.987$  ( $\omega^2 = 10.0$ ) για διαφορετικές τιμές του βήματος χρόνου  $\Delta t$  που καθορίζεται από τον αριθμό των βημάτων  $Nt = 50 - 100,000$ . Η λύση είναι για  $\theta_0 = 0.2$ ,  $\omega_0 = 0.0$  και συγκρίνεται με τη γνωστή λύση για μικρές γωνίες με  $\alpha(t) \approx -(g/l)\theta$ .

διαστήματος  $(t, t + \Delta t)$ . Με μια απλή παραλλαγή παίρνουμε τη μέθοδο Euler–Cromer η οποία παρουσιάζει βελτιωμένη συμπεριφορά, αν και αυτή είναι πρώτης τάξης με συνολικό σφάλμα  $\propto \Delta t$ . Για το λόγο αυτό, χρησιμοποιούμε για την προώθηση της γωνίας  $\theta$  την καινούργια τιμή της γωνιακής ταχύτητας

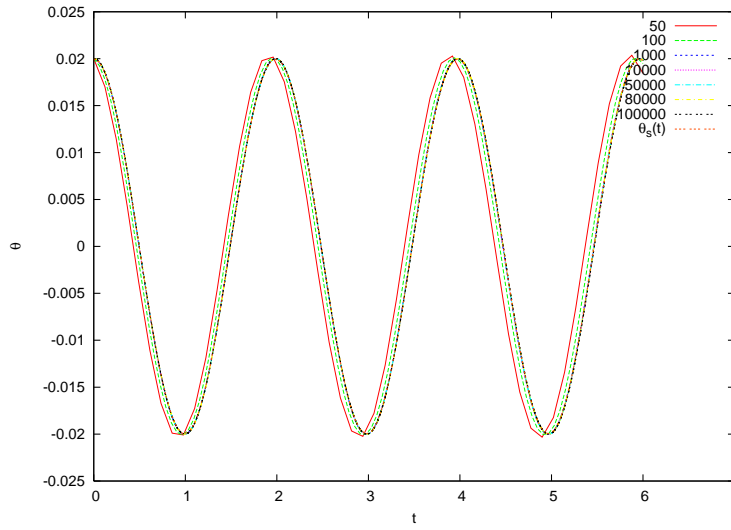
$$\begin{aligned}\omega_{n+1} &= \omega_n + \alpha_n \Delta t \\ \theta_{n+1} &= \theta_n + \omega_{n+1} \Delta t.\end{aligned}\quad (4.10)$$

Μια μέθοδος που βελτιώνει τους παραπάνω αλγόριθμους ως προς το σφάλμα διακριτοποίησης είναι ο αλγόριθμος Euler–Verlet ο οποίος δίνει ολικό σφάλμα<sup>3</sup>  $\sim (\Delta t)^2$ . Αυτός δίνεται από τις εξισώσεις

$$\begin{aligned}\theta_{n+1} &= 2\theta_n - \theta_{n-1} + \alpha_n (\Delta t)^2 \\ \omega_n &= \frac{\theta_{n+1} - \theta_{n-1}}{2\Delta t}.\end{aligned}\quad (4.11)$$

Η μέθοδος Euler–Verlet (4.11) είναι μια μέθοδος δύο βημάτων, αφού για την προώθηση της λύσης είναι αναγκαίο να γνωρίζουμε την τιμή της συνάρτησης σε δύο προηγούμενα βήματα. Άρα, πρέπει να καθορίσουμε

<sup>3</sup>Δείτε το Παράρτημα 4.7 για τις λεπτομέρειες.



Σχήμα 4.3: Σύγκλιση της μεθόδου Euler-Cromer, παρόμοια με το σχήμα 4.2. Παρατηρούμε πως η μέθοδος συγκλίνει πολύ γρηγορότερα από την Euler.

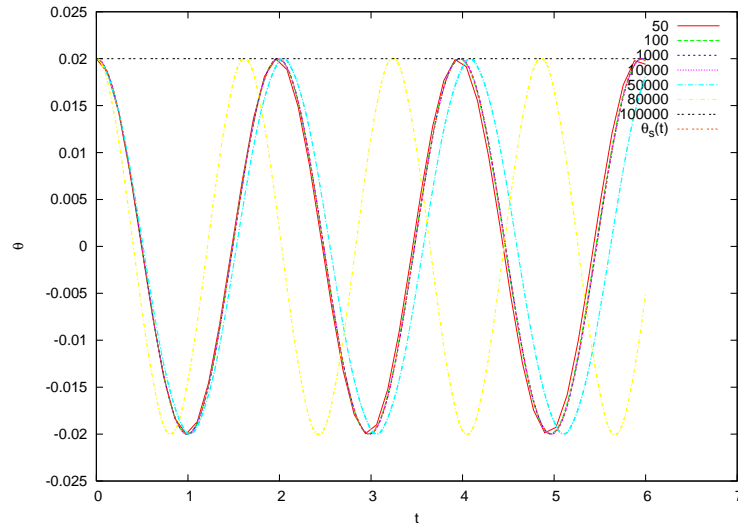
προσεκτικά τις αρχικές συνθήκες για τα δύο πρώτα βήματα. Για το λόγο αυτό χρησιμοποιούμε τον αλγόριθμο Euler για να προωθήσουμε τις αρχικές συνθήκες ένα βήμα πίσω. Αν  $\theta_1 = \theta(t_i)$ ,  $\omega_1 = \omega(t_i)$  είναι οι αρχικές συνθήκες, τότε ορίζουμε

$$\theta_0 = \theta_1 - \omega_1 \Delta t + \frac{1}{2} \alpha_1 (\Delta t)^2. \quad (4.12)$$

Επίσης, στο τελευταίο βήμα θα πρέπει να πάρουμε

$$\omega_N = \frac{\theta_N - \theta_{N-1}}{\Delta t}. \quad (4.13)$$

Παρόλο που η μέθοδος έχει μικρότερο συνολικό σφάλμα από τη μέθοδο Euler, το πρόβλημά της είναι ότι είναι ασταθής, όταν το  $\Delta t$  γίνει αρκετά μικρό. Στη δεύτερη των εξισώσεων (4.11) η γωνιακή ταχύτητα προκύπτει από το λόγο δύο μικρών αριθμών, εκ των οποίων ο αριθμητής είναι η διαφορά δύο μεγάλων, σχεδόν ίσων, αριθμών. Για μικρό χρόνο  $\Delta t$ , η τιμή της διαφοράς αυτής εξαρτάται μόνο από τα τελευταία δεκαδικά ψηφία της αναπαράστασης των αριθμών  $\theta_{n+1}$  και  $\theta_n$ , επειδή ο υπολογιστής έχει πεπερασμένη ακρίβεια στην αναπαράσταση τους και η ακρίβεια στον υπολογισμό του λόγου  $(\theta_{n+1} - \theta_n)/(2\Delta t)$  μειώνεται μέχρι ο λόγος αυτός να γίνει ακριβώς μηδέν λόγω μηδενισμού του αριθμητή! Στην πρώτη των εξισώσεων (4.11), ο όρος  $\alpha_n \Delta t^2$  είναι μικρότερος από τον αντίστοιχο όρο  $\alpha_n \Delta t$  της μεθόδου Euler κατά μία τάξη



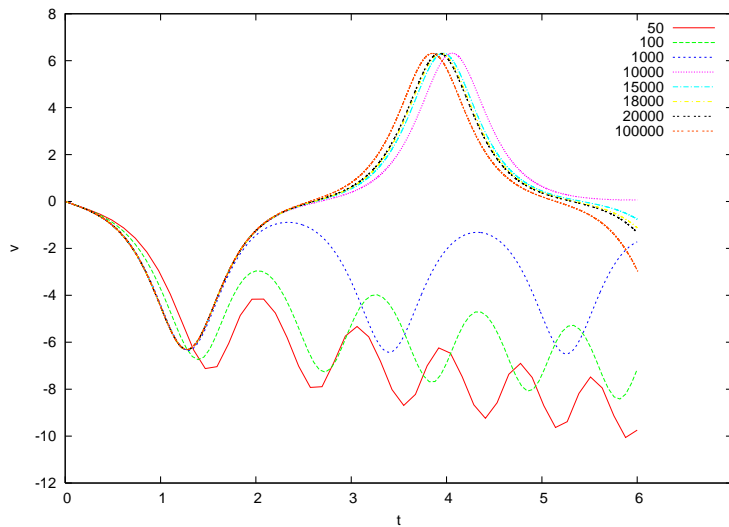
Σχήμα 4.4: Σύγκλιση της μεθόδου Euler-Verlet, όπως στο σχήμα 4.2. Παρατηρούμε πως η μέθοδος συγκλίνει πολύ γρηγορότερα από την Euler αλλά τα σφάλματα στρογγυλοποίησης κάνουν τη μέθοδο άχρηστη για  $Nt \gtrsim 50,000$  (προσέξτε τι γίνεται για  $Nt = 100,000$ . Γιατί;).

ως προς  $\Delta t$ . Μειώνοντας το  $\Delta t$ , γρήγορα έχουμε  $\alpha_n \Delta t^2 \ll 2\theta_n - \theta_{n-1}$  και η ακρίβεια της μεθόδου εκμηδενίζεται λόγω της πεπερασμένης ακρίβειας των πραγματικών αριθμών στη μνήμη του υπολογιστή<sup>4</sup>. Όταν οι αριθμοί  $\alpha_n \Delta t^2$  και  $2\theta_n - \theta_{n-1}$  διαφέρουν περισσότερο από περίπου επτά τάξεις μεγέθους, η πρόσθεση του πρώτου όρου στο δεύτερο είναι το ίδιο σαν να προσθέταμε μηδέν και η συνεισφορά του όρου της επιτάχυνσης εξαφανίζεται.<sup>5</sup>

Ο προγραμματισμός των μεθόδων αυτών είναι ιδιαίτερα απλός. Γράφουμε ένα πρόγραμμα που θα κάνει σύγκριση και των τριών μεθόδων, Euler, Euler-Cromer και Euler-Verlet. Για το λόγο αυτό, το κυρίως πρόγραμμα είναι απλά ένα interface με το χρήστη και για τους υπολογισμούς καλούνται διαφορετικές υπορουτίνες `euler`, `euler_cromer` και `euler_verlet`. Ο χρήστης απλά πρέπει να προγραμματίσει την κοινή σε όλους τους υπολογισμούς συνάρτηση `accel(x)` που δίνει τη γωνιακή επιτάχυνση συναρτήσει της γωνίας  $\theta$  (εδώ η μεταβλητή `REAL x`). Στην

<sup>4</sup>Δείτε παρατήρηση στην υποσημείωση 19 στη σελ. 121.

<sup>5</sup>Αριθμοί τύπου `real` έχουν προσεγγιστικά 7 σημαντικά δεκαδικά ψηφία. Η ακρίβεια των παραπάνω υπολογισμών καθορίζεται από τον αριθμό  $\epsilon$ , που είναι ο μικρότερος θετικός αριθμός, τέτοιος ώστε  $1+\epsilon > 1$ . Για μια μεταβλητή  $x$  συγκεκριμένου τύπου, ο αριθμός αυτός δίνεται από την *intrinsic* συνάρτηση Fortran `epsilon(x)`. Για μεταβλητές τύπου `real`,  $\epsilon \approx 1.2 \times 10^{-7}$  και για μεταβλητές τύπου `real(8)`  $\epsilon \approx 2.2 \times 10^{-16}$ .



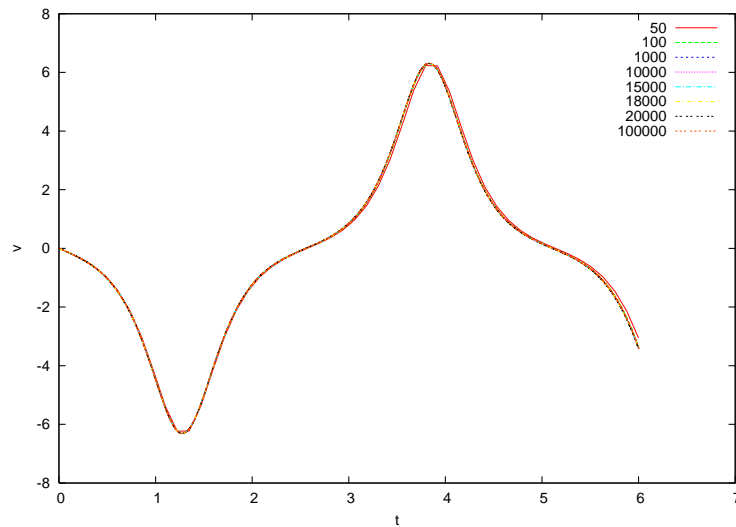
Σχήμα 4.5: Σύγκλιση της μεθόδου Euler για το απλό εκκρεμές όπως στο σχήμα 4.2 αλλά για  $\theta_0 = 3.0$ ,  $\omega_0 = 0.0$ . Εδώ δείχνουμε τη συμπεριφορά της γωνιακής ταχύτητας και παρατηρούμε μεγάλη αστάθεια για  $Nt \lesssim 1,000$ .

παράγραφο αυτή παίρνουμε  $\text{accel}(x) = -10.0 * \sin(x)$ .

Η δομή των δεδομένων είναι πολύ απλή: Σε τρία arrays REAL  $T(P)$ ,  $X(P)$  και  $V(P)$  αποθηκεύονται αντίστοιχα οι χρόνοι  $t_n$ , οι γωνιακές θέσεις  $\theta_n$  και οι γωνιακές ταχύτητες  $\omega_n$  για  $n = 1, \dots, Nt$ . Ο χρήστης προσδιορίζει το χρονικό διάστημα της ολοκλήρωσης από  $t_i = 0$  σε  $t_f = Tfi$ , καθώς και τον αριθμό των χρονικών σημείων  $Nt$  τα οποία πρέπει να είναι λιγότερα από το μέγεθος  $P$  των arrays. Μετά δίνει τις αρχικές συνθήκες  $\theta_0 = Xin$  και  $\omega_0 = Vin$ . Στη συνέχεια, οι υπορουτίνες των μεθόδων καλούνται και στην είσοδο τους παρέχουμε τις αρχικές συνθήκες, το διάστημα ολοκλήρωσης και τον αριθμό χρονικών σημείων  $Xin, Vin, Tfi, Nt$ . Στην έξοδο μας παρέχουν τα αποτελέσματα αποθηκευμένα στα arrays  $T, X, V$ . Στη συνέχεια, τα αποτελέσματα τυπώνονται στα αρχεία `euler.dat`, `euler_cromer.dat` και `euler_verlet.dat`.

Οι υπορουτίνες υπολογισμού ακολουθούν τη διαδικασία καθορισμού των αρχικών συνθηκών, υπολογισμού του χρόνου  $\Delta t \equiv h = Tfi/(Nt - 1)$  και εκτελούν τους βρόχους που θα προωθούν τη λύση με βήμα  $\Delta t$ . Σε κάθε βήμα τα αποτελέσματα αποθηκεύονται στα arrays  $T, X, V$ . Έτσι, το τμήμα του κώδικα

```
T(1) = 0.0
X(1) = Xin
```



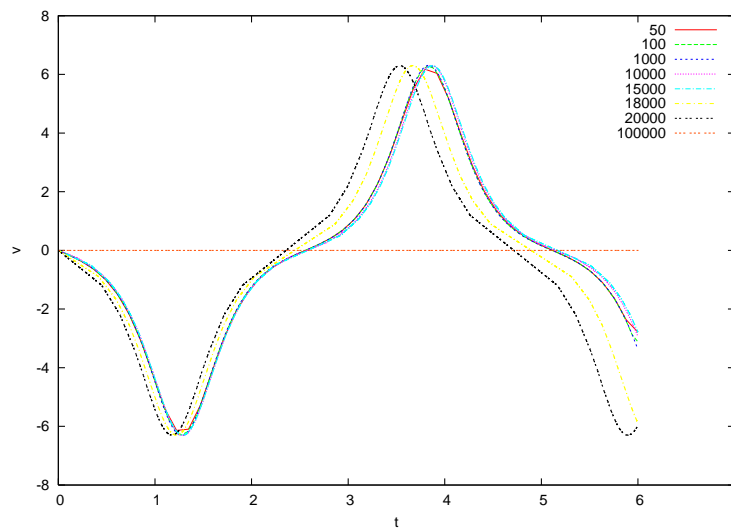
Σχήμα 4.6: Σύγκριση της μεθόδου Euler-Cromer, όπως στο σχήμα 4.5. Παρατηρούμε πως η μέθοδος συγκλίνει πολύ γρηγορότερα από την Euler.

```
V(1) = Vin
h = Tfi/(Nt-1)
do i = 2,Nt
  T(i) = T(i-1)+h
  X(i) = X(i-1)+V(i-1)*h
  V(i) = V(i-1)+accel(X(i-1))*h
enddo
```

εκτελεί τις απαραίτητες εντολές για τη μέθοδο Euler. Λίγη προσοχή πρέπει να δοθεί στη μέθοδο Euler-Verlet όπου πρέπει να καθοριστούν τα δύο πρώτα βήματα, καθώς και το τελευταίο για την ταχύτητα:

```
T(1) = 0.0
X(1) = Xin
V(1) = Vin
X0 = X(1) - V(1) * h + accel(X(1)) * h*h/2.0
T(2) = h
X(2) = 2.0*X(1) - X0 + accel(X(1)) * h*h
do i = 3,Nt
  .....
enddo
V(Nt) = (X(Nt)-X(Nt-1))/h
```

Για διευκόλυνση του αναγνώστη παραθέτουμε ολόκληρο το πρόγραμμα παρακάτω:



Σχήμα 4.7: Σύγκλιση της μεθόδου Euler-Verlet, όπως στο σχήμα 4.5. Παρατηρούμε πως η μέθοδος συγκλίνει πολύ γρηγορότερα από την Euler, αλλά τα σφάλματα στρογγυλοποίησης κάνουν τη μέθοδο πολύ γρήγορα ασταθή για  $Nt \gtrsim 18,000$ .

```

=====
!Program to integrate equations of motion for accelerations
!which are functions of x with the method of Euler,
!Euler-Cromer and Euler-Verlet.
!The user sets initial conditions and the subroutines return
!X(t) and V(t)=dX(t)/dt in arrays T(1..Nt),X(1..Nt),V(1..Nt)
!The user provides number of times Nt and the final
!time Tfi. Initial time is assumed to be t_i=0 and the
!integration step h = Tfi/(Nt-1)
!The user programs a real function accel(x) which gives the
!acceleration dV(t)/dt as function of X.
!NOTE: T(1) = 0 T(Nt) = Tfi
=====
program diff_eq_euler
  implicit none
  integer,parameter:: P=110000 ! The size of the arrays
  real,dimension(P):: T,X,V    ! time t,x(t),v(t)=dx/dt
  real      :: Xin,Vin,Tfi      ! initial conditions
  integer   :: Nt,i
  !The user provides initial conditions X_0,V_0 final time t_f
  !and Nt:
  print *, 'Enter X_0,V_0,t_f,Nt (t_i=0):'
  read(5,*)Xin,Vin,Tfi,Nt
  !This check is necessary in order to avoid memory
  !access violations:

```

```

if(Nt .ge. P )then
  print *, 'Nt must be strictly less than P. Nt,P= ',Nt,P
  stop
endif
!Xin= X(1), Vin=V(1), T(1)=0 and the routine gives evolution in
!T(2..Nt), X(2..Nt), V(2..Nt) which we print in a file
call euler(Xin,Vin,Tfi,Nt,T,X,V)
open(unit=20,file="euler.dat")
do i=1,Nt
!Each line in data file has time, position, velocity:
  write(20,*) T(i),X(i),V(i)
enddo
close(20) !we close the unit to be reused below
!
!We repeat everything for each method
call euler_cromer(Xin,Vin,Tfi,Nt,T,X,V)
open(unit=20,file="euler_cromer.dat")
do i=1,Nt
  write(20,*) T(i),X(i),V(i)
enddo
close(20)
!
call euler_verlet(Xin,Vin,Tfi,Nt,T,X,V)
open(unit=20,file="euler_verlet.dat")
do i=1,Nt
  write(20,*) T(i),X(i),V(i)
enddo
close(20)
!
end program diff_eq_euler
!=====
!Function which returns the value of acceleration at
!position x used in the integration subroutines
!euler, euler_cromer and euler_verlet
!=====
real function accel(x)
  implicit none
  real x
  accel = -10.0*sin(x)
end function accel
!=====
!Driver routine for integrating equations of motion
!using the Euler method
!Input:
!Xin=X(1), Vin=V(1) — initial condition at t=0,
!Tfi the final time and Nt the number of times
!Output:
!The arrays T(1..Nt), X(1..Nt), V(1..Nt) which
!gives x(t_k)=X(k), dx/dt(t_k)=V(k), t_k=T(k) k=1..Nt

```



```

!where for k=1 we have the initial condition.
!=====
subroutine euler(Xin,Vin,Tfi,Nt,T,X,V)
  implicit none
  integer :: Nt
  real,dimension(Nt) :: T,X,V !time t,x(t),v(t)=dx/dt
  real :: Xin,Vin,Tfi
  integer :: i
  real :: h,accel !**declare the function accel**
!Initial conditions set here:
  T(1) = 0.0
  X(1) = Xin
  V(1) = Vin
!h is the time step Dt
  h = Tfi/(Nt-1)
  do i = 2,Nt
    T(i) = T(i-1)+h ! time advances by Dt=h
    X(i) = X(i-1)+V(i-1)*h ! advancement and storage of ↵
      position
    V(i) = V(i-1)+accel(X(i-1))*h !and velocity.
  enddo
end subroutine euler
!=====
!Driver routine for integrating equations of motion
!using the Euler-Cromer method
!Input:
!Xin=X(1), Vin=V(1) — initial condition at t=0,
!Tfi the final time and Nt the number of times
!Output:
!The arrays T(1..Nt), X(1..Nt), V(1..Nt) which
!gives x(t_i)=X(i), dx/dt(t_i)=V(i), t_i=T(i) i=1..Nt
!where for i=1 we have the initial condition.
!=====
subroutine euler_cromer(Xin,Vin,Tfi,Nt,T,X,V)
  implicit none
  integer :: Nt
  real,dimension(Nt):: T,X,V !time t,x(t),v(t)=dx/dt
  real :: Xin,Vin,Tfi
  integer :: i
  real :: h,accel

  T(1) = 0.0
  X(1) = Xin
  V(1) = Vin
  h = Tfi/(Nt-1)
  do i = 2,Nt
    T(i) = T(i-1)+h
    V(i) = V(i-1)+accel(X(i-1))*h

```

```

    X(i) = X(i-1)+V(i)*h  !here is the difference compared to ↔
    Euler
enddo

end subroutine euler_cromer

!=====
!Driver routine for integrating equations of motion
!using the Euler – Verlet method
!Input:
!Xin=X(1), Vin=V(1) — initial condition at t=0,
!Tfi the final time and Nt the number of times
!Output:
!The arrays T(1..Nt), X(1..Nt), V(1..Nt) which
!gives x(t_i)=X(i), dx/dt(t_i)=V(i), t_i=T(i) i=1..Nt
!where for i=1 we have the initial condition.
!=====
subroutine euler_verlet(Xin,Vin,Tfi,Nt,T,X,V)
  implicit none
  integer :: Nt
  real,dimension(Nt):: T,X,V !time t,x(t),v(t)=dx/dt
  real :: Xin,Vin,Tfi
  integer :: i
  real :: h,h2,X0,o2h
  real :: accel
  !Initial conditions set here:
  T(1) = 0.0
  X(1) = Xin
  V(1) = Vin
  h = Tfi/(Nt-1) ! time step
  h2 = h*h ! time step squared
  o2h = 0.5/h ! h/2
  !We have to initialize one more step: X0 corresponds to 'X(0)'
  X0 = X(1) - V(1) * h + accel(X(1)) * h2/2.0
  T(2) = h
  X(2) = 2.0*X(1) - X0 + accel(X(1)) * h2
  !Now i starts from 3:
  do i = 3,Nt
    T(i) = T(i-1)+h
    X(i) = 2.0*X(i-1) - X(i-2) + accel(X(i-1))*h2
    V(i-1) = o2h * (X(i)-X(i-2))
  enddo
  !Notice that we have one more step for the velocity:
  V(Nt) = (X(Nt)-X(Nt-1))/h
end subroutine euler_verlet

```

Η μεταγλώττιση και το τρέξιμο του προγράμματος γίνονται με τις εντολές:

```

> gfortran euler.f90 -o euler
> ./euler
Enter X_0,V_0,t_f,Nt (t_i=0):
0.2 0.0 6.0 1000
> ls euler*.dat
euler_cromer.dat  euler.dat  euler_verlet.dat
> head -n 5 euler.dat
0.000000      0.2000000      0.000000
6.0060062E-03 0.2000000      -1.1932093E-02
1.2012012E-02 0.1999283      -2.3864185E-02
1.8018018E-02 0.1997850      -3.5792060E-02
2.4024025E-02 0.1995700      -4.7711499E-02

```

Η τελευταία εντολή μας δείχνει τις 5 πρώτες γραμμές του αρχείου euler.dat όπου βλέπουμε τις 3 στήλες με το χρόνο, θέση και ταχύτητα που δίνει η μέθοδος. Για να δούμε γραφικά τα αποτελέσματα μπορούμε να χρησιμοποιήσουμε το gnuplot. Οι εντολές

```

gnuplot> plot "euler.dat" using 1:2 with lines
gnuplot> plot "euler.dat" using 1:3 with lines

```

κάνουν τις γραφικές παραστάσεις των θέσεων και ταχυτήτων αντίστοιχα συναρτήσει του χρόνου. Στην τελευταία, μπορούμε να προσθέσουμε και τα αποτελέσματα των άλλων μεθόδων δίνοντας στη συνέχεια τις εντολές:

```

gnuplot> replot "euler_cromer.dat" using 1:3 with lines
gnuplot> replot "euler_verlet.dat" using 1:3 with lines

```

Τα αποτελέσματα φαίνονται στα σχήματα 4.2–4.7. Παρατηρούμε ότι η μέθοδος Euler είναι ασταθής εκτός αν πάρουμε το βήμα χρόνου πάρα πολύ μικρό. Η μέθοδος Euler–Cromer έχει καλύτερη συμπεριφορά. Τα αποτελέσματα συγκλίνουν γρήγορα και παραμένουν σταθερά και για μεγάλο αριθμό χρονικών σημείων  $Nt \sim 100,000$ . Η μέθοδος Euler–Verlet συγκλίνει γρήγορα, αλλά σύντομα παρατηρούμε τα σφάλματα συσσώρευσης. Το φαινόμενο αυτό είναι εντονότερο για αρχικές συνθήκες με μεγάλη αρχική γωνιακή απόκλιση, όπως φαίνεται στο σχήμα 4.7. Στα σχήματα 4.2–4.4, όπου η αρχική γωνιακή απόκλιση είναι μικρή, συγκρίνουμε τη λύση που παίρνουμε με τη λύση για το αρμονικό εκκρεμές

( $\sin(\theta) \approx \theta$ ):

$$\begin{aligned}\alpha(\theta) &= -\frac{g}{l}\theta \equiv -\Omega^2\theta \\ \theta(t) &= \theta_0 \cos(\Omega t) + (\omega_0/\Omega) \sin(\Omega t) \\ \omega(t) &= \omega_0 \cos(\Omega t) - (\theta_0\Omega) \sin(\Omega t).\end{aligned}\tag{4.14}$$

Παρατηρούμε ταύτιση για τις τιμές του  $\Delta t$  που οι μέθοδοι συγκλίνουν. Με τον τρόπο αυτό ελέγχουμε τον κώδικα ως προς την ορθότητα των αποτελεσμάτων. Για το γράφημα των παραπάνω συναρτήσεων μπορούμε να δώσουμε τις παρακάτω εντολές στο πρόγραμμα gnuplot<sup>6</sup>:

```
gnuplot> set dummy t
gnuplot> omega2 = 10
gnuplot> X0 = 0.2
gnuplot> V0 = 0.0
gnuplot> omega = sqrt(omega2)
gnuplot> x(t) = X0 * cos(omega * t) +(V0/omega)*sin(omega*t)
gnuplot> v(t) = V0 * cos(omega * t) -(omega*X0)*sin(omega*t)
gnuplot> plot x(t), v(t)
```

Η σύγκριση των αποτελεσμάτων με τα θεωρητικά, ιδιαίτερα όταν οι διαφορές δεν διακρίνονται με γυμνό μάτι ή όταν η ποσοτική ανάλυση είναι επιθυμητή, μπορεί να γίνει αναπαριστώντας γραφικά τις διαφορές των αριθμητικά υπολογισμένων τιμών από τις θεωρητικές τιμές. Οι σχετικές γραφικές παραστάσεις γίνονται με τις εντολές:

```
gnuplot> plot "euler.dat" using 1:($2-x($1)) with lines
gnuplot> plot "euler.dat" using 1:($3-v($1)) with lines
```

Η εντολή using 1:(\$2-x(\$1)) σε απλά ελληνικά λέει “Κάνε τη γραφική παράσταση χρησιμοποιώντας στο άξονα των y την τιμή της 2ης στήλης του euler.dat μείον την τιμή της συνάρτησης x(t) για t ίσο με την αντίστοιχη τιμή της πρώτης στήλης”. Με τον τρόπο αυτό (και με μικρή μετατροπή για να υπολογίζουμε την απόλυτη τιμή της διαφοράς) φτιάχνουμε τα σχήματα 4.11-4.14.

### 4.3 Μέθοδοι Runge–Kutta

Στην προηγούμενη παράγραφο είδαμε μία μέθοδο πεπερασμένων διαφορών ενός βήματος πρώτης τάξης, τη μέθοδο Euler. Αυτό σημαίνει πως

<sup>6</sup>Η εντολή set dummy t κάνει την ανεξάρτητη μεταβλητή στις συναρτήσεις να είναι η t αντί για την προεπιλεγμένη x.

όταν προσεγγίζουμε την ολοκλήρωση με  $N$  διακριτά βήματα από χρόνο  $t_i$  σε χρόνο  $t_f$  με βήμα  $\Delta t \equiv h = (t_f - t_i)/N$ , το ολικό σφάλμα διακριτοποίησης είναι τάξης  $\sim \mathcal{O}(h)$ . Γεννάται το ερώτημα αν είναι δυνατόν να βρεθεί αλγόριθμος ολοκλήρωσης ο οποίος να κάνει τα σφάλματα να είναι ανώτερης τάξης. Μια κλάση τέτοιων μεθόδων είναι οι μέθοδοι Runge–Kutta. Οι μέθοδοι αυτοί είναι επαγωγικοί ενός βήματος, δηλ. η επόμενη θέση προκύπτει απλά από τη γνώση της προηγούμενης. Σε μεθόδους δύο ή πολλαπλών βημάτων, όπως είναι η μέθοδος Euler–Verlet, η γνώση της επόμενης θέσης απαιτεί να γνωρίζουμε τη θέση του σωματιδίου για δύο ή περισσότερα προηγούμενα βήματα. Η μέθοδος Runge–Kutta τάξης  $p$  έχει ολικό σφάλμα  $\sim \mathcal{O}(h^p)$ . Αυτό σημαίνει ότι σε κάθε βήμα εισάγεται σφάλμα διακριτοποίησης τάξης  $\sim \mathcal{O}(h^{p+1})$ , αφού τότε το σφάλμα μετά από  $N = (t_f - t_i)/\Delta t$  βήματα θα είναι τάξης

$$\sim \mathcal{O}(h^{p+1}) \times N = \mathcal{O}(h^{p+1}) \times \frac{t_f - t_i}{\Delta t} \sim \mathcal{O}(h^{p+1}) \times \frac{1}{h} = \mathcal{O}(h^p). \quad (4.15)$$

Ας θεωρήσουμε για απλότητα το πρόβλημα με μια άγνωστη συνάρτηση  $x(t)$  η οποία εξελίσσεται στο χρόνο σύμφωνα με τη διαφορική εξίσωση:

$$\frac{dx}{dt} = f(t, x). \quad (4.16)$$

Ας δούμε πρώτα μία μέθοδο πρώτης τάξης. Η πιο αφελής προσέγγιση θα ήταν να προσεγγίσουμε την παράγωγο από την πεπερασμένη διαφορά

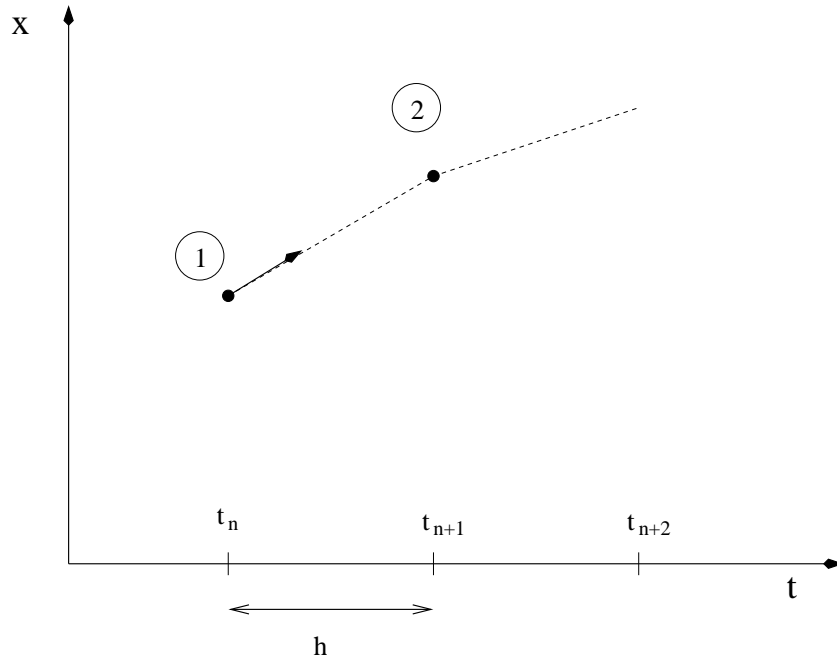
$$\frac{dx}{dt} \approx \frac{x_{n+1} - x_n}{\Delta t} = f(t_n, x_n) \Rightarrow x_{n+1} = x_n + hf(t_n, x_n) \quad (4.17)$$

Αναπτύσσοντας κατά Taylor βλέπουμε ότι το σφάλμα σε κάθε βήμα είναι  $\mathcal{O}(h^2)$ , άρα το σφάλμα για την εξέλιξη από  $t_i \rightarrow t_f$  είναι  $\mathcal{O}(h)$ . Πράγματι

$$x_{n+1} = x(t_n + h) = x_n + h \frac{dx}{dt}(x_n) + \mathcal{O}(h^2) = x_n + hf(t_n, x_n) + \mathcal{O}(h^2). \quad (4.18)$$

Η γεωμετρία του βήματος φαίνεται στο σχήμα 4.8. Επιλέγεται το σημείο 1 και από εκεί με γραμμική επέκταση στην κατεύθυνση της παραγώγου  $k_1 \equiv f(t_n, x_n)$  προσδιορίζουμε το σημείο  $x_{n+1}$ .

Βελτίωση της μεθόδου προκύπτει αν πάρουμε ένα ενδιάμεσο σημείο 2. Αυτή η διαδικασία φαίνεται στο σχήμα 4.9 και έγκειται στο να πάρουμε το ενδιάμεσο σημείο 2 στο μέσο του διαστήματος  $(t_n, t_{n+1})$  με



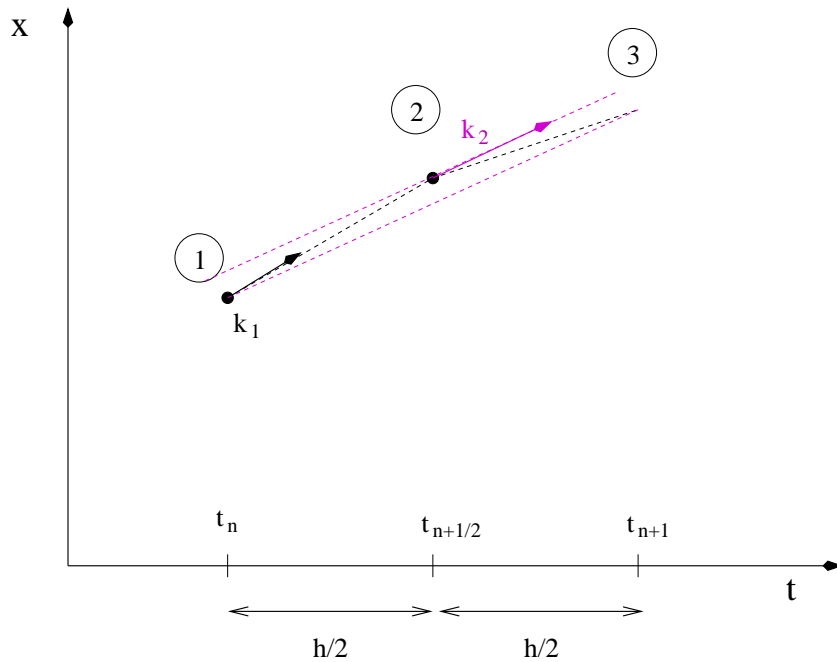
Σχήμα 4.8: Η γεωμετρία του βήματος της μεθόδου 1ης τάξης που δίνεται από την εξίσωση (4.17).

γραμμική προέκταση από το  $x_n$  χρησιμοποιώντας την κλίση που δίνεται από την παράγωγο στο  $x_n$   $k_1 \equiv f(t_n, x_n)$ . Στη συνέχεια, χρησιμοποιούμε ως εκτιμητή της παραγώγου στο διάστημα αυτό την κλίση στο σημείο 2 δηλ.  $k_2 \equiv f(t_{n+1/2}, x_{n+1/2}) = f(t_n + h/2, x_n + (h/2)k_1)$  και τη χρησιμοποιούμε για να προεκτείνουμε γραμμικά από το  $x_n$  στο  $x_{n+1}$ . Συνοψίζοντας, έχουμε

$$\begin{aligned} k_1 &= f(t_n, x_n) \\ k_2 &\equiv f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2} k_1\right) \\ x_{n+1} &= x_n + h k_2. \end{aligned} \tag{4.19}$$

Αυτό που θα δείξουμε είναι ότι παρόλο που χρειάζεται να υπολογίσουμε τη συνάρτηση  $f$  δύο φορές σε κάθε βήμα, διπλασιάζοντας ουσιαστικά τον υπολογιστικό χρόνο, το σφάλμα στο βήμα (4.19) είναι  $\mathcal{O}(h^3)$ , άρα το συνολικό σφάλμα είναι  $\mathcal{O}(h^2)$ , οπότε αναγκαστικά η (4.19) θα υπερτερήσει της (4.17) σε ακρίβεια, ακόμα και αν συγκριθούν σε δεδομένο υπολογιστικό χρόνο, δηλ. η (4.19) σε βήμα  $h$  και η (4.17) σε  $h/2$ .<sup>7</sup>

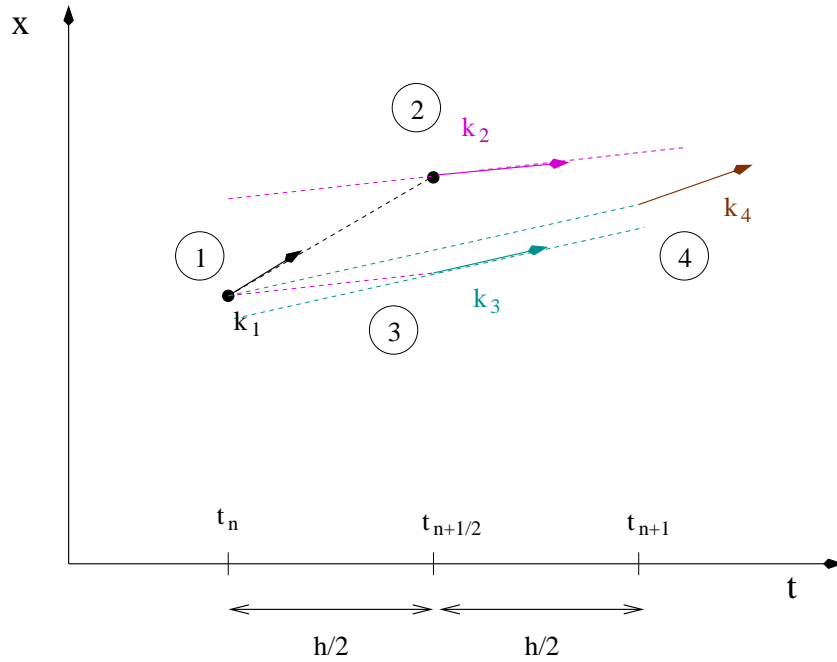
<sup>7</sup>Όπως φαίνεται στην πράξη, η (4.17) πάσχει και από προβλήματα σταθερότητας περισσότερο από την (4.19) οπότε συνίσταται να αποφεύγεται.



Σχήμα 4.9: Η γεωμετρία του βήματος της μεθόδου 2ης τάξης που δίνεται από την εξίσωση (4.19).

Η βελτίωση γίνεται περισσότερο αισθητή με τη μέθοδο Runge-Kutta 4ης τάξης. Στην περίπτωση αυτή έχουμε 4 υπολογισμούς της συνάρτησης  $f$ , αλλά το συνολικό σφάλμα είναι τώρα  $\mathcal{O}(h^4)$ , οπότε για τους ίδιους λόγους η μέθοδος θα υπερτερήσει τελικά σε ακρίβεια της (4.19)<sup>8</sup>. Η διαδικασία που θα ακολουθήσουμε εξηγείται γεωμετρικά στο σχήμα 4.10. Χρησιμοποιούμε τώρα 3 ενδιάμεσα σημεία για την προώθηση από το  $x_n$  στο  $x_{n+1}$ . Αρχικά χρησιμοποιώντας την κλίση που δίνεται από την παράγωγο στο  $x_n$   $k_1 \equiv f(t_n, x_n)$ , βρίσκουμε το ενδιάμεσο σημείο 2 στο μέσο του διαστήματος  $(t_n, t_{n+1} = t_n + h)$  δηλ.  $x_2 = x_n + (h/2)k_1$ . Υπολογίζουμε την παράγωγο της συνάρτησης στο σημείο 2 δηλ.  $k_2 \equiv f(t_n + h/2, x_n + (h/2)k_1)$  και τη χρησιμοποιούμε για να προεκτείνουμε γραμμικά από το  $x_n$  στο ενδιάμεσο σημείο 3, πάλι στο μέσο του διαστήματος  $(t_n, t_{n+1})$ , δηλ.  $x_3 = x_n + (h/2)k_2$ . Υπολογίζουμε την παράγωγο της συνάρτησης  $k_3 \equiv f(t_n + h/2, x_n + (h/2)k_2)$  και τη χρησιμοποιούμε για να προεκτείνουμε προς το σημείο 4, το οποίο τώρα το παίρνουμε στο άκρο του διαστήματος δηλ. με  $t_4 = t_n + h$ , οπότε παίρνουμε  $x_4 = x_n + hk_3$ . Κάνουμε ένα τέταρτο υπολογισμό της παραγώγου  $k_4 \equiv f(t_n + h, x_n + hk_3)$

<sup>8</sup>Πάντα; Χμμμμμ, όχι πάντα! Μεγαλύτερης τάξης δεν σημαίνει αναγκαστικά και μεγαλύτερης ακρίβειας, αν και τις περισσότερες φορές αυτό είναι σωστό.



Σχήμα 4.10: Η γεωμετρία του βήματος της μεθόδου 4ης τάξης που δίνεται από την εξίσωση (4.20).

και χρησιμοποιούμε και τις 4 παραγώγους  $k_1, k_2, k_3$  και  $k_4$  ως εκτιμητές της παραγώγου της συνάρτησης. Εκείνο που έδειξαν οι Runge–Kutta είναι ότι το σφάλμα διακριτοποίησης σε κάθε βήμα στην προώθηση της συνάρτησης γίνεται  $\mathcal{O}(h^5)$ , αν πάρουμε:

$$\begin{aligned}
 k_1 &= f(t_n, x_n) \\
 k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2} k_1\right) \\
 k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2} k_2\right) \\
 k_4 &= f(t_n + h, x_n + h k_3) \\
 x_{n+1} &= x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \quad (4.20)
 \end{aligned}$$

Ο δεύτερος όρος στην τελευταία εξίσωση είναι ένας μέσος όρος των 4 παραγώγων με το κατάλληλο βάρος, ώστε να πετύχουμε την εξουδετέρωση των σφαλμάτων μέχρι τάξης  $h^5$ .

Τέλος θα κλείσουμε συζητώντας μία απορία που πιθανώς θα έχει δημιουργηθεί στον αναγνώστη: Συζητήσαμε πώς είναι δυνατόν κανείς να μειώσει τα σφάλματα διακριτοποίησης χρησιμοποιώντας αλγόριθμους



των οποίων τα σφάλματα αυτά μειώνονται κατά το δυνατόν γρηγορότερα με το βήμα στο χρόνο  $h$ , συνήθως σαν  $\sim h^p$ . Άρα, μπορεί κανείς να υποθέσει ότι CPU χρόνου επιτρέποντος -το οποίο δεν αποτελεί σοβαρό εμπόδιο στις απλές περιπτώσεις- μπορούμε να πλησιάσουμε αυθαίρετα κοντά στην αναλυτική λύση σε επίπεδο ακρίβειας μηχανής. Αυτό όμως δεν είναι σωστό. Μια άλλη κατηγορία σφαλμάτων είναι τα σφάλματα στρογγυλοποίησης τα οποία προστίθενται σε κάθε βήμα εφαρμογής της μεθόδου. Αυτά συσσωρεύονται ανάλογα με τον αριθμό των βημάτων, οπότε για πολύ μικρό  $h$ , άρα και πολύ μεγάλο αριθμό βημάτων, αυτά θα γίνουν μεγαλύτερα από την επιθυμητή ακρίβεια. Αυτή η κατηγορία σφαλμάτων εξαρτάται από το hardware, τη γλώσσα προγραμματισμού ή/και το μεταγλωττιστή και τέλος, από τον αλγόριθμο. Για το τελευταίο ας δώσουμε ένα παράδειγμα: Έστω ότι θέλουμε να υπολογίσουμε την παράγωγο μιας συνάρτησης από τη σχέση

$$f'(t) = \frac{f(t+h) - f(t)}{h},$$

παίρνοντας το  $h$  αυθαίρετα μικρό. Αν υποθέσουμε ότι η παράγωγος και οι τιμές της συνάρτησης είναι πεπερασμένοι αριθμοί  $\sim \mathcal{O}(1)$ , τότε ο αριθμητής θα πρέπει να είναι  $\sim \mathcal{O}(h)$ . Όταν το  $h$  γίνει της τάξης της ακρίβειας των REAL (ή των REAL(8) κλπ), τότε ο αριθμητής που είναι η διαφορά δύο αριθμών περίπου ίσων και της τάξης της μονάδας θα αρχίσει να χάνει σημαντικά σε ακρίβεια σε σχέση με την πραγματική τιμή μέχρι που θα είναι ένας άχρηστος αριθμός. Αυτό συμβαίνει γενικά όταν αφαιρούμε αριθμούς περίπου ίσους ή όταν προσθέτουμε αριθμούς που η τάξη μεγέθους τους διαφέρει περισσότερο από την ακρίβεια του υπολογιστή. Άρα και όταν χρησιμοποιούμε τη σχέση

$$x_{n+1} = x_n + h\mathcal{O}(x_n)$$

για πολύ μικρό βήμα  $h$  και  $\mathcal{O}(x)$  της τάξης μεγέθους του  $x_n$ , τα σφάλματα στρογγυλοποίησης θα είναι σημαντικά και θα αυξάνουν προσθετικά με τον αριθμό των βημάτων.

#### 4.3.1 Προγραμματισμός της Runge–Kutta 4ης τάξης

Ας συζητήσουμε τώρα τον προγραμματισμό της μεθόδου Runge–Kutta 4ης τάξης για την περίπτωση της κίνησης ενός σωματιδίου σε μία διάσταση. Για το λόγο αυτό θα πρέπει να ολοκληρώσουμε το σύστημα διαφορικών εξισώσεων (4.5) που είναι ένα σύστημα εξισώσεων για τις

δύο συναρτήσεις του χρόνου  $x_1(t) \equiv x(t)$  και  $x_2(t) \equiv v(t)$  για τις οποίες έχουμε

$$\frac{dx_1}{dt} = f_1(t, x_1, x_2) \quad \frac{dx_2}{dt} = f_2(t, x_1, x_2) \quad (4.21)$$

Στην περίπτωση αυτή, η μέθοδος Runge–Kutta 4ης τάξης που δίνεται στην εξίσωση (4.20) γενικεύεται ως εξής:

$$\begin{aligned} k_{11} &= f_1(t_n, x_{1,n}, x_{2,n}) \\ k_{21} &= f_2(t_n, x_{1,n}, x_{2,n}) \\ k_{12} &= f_1\left(t_n + \frac{h}{2}, x_{1,n} + \frac{h}{2}k_{11}, x_{2,n} + \frac{h}{2}k_{21}\right) \\ k_{22} &= f_2\left(t_n + \frac{h}{2}, x_{1,n} + \frac{h}{2}k_{11}, x_{2,n} + \frac{h}{2}k_{21}\right) \\ k_{13} &= f_1\left(t_n + \frac{h}{2}, x_{1,n} + \frac{h}{2}k_{12}, x_{2,n} + \frac{h}{2}k_{22}\right) \\ k_{23} &= f_2\left(t_n + \frac{h}{2}, x_{1,n} + \frac{h}{2}k_{12}, x_{2,n} + \frac{h}{2}k_{22}\right) \\ k_{14} &= f_1(t_n + h, x_{1,n} + h k_{13}, x_{2,n} + h k_{23}) \\ k_{24} &= f_2(t_n + h, x_{1,n} + h k_{13}, x_{2,n} + h k_{23}) \\ x_{1,n+1} &= x_{1,n} + \frac{h}{6}(k_{11} + 2k_{12} + 2k_{13} + k_{14}) \\ x_{2,n+1} &= x_{1,n} + \frac{h}{6}(k_{21} + 2k_{22} + 2k_{23} + k_{24}). \end{aligned} \quad (4.22)$$

Ο προγραμματισμός της μεθόδου είναι απλός. Στο κυρίως πρόγραμμα έχουμε απλό interface με το χρήστη και του ζητάμε τα απαραίτητα δεδομένα: Το χρόνο ολοκλήρωσης από  $t_i = T_i$  έως  $t_f = T_f$  και τον αριθμό των χρονικών σημείων  $N_t$ . Τις αρχικές συνθήκες  $x_1(t_i) = X_{10}$ ,  $x_2(t_i) = X_{20}$ . Η δομή των δεδομένων είναι απλή: Τρία arrays  $T(P)$ ,  $X_1(P)$ ,  $X_2(P)$  αποθηκεύουν τις τιμές του χρόνου  $t_i \equiv t_1, t_2, \dots, t_{N_t} \equiv t_f$  και τις αντίστοιχες τιμές των συναρτήσεων  $x_1(t_k)$  και  $x_2(t_k)$ ,  $k = 1, \dots, N_t$ . Το πρόγραμμα καλεί την υπορουτίνα  $RK(T, X_1, X_2, T_i, T_f, X_{10}, X_{20}, N_t)$  η οποία είναι ο οδηγός της μεθόδου, δηλ. οδηγεί την καρδιά του προγράμματος την υπορουτίνα  $RKSTEP(t, x_1, x_2, dt)$  η οποία εφαρμόζει τους τύπους (4.22) και προωθεί τις τιμές των συναρτήσεων  $x_1, x_2$  τη χρονική στιγμή  $t$  κατά ένα βήμα  $h = dt$ . Κάθε βήμα, αφού οριστούν οι αρχικές συνθήκες καταγράφεται στην RK στα arrays  $T, X_1$  και  $X_2$ . Όταν η RK επιστρέφει τον έλεγχο στο κυρίως πρόγραμμα, τα αποτελέσματα είναι καταχωρημένα στα  $T, X_1$  και  $X_2$ , τα οποία τυπώνονται στο αρχείο `rk.dat`. Παρακάτω παραθέτουμε το πρόγραμμα για να διευκολύνουμε τον αναγνώστη:

```

!=====
!Program to solve a 2 ODE system using Runge–Kutta Method
!User must supply derivatives
!dx1/dt=f1(t,x1,x2) dx2/dt=f2(t,x1,x2)
!as real functions
!Output is written in file rk.dat
!=====
program rk_solve
  implicit none
  integer , parameter :: P=110000
  real , dimension(P) :: T,X1,X2
  real :: Ti,Tf,X10,X20
  integer :: Nt
  integer :: i
!Input:
  print *, 'Runge–Kutta Method for 2–ODEs Integration '
  print *, 'Enter Nt,Ti,TF,X10,X20: '
  read *, Nt,Ti,Tf,X10,X20
  print *, 'Nt = ',Nt
  print *, 'Time: Initial Ti =',Ti, ' Final Tf=',Tf
  print *, '          X1(Ti)=',X10, ' X2(Ti)=',X20
  if(Nt.gt.P) stop 'Nt>P'
!The Calculation:
  call RK(T,X1,X2,Ti,Tf,X10,X20,Nt)
!Output:
  open(unit=11,file='rk.dat')
  do i=1,Nt
    write(11,*)T(i),X1(i),X2(i)
  enddo
  close(11)

end program rk_solve
!=====
!The functions f1,f2(t,x1,x2) provided by the user
!=====
real function f1(t,x1,x2)
  implicit none
  real :: t,x1,x2
  f1=x2 !dx1/dt= v = x2
end function f1
!
real function f2(t,x1,x2)
  implicit none
  real :: t,x1,x2
  f2=-10.0D0*x1 !harmonic oscillator
end function f2
!=====
!RK(T,X1,X2,Ti,Tf,X10,X20,Nt) is the driver
!for the Runge–Kutta integration routine RKSTEP

```

```

!Input: Initial and final times Ti,Tf
!      Initial values at t=Ti X10,X20
!      Number of steps of integration: Nt-1
!      Size of arrays T,X1,X2
!Output: real arrays T(Nt),X1(Nt),X2(Nt) where
!T(1) = Ti X1(1) = X10 X2(1) = X20
!      X1(k) = X1(at t=T(k)) X2(k) = X2(at t=T(k))
!T(Nt)=TF
!=====
subroutine RK(T,X1,X2,Ti,Tf,X10,X20,Nt)
  implicit none
  integer :: Nt
  real,dimension(Nt):: T,X1,X2
  real :: Ti,Tf,X10,X20
  real :: dt
  real :: TS,X1S,X2S !values of time and X1,X2 at given step
  integer :: i
!Initialize variables:
  dt = (Tf-Ti)/(Nt-1)
  T(1) = Ti
  X1(1) = X10
  X2(1) = X20
  TS = Ti
  X1S = X10
  X2S = X20
!Make RK steps: The arguments of RKSTEP
!are replaced with the new ones!
  do i=2,Nt
    call RKSTEP(TS,X1S,X2S,dt)
    T(i) = TS
    X1(i) = X1S
    X2(i) = X2S
  enddo
end subroutine RK
!=====
!Subroutine RKSTEP(t,x1,x2,dt)
!Runge-Kutta Integration routine of ODE
!dx1/dt=f1(t,x1,x2) dx2/dt=f2(t,x1,x2)
!User must supply derivative functions:
!real function f1(t,x1,x2)
!real function f2(t,x1,x2)
!Given initial point (t,x1,x2) the routine advances it
!by time dt.
!Input : Initial time t and function values x1,x2
!Output: Final time t+dt and function values x1,x2
!Careful!: values of t,x1,x2 are overwritten...
!=====
subroutine RKSTEP(t,x1,x2,dt)
  implicit none

```

```

real :: t,x1,x2,dt
real :: f1,f2
real :: k11,k12,k13,k14,k21,k22,k23,k24
real :: h,h2,h6

h =dt      !h =dt, integration step
h2 =0.5D0*h !h2=h/2
h6 =h/6.0   !h6=h/6

k11=f1(t,x1,x2)
k21=f2(t,x1,x2)
k12=f1(t+h2,x1+h2*k11,x2+h2*k21)
k22=f2(t+h2,x1+h2*k11,x2+h2*k21)
k13=f1(t+h2,x1+h2*k12,x2+h2*k22)
k23=f2(t+h2,x1+h2*k12,x2+h2*k22)
k14=f1(t+h ,x1+h *k13,x2+h *k23)
k24=f2(t+h ,x1+h *k13,x2+h *k23)

t =t+h
x1 =x1+h6*(k11+2.0D0*(k12+k13)+k14)
x2 =x2+h6*(k21+2.0D0*(k22+k23)+k24)

end subroutine RKSTEP

```

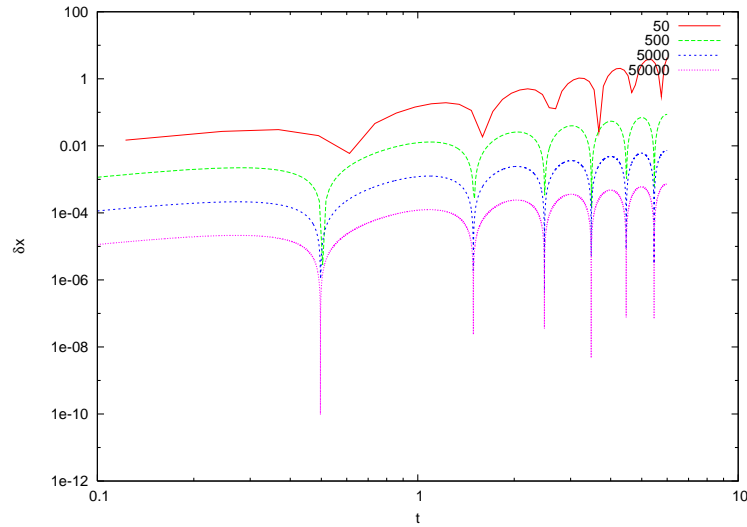
## 4.4 Σύγκριση των Μεθόδων

Στην παράγραφο αυτή θα κάνουμε έλεγχο ορθότητας των προγραμμάτων μας και θα μετρήσουμε την ακρίβειά τους ως προς τα ολικά σφάλματα διακριτοποίησης. Το πιο απλό τεστ στο οποίο μπορούμε να τα υποβάλλουμε είναι να συγκρίνουμε τα αριθμητικά αποτελέσματα σε ένα σύστημα για το οποίο έχουμε γνωστή την αναλυτική λύση. Θα διαλέξουμε να το κάνουμε για την περίπτωση του απλού αρμονικού ταλαντωτή. Οι αλλαγές που θα κάνουμε στα προγράμματα είναι μικρές και θα τις αναφέρουμε περιληπτικά. Κατ' αρχήν, μετατρέπουμε όλες τις μεταβλητές REAL σε διπλής ακρίβειας REAL(8). Απλά αλλάζουμε τις δηλώσεις στα κατάλληλα σημεία του προγράμματος και προσθέτουμε ένα D0 σε όλες τις σταθερές (λ.χ. 0.5  $\rightarrow$  0.5D0 κλπ). Στη συνέχεια, μετατρέπουμε τις συναρτήσεις της επιτάχυνσης σε αυτή του αρμονικού ταλαντωτή  $a = -\omega^2 x$  και παίρνουμε  $\omega^2 = 10$  ( $T \approx 1.987$ ). Έτσι, στο πρόγραμμα euler.f90 έχουμε

```

real(8) function accel(x)
implicit none

```



Σχήμα 4.11: Η απόκλιση της αριθμητικής λύσης με τη μέθοδο Euler από την αναλυτική για τον απλό αρμονικό ταλαντωτή. Οι παράμετροι είναι  $\omega^2 = 10$ ,  $t_i = 0$ ,  $t_f = 6$ ,  $x(0) = 0.2$ ,  $v(0) = 0$  και ο αριθμός των βημάτων είναι  $N = 50, 500, 5,000, 50,000$ . Παρατηρούμε ότι κάθε φορά το σφάλμα περίπου υποδεκαπλασιάζεται σύμφωνα με την αναμενόμενη ακρίβεια της μεθόδου  $\sim \mathcal{O}(\Delta t)$ .

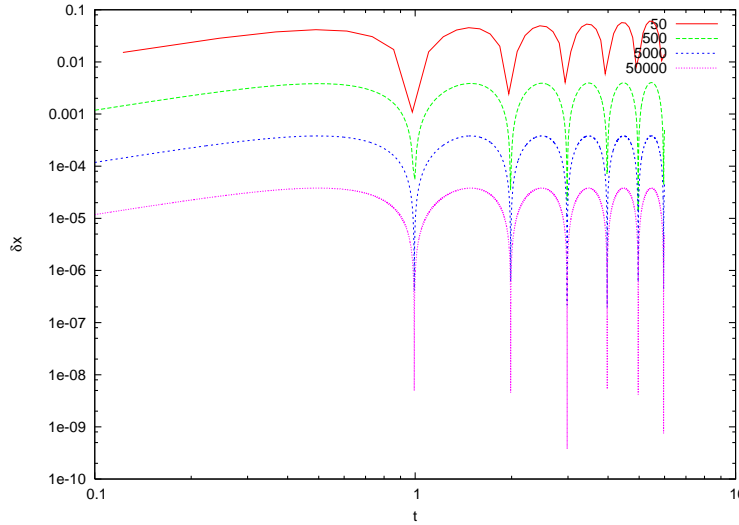
```
real(8) :: x
accel = -10.0D0*x
end function accel
```

ενώ στο πρόγραμμα rk.f90 έχουμε

```
real(8) function f2(t,x1,x2)
implicit none
real(8) :: t,x1,x2
f2=-10.0D0*x1
end function f2
```

Στη συνέχεια, τρέχουμε τα προγράμματα για δεδομένο χρονικό διάστημα από  $t_i = 0$  σε  $t_f = 6$  με αρχικές συνθήκες  $x_0 = 0.2$ ,  $v_0 = 0$  και μεταβάλλουμε το χρονικό βήμα  $\Delta t$  αλλάζοντας τον αριθμό των βημάτων  $N_{t-1}$ . Στη συνέχεια, συγκρίνουμε τη λύση που παίρνουμε με αυτή του απλού αρμονικού ταλαντωτή

$$\begin{aligned} a(x) &= -\omega^2 x \\ x_h(t) &= x_0 \cos(\omega t) + (v_0/\omega) \sin(\omega t) \\ v_h(t) &= v_0 \cos(\omega t) - (x_0 \omega) \sin(\omega t), \end{aligned} \quad (4.23)$$



Σχήμα 4.12: Όπως στο σχήμα 4.11 για τη μέθοδο Euler-Cromer. Με κάθε δεκαπλασιασμό του αριθμού βημάτων, το σφάλμα περίπου υποδεκαπλασιάζεται σύμφωνα με την αναμενόμενη ακρίβεια της μεθόδου  $\sim \mathcal{O}(\Delta t)$ .

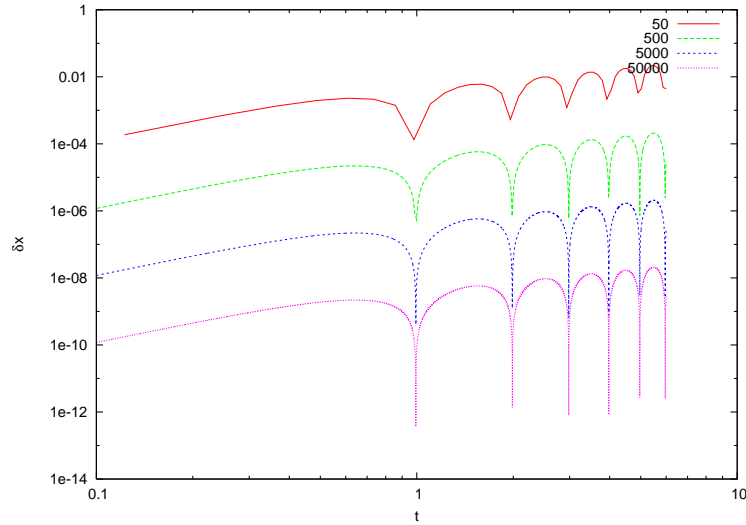
και μελετούμε τη σχέση των αποκλίσεων  $\delta x(t) = |x(t) - x_h(t)|$  και  $\delta v(t) = |v(t) - v_h(t)|$  με το βήμα  $\Delta t$ . Τα αποτελέσματά μας φαίνονται στα σχήματα 4.11–4.14. Παρατηρούμε πως για τις μεθόδους Euler και Euler-Cromer τα σφάλματα είναι τάξης  $\mathcal{O}(\Delta t)$  όπως είχαμε προβλέψει, αλλά για τη δεύτερη τα σφάλματα είναι αριθμητικά μικρότερα από αυτά της πρώτης. Για τη μέθοδο Euler-Verlet το σφάλμα βρίσκεται να είναι τάξης  $\mathcal{O}(\Delta t^2)$ , ενώ για την Runge-Kutta είναι τάξης<sup>9</sup>  $\mathcal{O}(\Delta t^4)$ .

Μία άλλη μέθοδος για να ελέγξουμε την ορθότητα των αποτελεσμάτων μας είναι να βρούμε μια διατηρούμενη ποσότητα όπως η ενέργεια, ορμή, στροφορμή κλπ και να εξετάζουμε αν αυτή αποκλίνει από την αρχική της τιμή. Στην περίπτωσή μας υπολογίζουμε τη μηχανική ενέργεια

$$E = \frac{1}{2}mv^2 + \frac{1}{2}m\omega^2 x^2 \quad (4.24)$$

σε κάθε βήμα και από αυτή την απόκλιση  $\delta E = |E - E_0|$ . Τα αποτελέσματα φαίνονται στα σχήματα 4.15–4.18.

<sup>9</sup>Επιβεβαιώστε τους ισχυρισμούς αυτούς, αρχικά από τα σχήματα 4.11-4.14 και μετά με αριθμητικό υπολογισμό. Διαλέξτε, λ.χ. μια συγκεκριμένη τιμή του χρόνου  $t$  και κάνετε τη γραφική παράσταση των σφαλμάτων συναρτήσει του  $\Delta t$ ,  $\Delta t^2$  και  $\Delta t^4$  αντίστοιχα.



Σχήμα 4.13: Όπως στο σχήμα 4.11 για τη μέθοδο Euler-Verlet. Με κάθε δεκαπλασιασμό του αριθμού βημάτων, το σφάλμα περίπου υποεκατονταπλασιάζεται σύμφωνα με την αναμενόμενη ακρίβεια της μεθόδου  $\sim \mathcal{O}(\Delta t^2)$ .

## 4.5 Ταλαντώσεις με Απόσβεση και Διέγερση

Στην παράγραφο αυτή θα μελετήσουμε τον απλό αρμονικό ταλαντωτή του οποίου η κίνηση υπόκειται σε τριβή ανάλογη της ταχύτητάς του και σε εξωτερική δύναμη, την οποία για απλότητα θα πάρουμε να έχει ημιτονοειδή εξάρτηση από το χρόνο.

$$\frac{d^2x}{dt^2} + \gamma \frac{dx}{dt} + \omega_0^2 x = a_0 \sin \omega t, \quad (4.25)$$

με  $F(t) = ma_0 \sin \omega t$  και  $\omega$  η κυκλική συχνότητα της οδηγούσας δύναμης.

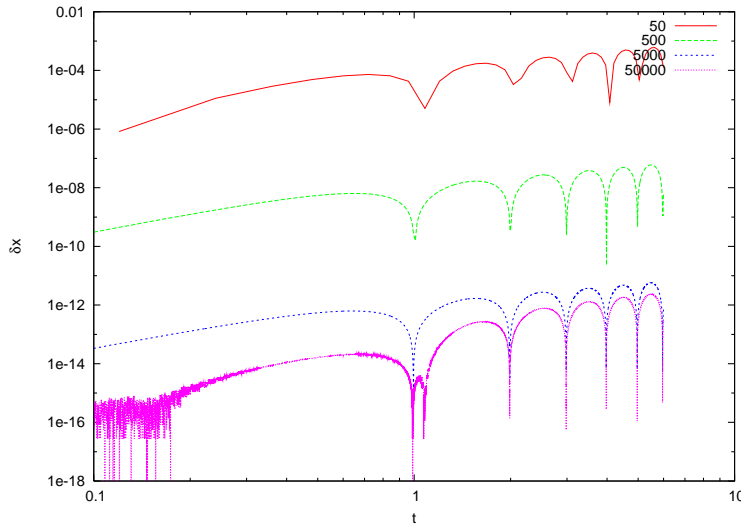
Ας θεωρήσουμε αρχικά το σύστημα με  $a_0 = 0$ . Οι πραγματικές λύσεις της διαφορικής εξίσωσης<sup>10</sup> που είναι πεπερασμένες για  $t \rightarrow +\infty$  δίνονται, διακρίνοντας τις περιπτώσεις,

$$x_0(t) = c_1 e^{-(\gamma + \sqrt{\gamma^2 - 4\omega_0^2})t/2} + c_2 e^{-(\gamma - \sqrt{\gamma^2 - 4\omega_0^2})t/2}, \quad \gamma^2 - 4\omega_0^2 > 0, \quad (4.26)$$

$$x_0(t) = c_1 e^{-\gamma t/2} + c_2 e^{-\gamma t/2} t, \quad \gamma^2 - 4\omega_0^2 = 0, \quad (4.27)$$

<sup>10</sup> Προκύπτουν εύκολα αντικαθιστώντας τη δοκιμαστική λύση  $x(t) = Ae^{-\Omega t}$  και λύνοντας ως προς  $\Omega$ .





Σχήμα 4.14: Όπως στο σχήμα 4.11 για τη μέθοδο Runge-Kutta 4ης τάξης. Με κάθε δεκαπλασιασμό του αριθμού βημάτων, το σφάλμα περίπου μειώνεται κατά  $10^{-4}$  σύμφωνα με την αναμενόμενη ακρίβεια της μεθόδου  $\sim O(\Delta t^4)$ . Στα 50,000 βήματα γίνεται φανερά τα σφάλματα στρογγυλοποίησης.

$$x_0(t) = c_1 e^{-\gamma t/2} \cos\left(\sqrt{-\gamma^2 + 4\omega_0^2} t/2\right) + c_2 e^{-\gamma t/2} \sin\left(\sqrt{-\gamma^2 + 4\omega_0^2} t/2\right), \quad \gamma^2 - 4\omega_0^2 < 0 \quad (4.28)$$

Στην τελευταία περίπτωση η λύση ταλαντώνεται με πλάτος που φθίνει εκθετικά με το χρόνο.

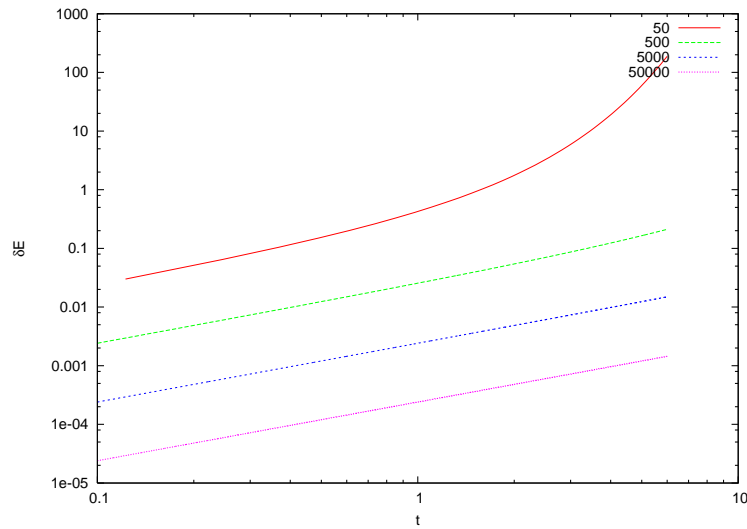
Για την περίπτωση που  $a_0 > 0$ , η γενική λύση προκύπτει από το άθροισμα μιας ειδικής λύσης  $x_s(t)$  και της λύσης της ομογενούς εξίσωσης  $x_0(t)$ . Μια ειδική λύση προκύπτει από τη δοκιμαστική λύση  $x_s(t) = A \sin \omega t + B \cos \omega t$  την οποία αντικαθιστούμε στην (4.25) και λύνουμε για τα  $A$  και  $B$ . Βρίσκουμε ότι

$$x_s(t) = \frac{a_0 [(\omega_0^2 - \omega^2) \cos \omega t + \gamma \omega \sin \omega t]}{(\omega_0^2 - \omega^2)^2 + \omega^2 \gamma^2} \quad (4.29)$$

και

$$x(t) = x_0(t) + x_s(t). \quad (4.30)$$

Η λύση  $x_0(t)$  είναι εκθετικά φθίνουσα με το χρόνο και τελικά επικρατεί η  $x_s(t)$ . Η μόνη περίπτωση που αυτό δεν ισχύει είναι στην περίπτωση του συντονισμού χωρίς απόσβεση  $\omega = \omega_0$ ,  $\gamma = 0$ . Η λύση στην περίπτωση



Σχήμα 4.15: Όπως στο σχήμα 4.11 στην περίπτωση της ενέργειας για τη μέθοδο Euler.

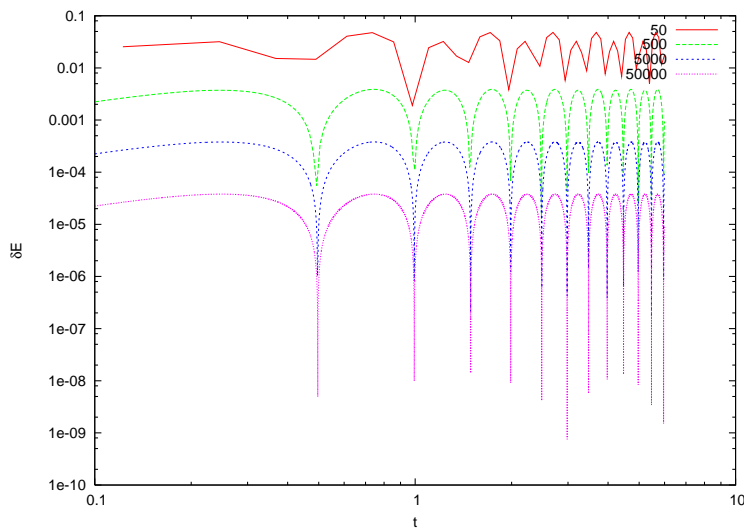
αυτή βρίσκεται εύκολα να είναι η

$$x(t) = c_1 \cos \omega t + c_2 \sin \omega t + \frac{a_0}{4\omega^2} (\cos \omega t + 2(\omega t) \sin \omega t) . \quad (4.31)$$

Οι δύο πρώτοι όροι είναι αυτοί του απλού αρμονικού ταλαντωτή, ενώ ο τελευταίος όρος αυξάνει το πλάτος της μετατόπισης γραμμικά με το χρόνο καταδεικνύοντας τη συνεχή ροή ενέργειας από την εξωτερική δύναμη στον ταλαντωτή.

Ο προγραμματισμός του συστήματος γίνεται με απλή μετατροπή του κώδικα rk.f90. Οι βασικές ρουτίνες RK(T,X1,X2,T0,TF,X10,X20,Nt) και RKSTEP(t,x1,x2,dt) μένουν ως έχουν και αλλάζει απλά το interface με το χρήστη. Εισάγουμε τις βασικές παραμέτρους  $\omega_0$ ,  $\omega$ ,  $\gamma$ ,  $a_0$  διαδραστικά στην καθιερωμένη είσοδο (standard input). Επειδή θέλουμε να τις χρησιμοποιήσουμε στη συνάρτηση f2(t,x1,x2) που δίνει την επιτάχυνση χωρίς να αλλάξουμε τη δομή των οδηγιών της μεθόδου Runge-Kutta, πρέπει να τις ορίσουμε σε κοινή θέση στη μνήμη για το κυρίως πρόγραμμα και την εν λόγω συνάρτηση. Αυτό στη Fortran γίνεται με τη χρήση COMMON BLOCKS. Αυτά ορίζονται μετά τη δήλωση των μεταβλητών και ορίζουν για αυτές μια συγκεκριμένη θέση στη μνήμη στις οποίες αποθηκεύονται οι τιμές τους. Το κομμάτι κώδικα

```
real(8) :: omega_0, omega, gamma, a_0, omega_02, omega2
common /params/ omega_0, omega, gamma, a_0, omega_02, omega2
```



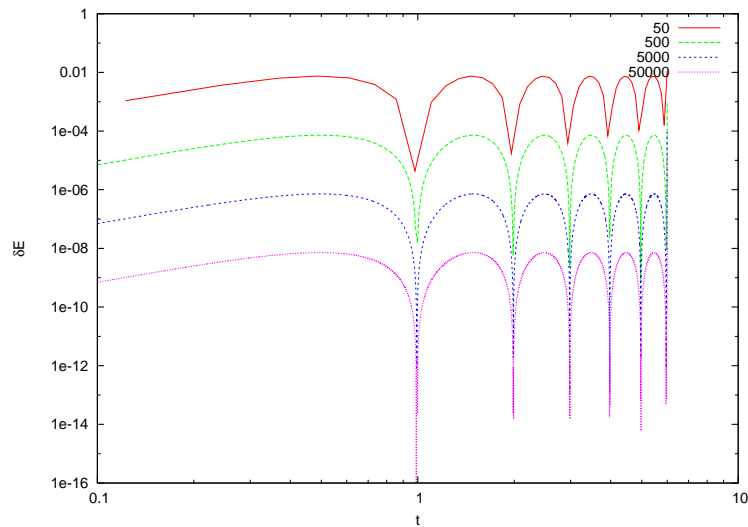
Σχήμα 4.16: Όπως στο σχήμα 4.11 στην περίπτωση της ενέργειας για τη μέθοδο Euler-Cromer.

σε οποιαδήποτε ρουτίνα δίνει πρόσβαση στη μνήμη στη “θέση” `params` στην οποία αποθηκεύονται οι τιμές των μεταβλητών. Το μόνο άλλο σημείο που χρειάζεται προσοχή στο πρόγραμμα είναι η συνάρτηση της επιτάχυνσης  $f_2(t, x_1, x_2)$  η οποία τώρα έχει όρο που εξαρτάται από την ταχύτητα που εδώ συμβολίζουμε με τη μεταβλητή  $x_2$ :

```
real(8) function f2(t,x1,x2)
  implicit none
  real(8) omega_0, omega, gamma, a_0, omega_02, omega2
  common /params/ omega_0, omega, gamma, a_0, omega_02, omega2
  real(8) t, x1, x2, a
  a = a_0*cos(omega*t)
  f2=-omega_02*x1-gamma*x2+a
end function f2
```

Για διευκόλυνση του αναγνώστη παραθέτουμε όλο το interface, παραλείποντας φυσικά τις υπορουτίνες RK, RKSTEP που έχουμε παραθέσει πρωτύτερα. Το πρόγραμμα αποθηκεύεται στο αρχείο `dlo.f90`.

```
!=====
!Program to solve Damped Linear Oscillator
!using 4th order Runge-Kutta Method
!Output is written in file dlo.dat
!=====
```

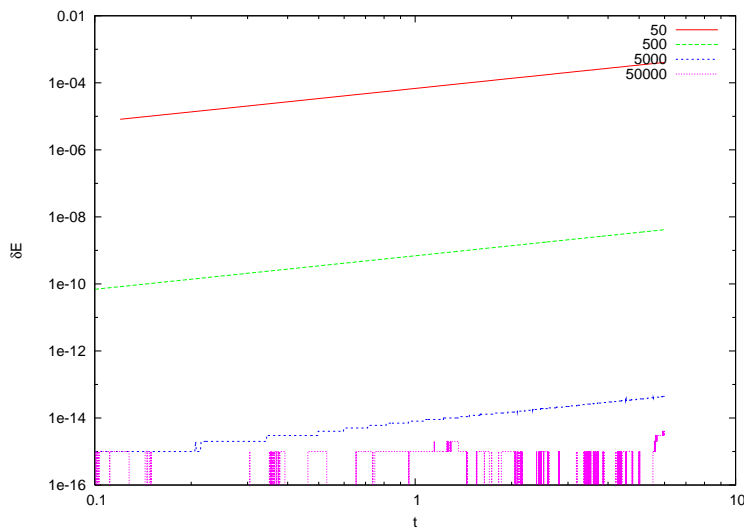


Σχήμα 4.17: Όπως στο σχήμα 4.11 στην περίπτωση της ενέργειας για τη μέθοδο Euler-Verlet.

```

program dlo_solve
  implicit none
  integer, parameter :: P=110000
  real(8), dimension(P) :: T,X1,X2
  real(8) :: Ti,Tf,X10,X20
  real(8) :: Energy
  real(8) :: omega_0,omega,gamma,a_0,omega_02,omega2
  common /params/omega_0,omega,gamma,a_0,omega_02,omega2
  integer :: Nt, i
!Input:
  print *, 'Runge-Kutta Method for DLO Integration'
  print *, 'Enter omega_0, omega, gamma, a_0:'
  read *, omega_0,omega,gamma,a_0
  omega_02 = omega_0*omega_0
  omega2 = omega *omega
  print *, 'omega_0= ',omega_0,' omega= ', omega
  print *, 'gamma= ',gamma,' a_0= ',a_0
  print *, 'Enter Nt,Ti,Tf,X10,X20:'
  read *, Nt,Ti,Tf,X10,X20
  print *, 'Nt = ',Nt
  print *, 'Time: Initial Ti =',Ti,' Final Tf=',Tf
  print *, 'X1(Ti)=',X10,' X2(Ti)=',X20
  if(Nt.gt.P) stop 'Nt>P'
!The Calculation:
  call RK(T,X1,X2,Ti,Tf,X10,X20,Nt)
!Output:
  open(unit=11,file='dlo.dat')

```

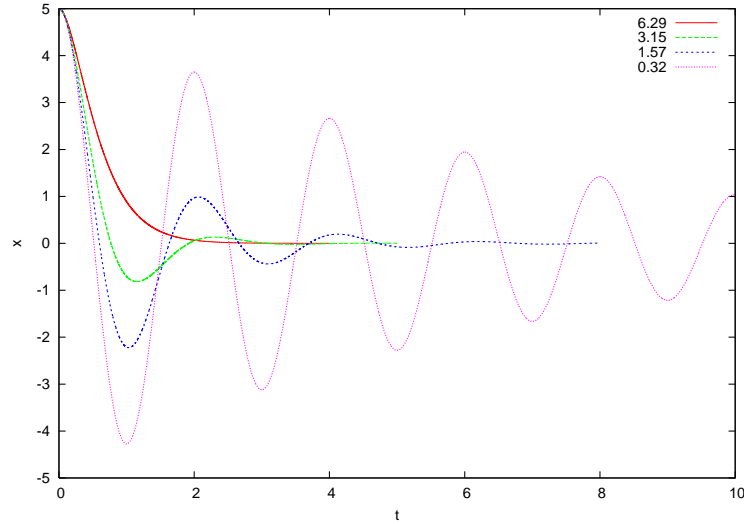


Σχήμα 4.18: Όπως στο σχήμα 4.11 στην περίπτωση της ενέργειας για τη μέθοδο Runge-Kutta 4ης τάξης. Για μεγάλο αριθμό βημάτων το σφάλμα είναι σφάλμα στρογγυλοποίησης.

```

write(11,*) '# Damped Linear Oscillator - dlo'
write(11,*) '# omega_0= ', omega_0, ' omega= ', omega, &
    ' gamma= ', gamma, ' a_0= ', a_0
do i=1,Nt
    Energy = 0.5D0*X2(i)*X2(i)+0.5D0*omega_02*X1(i)*X1(i)
    write(11,*) T(i), X1(i), X2(i), Energy
enddo
close(11)
end program dlo_solve
!=====
!The functions f1,f2(t,x1,x2) provided by the user
!=====
real(8) function f1(t,x1,x2)
    implicit none
    real(8) t,x1,x2
    f1=x2          !dx1/dt= v = x2
end function f1
!-----
real(8) function f2(t,x1,x2)
    implicit none
    real(8) omega_0, omega, gamma, a_0, omega_02, omega2
    common /params/ omega_0, omega, gamma, a_0, omega_02, omega2
    real(8) t,x1,x2,a
    a = a_0*cos(omega*t)
    f2=-omega_02*x1-gamma*x2+a
end function f2

```



Σχήμα 4.19: Η θέση συναρτήσει του χρόνου για τον αρμονικό ταλαντωτή με απόσβεση για διαφορετικές τιμές του συντελεστή απόσβεσης  $\gamma$  με  $\omega_0 = 3.145$ .

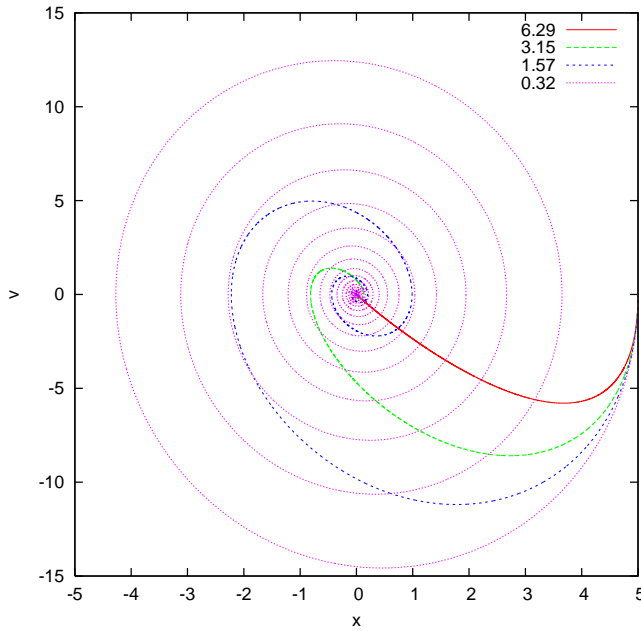
Τα αποτελέσματα φαίνονται στα σχήματα 4.19–4.22. Στο σχήμα 4.19 παρατηρούμε τη μετάβαση από την φάση που η κίνηση αποσβένεται χωρίς ταλάντωση για  $\gamma > 2\omega_0$  στη φάση που το σύστημα ταλαντώνεται με εκθετικά φθίνων με το χρόνο πλάτος για  $\gamma < 2\omega_0$ . Η εκθετική μείωση του πλάτους φαίνεται στο σχήμα 4.21, ενώ η εξάρτηση της περιόδου  $T$  από το συντελεστή απόσβεσης  $\gamma$  στο σχήμα 4.22. Το εν λόγω σχήμα προκύπτει από τη Σχέση (4.28) την οποία γράφουμε στη μορφή

$$4\omega_0^2 - \left(\frac{2\pi}{T}\right)^2 = \gamma^2. \quad (4.32)$$

Το δεξί μέλος της εξίσωσης τοποθετείται στον οριζόντιο άξονα, ενώ στον κάθετο τοποθετούμε το αριστερό. Η παραπάνω σχέση προβλέπει ότι οι δύο ποσότητες είναι ίσες και οι μετρήσεις πρέπει να βρίσκονται πάνω στη διαγώνιο  $y = x$ . Οι μετρήσεις για την “περίοδο”  $T$  παίρνονται μετρώντας το χρόνο μεταξύ δύο διαδοχικών ακρότατων ( $v = 0$ ) στην τροχιά  $x(t)$  (βλ. σχήμα 4.19).

Τέλος, σημαντικό είναι να μελετήσουμε το σχήμα 4.20 στο οποίο βλέπουμε την τροχιά του συστήματος στο χώρο των φάσεων<sup>11</sup> Το σύστημα για  $t \rightarrow +\infty$  καταλήγει στο σημείο  $(0, 0)$  για οποιαδήποτε τιμή

<sup>11</sup>Για την ακρίβεια ο χώρος των φάσεων είναι ο χώρος των θέσεων–ορμών, αλλά στην περίπτωσή μας η διαφορά είναι τετριμμένη.



Σχήμα 4.20: Η τροχιά στο χώρο των φάσεων για τον αρμονικό ταλαντωτή με απόσβεση για διαφορετικές τιμές του συντελεστή απόσβεσης  $\gamma$  με  $\omega_0 = 3.145$ . Παρατηρούμε την ύπαρξη ελκυστή στο  $(x, v) = (0, 0)$  στον οποίο καταλήγει το σύστημα για  $t \rightarrow +\infty$ .

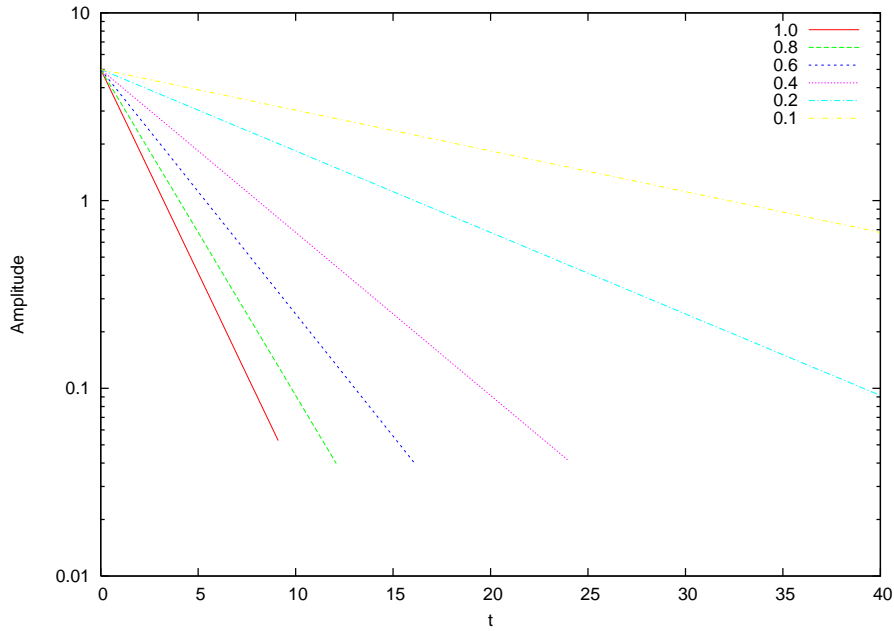
της σταθεράς  $\gamma$ . Το σημείο αυτό για το σύστημα είναι ένας “ελκυστής” (attractor).

Στη συνέχεια, προσθέτουμε την εξωτερική δύναμη και μελετούμε την απόκριση του συστήματος σε αυτή. Το πρώτο που παρατηρούμε στο σχήμα 4.23 είναι ότι το σύστημα μετά από μια μεταβατική κατάσταση (transient state) η οποία εξαρτάται από τις αρχικές συνθήκες, καταλήγει σε μία σταθερή κατάσταση η οποία δεν εξαρτάται από τις αρχικές συνθήκες. Στη συγκεκριμένη περίπτωση αυτό προβλέπεται εύκολα από τις Σχέσεις (4.26)–(4.28) όπου, αφού οι εκθετικοί όροι γίνουν αμελητέοι, επικρατεί ο όρος  $x_s(t)$  της (4.29). Ο τελευταίος μπορεί να γραφτεί στη μορφή

$$x(t) = x_0(\omega) \cos(\omega t + \delta(\omega))$$

$$x_0(\omega) = \frac{a_0}{\sqrt{(\omega_0^2 - \omega^2)^2 + \gamma^2 \omega^2}}, \quad \tan \delta(\omega) = \frac{\omega \gamma}{\omega^2 - \omega_0^2}. \quad (4.33)$$

Την παραπάνω σχέση την επιβεβαιώνουμε στο σχήμα 4.24 όπου μελετάμε την εξάρτηση του πλάτους  $x_0(\omega)$  από τη γωνιακή συχνότητα της εξωτερικής δύναμης. Τέλος, μελετάμε την τροχιά του συστήματος στο χώρο των φάσεων. Παρατηρούμε την ύπαρξη ελκυστή για το σύστημα,



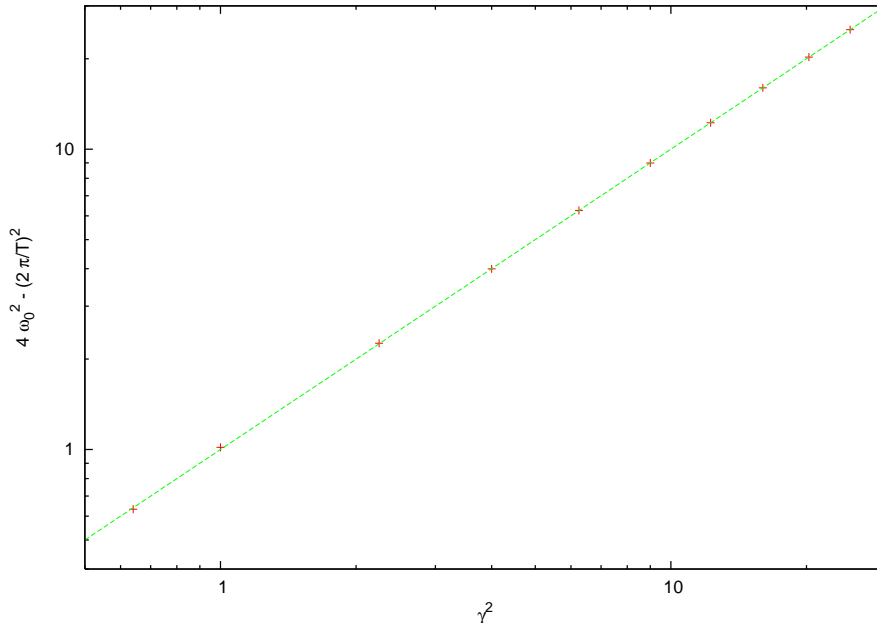
Σχήμα 4.21: Το πλάτος ταλάντωσης για τον αρμονικό ταλαντωτή με απόσβεση για διαφορετικές τιμές του συντελεστή απόσβεσης  $\gamma$  με  $\omega_0 = 3.145$ . Παρατηρούμε την εκθετική μείωση του πλάτους με το χρόνο.

δηλ. την ύπαρξη υπόχωρου πάνω στον οποίο κινείται το σύστημα όταν  $t \rightarrow +\infty$  ο οποίος είναι ανεξάρτητος των αρχικών συνθηκών. Στην περίπτωση αυτή, ο ελκυστής είναι καμπύλη μιας διάστασης σε αντίθεση με την περίπτωση που δεν έχουμε εξωτερική δύναμη και ο ελκυστής είναι μηδενοδιάστατο σημείο (βλ. σχήμα 4.20).

## 4.6 Εκκρεμές με Απόσβεση και Διέγερση

Στην παράγραφο αυτή θα μελετήσουμε ένα μη γραμμικό δυναμικό σύστημα με μη-τετριμμένες, χαοτικές, ιδιότητες. Πολλά ενδιαφέροντα δυναμικά συστήματα στη φύση παρουσιάζουν χαοτική συμπεριφορά. Αυτά είναι ντετερμινιστικά συστήματα των οποίων, δεδομένων των αρχικών συνθηκών, η κατάσταση τους μπορεί να υπολογιστεί για οποιαδήποτε χρονική στιγμή. Αλλά, ελάχιστα διαφορετικές αρχικές συνθήκες, δίνουν πολύ διαφορετικές τροχιές, με αποτέλεσμα να είναι πρακτικά αδύνατο να κάνουμε προβλέψεις σε βάθος χρόνου. Το σύστημα αυτό είναι το εκκρεμές στο ομογενές πεδίο βαρύτητας της γης με δύναμη απόσβεσης ανάλογη της ταχύτητας του εκκρεμούς και μια εξωτερική οδηγούσα δύ-





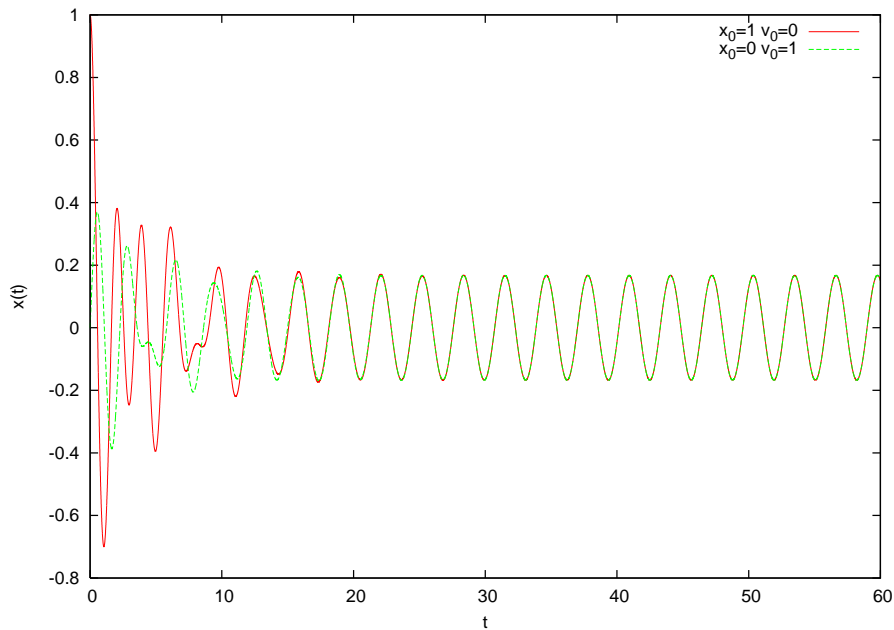
Σχήμα 4.22: Η περίοδος ταλάντωσης για τον αρμονικό ταλαντωτή με απόσβεση για διαφορετικές τιμές του συντελεστή απόσβεσης  $\gamma$  με  $\omega_0 = 3.145$ . Στους άξονες επιλέγουμε τις κατάλληλες ποσότητες για τη γραφική επιβεβαίωση της εξίσωσης (4.28), δηλ.  $(2\pi/T)^2 = 4\omega_0^2 - \gamma^2$ . Τα σημεία είναι οι μετρήσεις ενώ η ευθεία γραμμή είναι η θεωρητική πρόβλεψη, δηλ. η διαγώνιος  $y = x$

ναμη, κατακόρυφη στη διεύθυνση και μέτρο/φορά που μεταβάλλεται συνημιτονοειδώς με το χρόνο:

$$\frac{d^2\theta}{dt^2} + \gamma \frac{d\theta}{dt} + \omega_0^2 \sin \theta = -2A \cos \omega t \sin \theta. \quad (4.34)$$

Στην παραπάνω εξίσωση  $\theta$  είναι η γωνία με την κατακόρυφο,  $\gamma$  ο συντελεστής απόσβεσης,  $\omega_0^2 = g/L$  η φυσική κυκλική συχνότητα του εκκρεμούς και  $\omega$ ,  $2A$  η κυκλική συχνότητα και το πλάτος της εξωτερικής γωνιακής επιτάχυνσης που προκαλείται από την εξωτερική δύναμη.

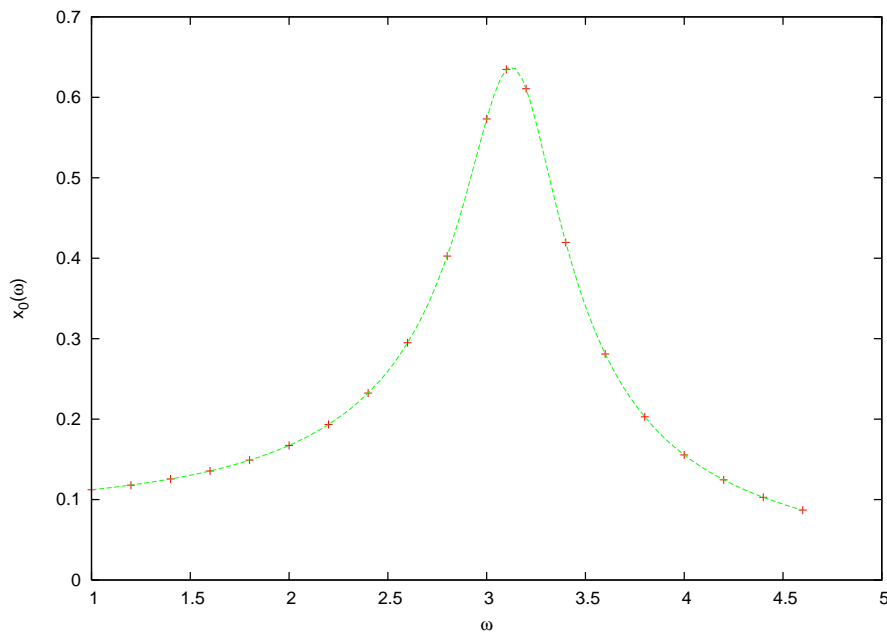
Όταν δεν υπάρχει εξωτερική δύναμη το σύστημα έχει, εξαιτίας των αποσβέσεων, ελκυστή το σημείο  $(\theta, \dot{\theta}) = (0, 0)$ . Αυτό θα συνεχίσει να συμβαίνει, καθώς αυξάνουμε το  $A$  από το μηδέν και ο ελκυστής παραμένει σταθερός για αρκετά μικρό  $A$ . Για κάποια τιμή  $A_c$  ο ελκυστής γίνεται ασταθής και η συμπεριφορά του συστήματος γίνεται πολύπλοκη. Αυτή θα μελετηθεί λεπτομερέστερα σε επόμενο κεφάλαιο, εδώ θα κάνουμε μια εισαγωγική προσέγγιση έχοντας κατά νου ότι το σύστημα αυτό παρουσιάζει ενδιαφέρον.



Σχήμα 4.23: Η περίοδος ταλάντωσης για τον αρμονικό ταλαντωτή με απόσβεση και εξωτερική δύναμη για διαφορετικές αρχικές συνθήκες. Έχουμε πάρει  $\omega_0 = 3.145$ ,  $\omega = 2.0$ ,  $\gamma = 0.5$  και  $a_0 = 1.0$ . Παρατηρούμε ότι το σύστημα παρουσιάζει μια μεταβατική κατάσταση μετά το πέρας της οποίας ταλαντώνεται σύμφωνα με τη σχέση  $x(t) = x_0(\omega) \cos(\omega t + \delta)$ .

Ο προγραμματισμός του συστήματος γίνεται με τετριμμένες αλλαγές του προγράμματος `dlo.f90`. Οι μετατροπές στο πρόγραμμα φαίνονται παρακάτω, έχοντας κατά νου πως  $X1 \leftrightarrow \theta$ ,  $X2 \leftrightarrow \dot{\theta}$ ,  $a_0 \leftrightarrow A$ . Το πρόγραμμα το αποθηκεύουμε στο αρχείο `fdp.f90` (`fdp`= Forced Damped Pendulum). Οι εντολές ανάμεσα στις τελίτσες είναι πανομοιότυπες με τα προγράμματα `dlo.f90`, `rk.f90`.

```
!=====
!Program to solve Forced Damped Pendulum
!using 4th order Runge-Kutta Method
!Output is written in file fdp.dat
!=====
program dlo_solve
  implicit none
  integer, parameter :: P=1010000
  .....
  Energy = 0.5D0*X2(i)*X2(i)+omega_02*(1.0D0-cos(X1(i)))
  .....
```

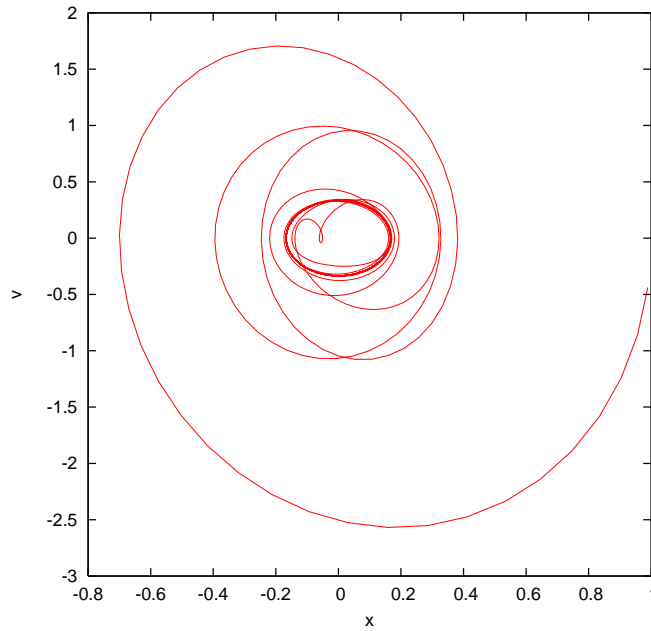


Σχήμα 4.24: Το πλάτος ταλάντωσης  $x_0(\omega)$  για τον αρμονικό ταλαντωτή με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 3.145$ ,  $\gamma = 0.5$  και  $a_0 = 1.0$ . Παρατηρούμε συντονισμό για  $\omega \approx \omega_0$ . Τα σημεία είναι οι μετρήσεις μας και η γραμμή που τα συνδέει είναι η θεωρητική πρόβλεψη (4.33).

```

end program dlo_solve
!
real(8) function f2(t,x1,x2)
  implicit none
  real(8) omega_0, omega, gamma, a_0, omega_02, omega2
  common /params/ omega_0, omega, gamma, a_0, omega_02, omega2
  real(8) t, x1, x2
  f2 = -(omega_02 + 2.0D0*a_0*cos(omega*t))*sin(x1) - gamma*x2
end function f2
!=====
subroutine RKSTEP(t,x1,x2,dt)
  implicit none
  .....
  real(8), parameter :: pi = 3.14159265358979324D0
  real(8), parameter :: pi2 = 6.28318530717958648D0
  .....
  x1 = x1 + h6*(k11 + 2.0D0*(k12+k13)+k14)
  x2 = x2 + h6*(k21 + 2.0D0*(k22+k23)+k24)
  if( x1 .gt. pi) x1 = x1 - pi2
  if( x1 .lt. -pi) x1 = x1 + pi2
end subroutine RKSTEP

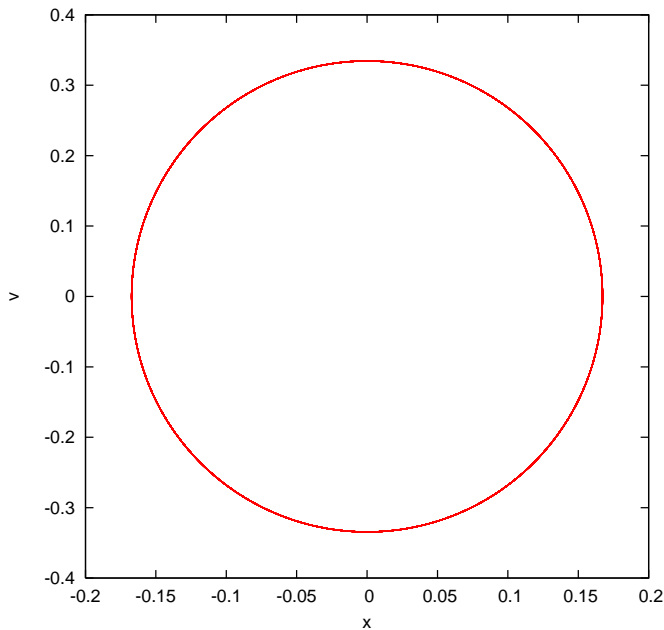
```



Σχήμα 4.25: Η τροχιά στο χώρο των φάσεων για τον αρμονικό ταλαντωτή με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 3.145$ ,  $\omega = 2.0$ ,  $\gamma = 0.5$  και  $a_0 = 1.0$ .

Τις τελευταίες γραμμές στο πρόγραμμα τις προσθέσαμε, ώστε να κρατήσουμε τη γωνία στο διάστημα  $[-\pi, \pi]$ .

Για να μελετήσουμε τις ιδιότητες του συστήματος θα θέσουμε  $\omega_0 = 1$ ,  $\omega = 2$ , και  $\gamma = 0.2$ , εκτός αν αναφέρουμε ρητώς διαφορετικά. Η φυσική περίοδος του εκκρεμούς είναι  $T_0 = 2\pi/\omega_0 = 2\pi \approx 6.28318530717958648$ , ενώ αυτή της εξωτερικής δύναμης  $T = 2\pi/\omega = \pi \approx 3.14159265358979324$ . Το σύστημα για  $A < A_c$  με  $A_c \approx 0.18$  έχει σταθερό ελκυστή το σημείο  $(\theta, \dot{\theta}) = (0, 0)$ , ενώ για  $A_c < A < 0.71$  ο ελκυστής είναι κλειστή καμπύλη. Η περίοδος της ταλάντωσης βρίσκεται να είναι διπλάσια αυτής της εξωτερικής δύναμης. Για  $0.72 < A < 0.79$  ο ελκυστής είναι ανοιχτή καμπύλη, επειδή το εκκρεμές εκτελεί ολόκληρους κύκλους στη σταθερή του κατάσταση. Η περίοδος στο διάστημα αυτό γίνεται ίση με αυτή της εξωτερικής δύναμης. Για  $0.79 < A \lesssim 1.033$  η περίοδος διπλασιάζεται διαδοχικά για κρίσιμες τιμές του  $A$ , η τροχιά όμως συνεχίζει να είναι περιοδική. Για μεγαλύτερες τιμές του  $A$  η τροχιά παύει να είναι περιοδική και το σύστημα έχει χαοτική συμπεριφορά. Για  $A \approx 3.1$  βρίσκουμε το σύστημα να έχει πάλι περιοδική κίνηση, ενώ για  $A \approx 3.8 - 4.448$  να έχουμε το φαινόμενο διπλασιασμού της περιόδου. Για  $A \approx 4.4489$  έχουμε καθαρή χαοτική συμπεριφορά κ.ο.κ. Τα αποτελέσματα αυτά

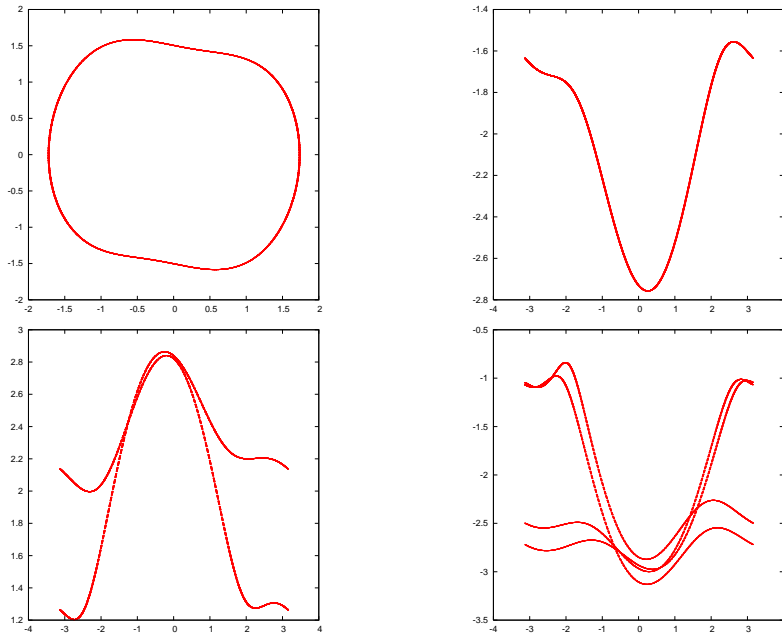


Σχήμα 4.26: Η τροχιά στο χώρο των φάσεων για τον αρμονικό ταλαντωτή με απόσβεση και εξωτερική δύναμη για  $t > 100$ . Έχουμε πάρει  $\omega_0 = 3.145$ ,  $\omega = 2.0$ ,  $\gamma = 0.5$  και  $a_0 = 1.0$ . Παρατηρούμε τον ελκυστή που τώρα είναι καμπύλη (έλλειψη) μιας διάστασης.

περιγράφονται στα σχήματα 4.27–4.29.

Για την ανάλυση της συμπεριφοράς του εκκρεμούς και κυρίως για τη διευκόλυνση της διάκρισης μεταξύ περιοδικής και χαοτικής συμπεριφοράς μπορεί κανείς να μελετήσει τα λεγόμενα διαγράμματα Poincaré. Στα διαγράμματα αυτά τοποθετούμε ένα σημείο στο χώρο των φάσεων κάθε φορά που ο χρόνος είναι ακέραιο πολλαπλάσιο της περιόδου της εξωτερικής δύναμης. Με τον τρόπο αυτό, αν η κίνηση είναι περιοδική με περίοδο ίση με την περίοδο της εξωτερικής δύναμης, θα έχουμε ένα σημείο στο διάγραμμα και γενικότερα θα έχουμε  $n$  σημεία, αν η περίοδος είναι  $n$ -πολλαπλάσιο της  $T = 2\pi/\omega$ . Οπότε κανείς περιμένει, όταν παρατηρείται το φαινόμενο διπλασιασμού της περιόδου, το διάγραμμα Poincaré να αποκτά επί πλέον μεμονωμένα σημεία, ενώ όταν η συμπεριφορά είναι χαοτική, τα σημεία να ανήκουν σε έναν υπόχωρο του χώρου των φάσεων που έχει πολυπλοκότερη δομή. Αυτό μπορούμε εύκολα να το προγραμματίσουμε στον κώδικά μας στο `fdp.f90` ή εναλλακτικά να πάρουμε τη σχετική πληροφορία από το αρχείο εξόδου `fdp.dat` με το πρόγραμμα `awk` το οποίο τρέχουμε από τη γραμμή εντολών<sup>12</sup>:

<sup>12</sup>Η εντολή μπορεί να γραφτεί σε μία γραμμή χωρίς το τελικό `\` της πρώτης γραμμής.



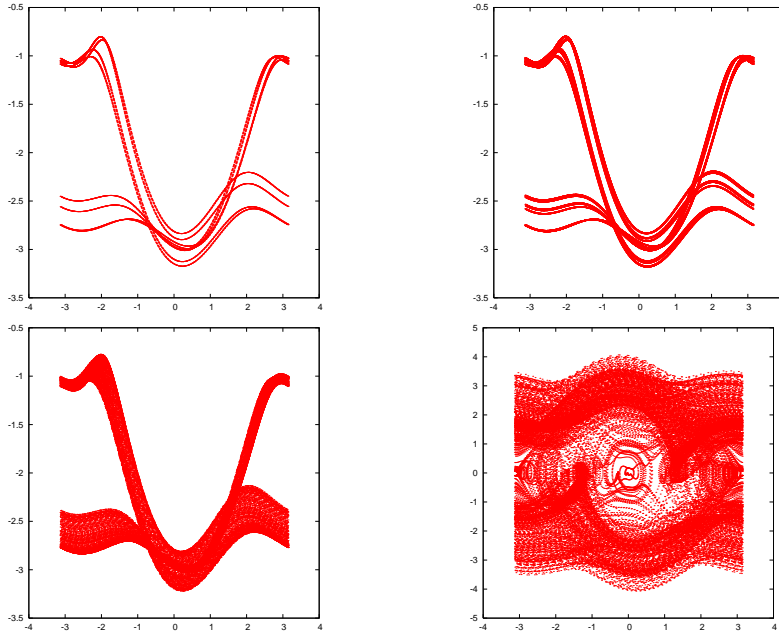
Σχήμα 4.27: Τροχιά στο χώρο των φάσεων για το εκκρεμές με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 1.0$ ,  $\omega = 2.0$ ,  $\gamma = 0.2$  και  $A = 0.60, 0.72, 0.85, 1.02$ . Παρατηρούμε το φαινόμενο του διπλασιασμού της περιόδου.

```
awk -v o=$omega -v nt=$Nt -v tf=$TF \
  'BEGIN{T=6.283185307179/o;dt=tf/nt;} $1%T<dt{ print $2,$3}'\
  fdp.dat
```

όπου  $\$omega$ ,  $\$Nt$ ,  $\$TF$  οι τιμές της κυκλικής συχνότητας  $\omega$ , αριθμού χρονικών σημείων και τελικού χρόνου  $t_f$ . Στο πρόγραμμα υπολογίζουμε την περίοδο  $T$  και το βήμα χρόνου  $dt$ . Στη συνέχεια, τυπώνουμε εκείνες τις γραμμές του αρχείου των οποίων ο χρόνος είναι ακέραιο πολλαπλάσιο της περιόδου με ακρίβεια χρόνου  $dt$ <sup>13</sup>. Αυτό γίνεται με την πράξη `modulo $1 % T < dt` που είναι TRUE, όταν το υπόλοιπο της διαίρεσης της πρώτης στήλης ( $\$1$ ) του αρχείου `fdp.dat` έχει υπόλοιπο διαίρεσης με την περίοδο  $T$  μικρότερο από  $dt$ . Τα αποτελέσματα για τη χαοτική φάση φαίνονται στο σχήμα 4.30.

Κλείνουμε τη μελέτη μας με την παρουσίαση μιας ακόμα έννοιας που μας βοηθάει στην ανάλυση των ιδιοτήτων του εκκρεμούς. Αυτή είναι η έννοια της “λεκάνης του ελκυστή” (basin of attraction) η οποία

<sup>13</sup>Φυσικά αυτό κάνει τα μεμονωμένα σημεία να φαίνονται λίγο συγκεχυμένα στο διάγραμμα Poincaré.



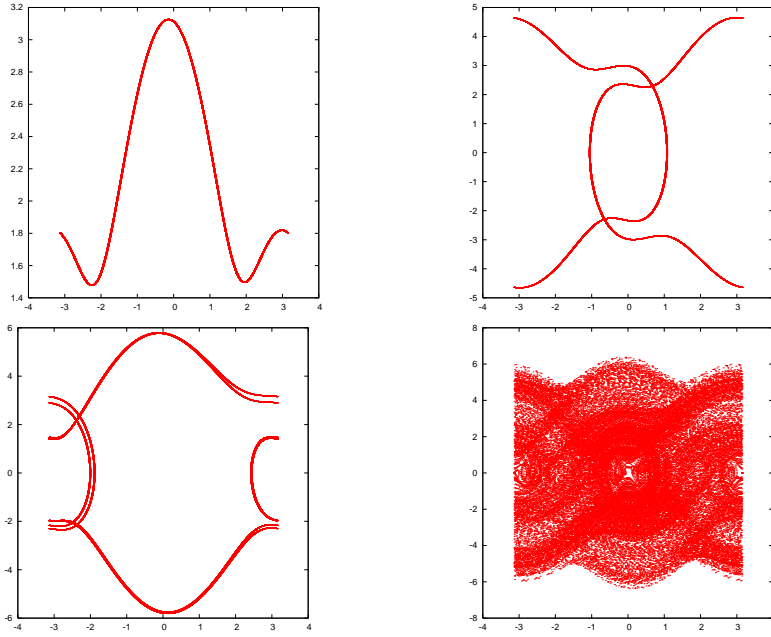
Σχήμα 4.28: Τροχιά στο χώρο των φάσεων για το εκκρεμές με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 1.0$ ,  $\omega = 2.0$ ,  $\gamma = 0.2$  και  $A = 1.031, 1.033, 1.04, 1.4$ . Παρατηρούμε την εμφάνιση χαοτικής συμπεριφοράς.

είναι το σύνολο των αρχικών συνθηκών στο χώρο των φάσεων που οδηγούν το σύστημα στο συγκεκριμένο ελκυστή. Στην περίπτωσή μας, το εκκρεμές για  $A > 0.79$  εκτελεί κυκλική κίνηση είτε με θετική είτε με αρνητική φορά (μετά την παρέλευση της παροδικής φάσης φυσικά) οι οποίες αποτελούν τους δύο ελκυστές του συστήματος. Παίρνοντας ένα μεγάλο δείγμα από αρχικές συνθήκες και σημειώνοντας το πρόσημο της γωνιακής ταχύτητας μετά την παρέλευση της παροδικής φάσης, παίρνουμε το σχήμα 4.31. Στην περιοδική φάση διακρίνουμε περιοχές, των οποίων τα περιγράμματα δεν είναι καθαρά, οι οποίες δεν οδηγούν στο συγκεκριμένο ελκυστή.

## 4.7 Παράρτημα: Στη Μέθοδο Euler-Verlet

Οι Σχέσεις (4.11) προκύπτουν από το ανάπτυγμα κατά Taylor

$$\begin{aligned}\theta(t + \Delta t) &= \theta(t) + (\Delta t)\theta'(t) + \frac{(\Delta t)^2}{2!}\theta''(t) + \frac{(\Delta t)^3}{3!}\theta'''(t) + \mathcal{O}((\Delta t)^4) \\ \theta(t - \Delta t) &= \theta(t) - (\Delta t)\theta'(t) + \frac{(\Delta t)^2}{2!}\theta''(t) - \frac{(\Delta t)^3}{3!}\theta'''(t) + \mathcal{O}((\Delta t)^4).\end{aligned}$$



Σχήμα 4.29: Τροχιά στο χώρο των φάσεων για το εκκρεμές με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 1.0$ ,  $\omega = 2.0$ ,  $\gamma = 0.2$  και  $A = 1.568, 3.8, 4.44, 4.5$ . Παρατηρούμε την παύση και επανεμφάνιση χαοτικής συμπεριφοράς.

Προσθέτοντας και αφαιρώντας κατά μέλη παίρνουμε

$$\begin{aligned}\theta(t + \Delta t) + \theta(t - \Delta t) &= 2\theta(t) + (\Delta t)^2 \theta''(t) + \mathcal{O}((\Delta t)^4) \\ \theta(t + \Delta t) - \theta(t - \Delta t) &= 2(\Delta t) \theta'(t) + \mathcal{O}((\Delta t)^3)\end{aligned}\quad (4.35)$$

που δίνουν

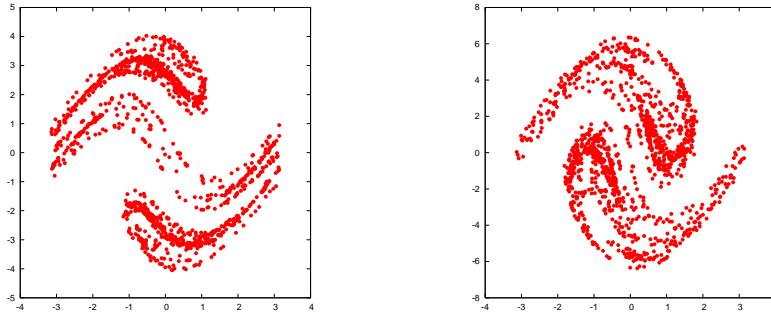
$$\begin{aligned}\theta(t + \Delta t) &= 2\theta(t) - \theta(t - \Delta t) + (\Delta t)^2 \alpha(t) + \mathcal{O}((\Delta t)^4) \\ \omega(t) &= \frac{\theta(t + \Delta t) - \theta(t - \Delta t)}{2(\Delta t)} + \mathcal{O}((\Delta t)^2)\end{aligned}\quad (4.36)$$

που είναι οι σχέσεις (4.11). Από την πρώτη σχέση παίρνουμε και τις εξισώσεις της μεθόδου Euler (4.9):

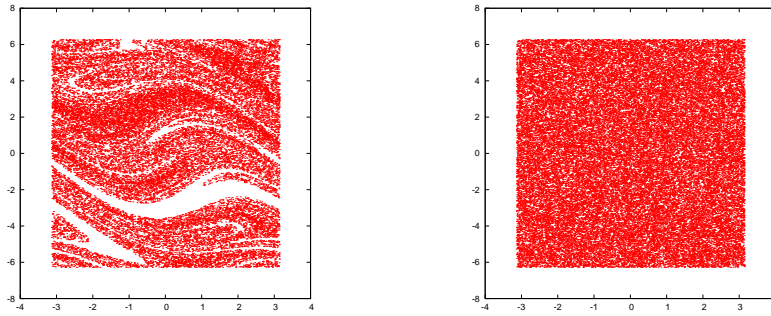
$$\theta(t + \Delta t) = \theta(t) + \omega(t)(\Delta t) + \mathcal{O}((\Delta t)^2) \quad (4.37)$$

Στις προσομοιώσεις το σημαντικό είναι το συνολικό σφάλμα που συσσωρεύεται μετά από τα  $N - 1$  βήματα της ολοκλήρωσης. Ειδικά για τη συγκεκριμένη μέθοδο πρέπει να δούμε τα σφάλματα που συσσωρεύονται ιδιαίτερα προσεκτικά:





Σχήμα 4.30: Διάγραμμα Poincaré για το εκκρεμές με απόσβεση και εξωτερική δύναμη όταν παρουσιάζει χαοτική συμπεριφορά. Έχουμε πάρει  $\omega_0 = 1.0$ ,  $\omega = 2.0$ ,  $\gamma = 0.2$  και  $A = 1.4, 4.5$ .



Σχήμα 4.31: Basin of attraction για το εκκρεμές με απόσβεση και εξωτερική δύναμη. Έχουμε πάρει  $\omega_0 = 1.0$ ,  $\omega = 2.0$ ,  $\gamma = 0.2$  και  $A = 0.85, 1.4$ . Διακρίνουμε την περιοδική από τη χαοτική συμπεριφορά.

- Το σφάλμα στην ταχύτητα  $\omega(t)$  δε συσσωρεύεται, γιατί υπολογίζεται από τη διαφορά των θέσεων  $\theta(t + \Delta t) - \theta(t - \Delta t)$ .
- Στη θέση, το σφάλμα συσσωρεύεται ως εξής: Έστω  $\delta\theta(t)$  το συνολικό σφάλμα που έχει συσσωρευτεί από την ολοκλήρωση από χρόνο  $t_0$  έως  $t$ . Τότε σύμφωνα με τα αναπτύγματα (4.36), το σφάλμα στο πρώτο βήμα είναι  $\delta\theta(t_0 + \Delta t) = \mathcal{O}((\Delta t)^4)$ . Τότε<sup>14</sup>

$$\begin{aligned}
 \theta(t_0 + 2\Delta t) &= 2\theta(t_0 + \Delta t) - \theta(t_0) + \Delta t^2 \alpha(t_0 + \Delta t) + \mathcal{O}((\Delta t)^4) \Rightarrow \\
 \delta\theta(t_0 + 2\Delta t) &= 2\delta\theta(t_0 + \Delta t) - \delta\theta(t_0) + \mathcal{O}((\Delta t)^4) \\
 &= 2\mathcal{O}((\Delta t)^4) - 0 + \mathcal{O}((\Delta t)^4) \\
 &= 3\mathcal{O}((\Delta t)^4)
 \end{aligned}$$

<sup>14</sup>Θυμίζουμε ότι η επιτάχυνση  $\alpha(t)$  δίνεται, οπότε  $\delta\alpha(t) = 0$ .

Στα επόμενα βήματα παίρνουμε

$$\begin{aligned}\theta(t_0 + 3\Delta t) &= 2\theta(t_0 + 2\Delta t) - \theta(t_0 + \Delta t) + \Delta t^2 \alpha(t_0 + 2\Delta t) + \mathcal{O}((\Delta t)^4) \Rightarrow \\ \delta\theta(t_0 + 3\Delta t) &= 2\delta\theta(t_0 + 2\Delta t) - \delta\theta(t_0 + \Delta t) + \mathcal{O}((\Delta t)^4) \\ &= 6\mathcal{O}((\Delta t)^4) - \mathcal{O}((\Delta t)^4) + \mathcal{O}((\Delta t)^4) \\ &= 6\mathcal{O}((\Delta t)^4)\end{aligned}$$

$$\begin{aligned}\theta(t_0 + 4\Delta t) &= 2\theta(t_0 + 3\Delta t) - \theta(t_0 + 2\Delta t) + \Delta t^2 \alpha(t_0 + 3\Delta t) + \mathcal{O}((\Delta t)^4) \Rightarrow \\ \delta\theta(t_0 + 4\Delta t) &= 2\delta\theta(t_0 + 3\Delta t) - \delta\theta(t_0 + 2\Delta t) + \mathcal{O}((\Delta t)^4) \\ &= 12\mathcal{O}((\Delta t)^4) - 3\mathcal{O}((\Delta t)^4) + \mathcal{O}((\Delta t)^4) \\ &= 10\mathcal{O}((\Delta t)^4)\end{aligned}$$

Και επαγωγικά, αν  $\delta\theta(t_0 + (n-1)\Delta t) = \frac{(n-1)n}{2}\mathcal{O}((\Delta t)^4)$  στο επόμενο βήμα παίρνουμε

$$\begin{aligned}\theta(t_0 + n\Delta t) &= 2\theta(t_0 + (n-1)\Delta t) - \theta(t_0 + (n-2)\Delta t) + \Delta t^2 \alpha(t_0 + (n-1)\Delta t) \\ &\quad + \mathcal{O}((\Delta t)^4) \Rightarrow \\ \delta\theta(t_0 + n\Delta t) &= 2\delta\theta(t_0 + (n-1)\Delta t) - \delta\theta(t_0 + (n-2)\Delta t) + \mathcal{O}((\Delta t)^4) \\ &= 2\frac{(n-1)n}{2}\mathcal{O}((\Delta t)^4) - \frac{(n-2)(n-1)}{2}\mathcal{O}((\Delta t)^4) + \mathcal{O}((\Delta t)^4) \\ &= \frac{n(n+1)}{2}\mathcal{O}((\Delta t)^4)\end{aligned}$$

Άρα, τελικά,

$$\delta\theta(t_0 + n\Delta t) = \frac{n(n+1)}{2}\mathcal{O}((\Delta t)^4) \sim \frac{1}{\Delta t^2}\mathcal{O}((\Delta t)^4) \sim \mathcal{O}((\Delta t)^2) \quad (4.38)$$

Άρα το ολικό σφάλμα είναι  $\mathcal{O}((\Delta t)^2)$ .

Για πληρότητα αναφέρουμε και τον αλγόριθμο Velocity Verlet ή μέθοδο Leapfrog. Στην περίπτωση αυτή χρησιμοποιούμε ρητά την ταχύτητα:

$$\begin{aligned}\theta_{n+1} &= \theta_n + \omega_n \Delta t + \frac{1}{2}\alpha_n \Delta t^2 \\ \omega_{n+\frac{1}{2}} &= \omega_n + \frac{1}{2}\alpha_n \Delta t \\ \omega_{n+1} &= \omega_{n+\frac{1}{2}} + \frac{1}{2}\alpha_{n+1} \Delta t.\end{aligned} \quad (4.39)$$

Στο τελευταίο βήμα χρειαζόμαστε την επιτάχυνση  $\alpha_{n+1}$ , οπότε πρέπει αυτή να εξαρτάται μόνο από τη θέση  $\theta_{n+1}$  και όχι από την ταχύτητα.

Οι μέθοδοι Verlet είναι δημοφιλείς σε προσομοιώσεις molecular dynamics, κυρίως συστημάτων με πολλά σωματίδια. Έχουν το ιδιαίτερο προσόν ότι υλοποιούνται σχετικά εύκολα οι περιορισμοί (constraints) στους οποίους υπόκειται τα σωματίδια που αποτελούν το σύστημα.

## 4.8 Παράρτημα: Runge–Kutta 2ης τάξης

Στην παράγραφο αυτή θα δείξουμε με δύο τρόπους γιατί η επιλογή του ενδιάμεσου σημείου 2 στην εξίσωση (4.17) μειώνει το σφάλμα κατά μία δύναμη του βήματος  $h$ . Όπως θα φανεί, η επιλογή του μέσου του διαστήματος για το σημείο 2 δεν είναι τυχαία (οπότε λ.χ. το σημείο με  $t = t_n + 0.4h$  δεν θα είχε το ίδιο αποτέλεσμα). Πράγματι από τη σχέση

$$\frac{dx}{dt} = f(t, x) \Rightarrow x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} f(t, x) dt. \quad (4.40)$$

Αναπτύσσοντας κατά Taylor γύρω από το σημείο  $(t_{n+1/2}, x_{n+1/2})$ , παίρνουμε

$$f(t, x) = f(t_{n+1/2}, x_{n+1/2}) + (t - t_{n+1/2}) \frac{df}{dt}(t_{n+1/2}) + \mathcal{O}(h^2). \quad (4.41)$$

Οπότε

$$\begin{aligned} & \int_{t_n}^{t_{n+1}} f(t, x) dt \\ &= f(t_{n+1/2}, x_{n+1/2})(t_{n+1} - t_n) + \frac{df}{dt}(t_{n+1/2}) \frac{(t - t_{n+1/2})^2}{2} \Big|_{t_n}^{t_{n+1}} \\ & \quad + \mathcal{O}(h^2)(t_{n+1} - t_n) \\ &= f(t_{n+1/2}, x_{n+1/2})h + \frac{df}{dt}(t_{n+1/2}) \left\{ \frac{(t_{n+1} - t_{n+1/2})^2}{2} - \frac{(t_n - t_{n+1/2})^2}{2} \right\} \\ & \quad + \mathcal{O}(h^2)h \\ &= f(t_{n+1/2}, x_{n+1/2})h + \frac{df}{dt}(t_{n+1/2}) \left\{ \frac{h^2}{2} - \frac{(-h)^2}{2} \right\} + \mathcal{O}(h^3) \\ &= f(t_{n+1/2}, x_{n+1/2})h + \mathcal{O}(h^3). \end{aligned} \quad (4.42)$$

Παρατηρούμε ότι για την εξαφάνιση του όρου  $\mathcal{O}(h)$  είναι αναγκαία η τοποθέτηση του βοηθητικού σημείου στο χρόνο  $t_{n+1/2}$ .

Η επιλογή αυτή δεν είναι μοναδική. Αυτό μπορεί να φανεί και από μια διαφορετική ανάλυση του αναπτύγματος κατά Taylor. Αναπτύσσοντας τώρα γύρω από το σημείο  $(t_n, x_n)$ , παίρνουμε

$$\begin{aligned}
 x_{n+1} &= x_n + (t_{n+1} - t_n) \frac{dx_n}{dt} + \frac{1}{2} (t_{n+1} - t_n)^2 \frac{d^2 x_n}{dt^2} + \mathcal{O}(h^3) \\
 &= x_n + hf_n + \frac{h^2}{2} \frac{df_n}{dt} + \mathcal{O}(h^3) \\
 &= x_n + hf_n + \frac{h^2}{2} \left( \frac{\partial f_n}{\partial t} + \frac{\partial f_n}{\partial x} \frac{dx_n}{dt} \right) + \mathcal{O}(h^3) \\
 &= x_n + hf_n + \frac{h^2}{2} \left( \frac{\partial f_n}{\partial t} + \frac{\partial f_n}{\partial x} f_n \right) + \mathcal{O}(h^3), \tag{4.43}
 \end{aligned}$$

όπου για συντομία θέσαμε  $f_n \equiv f(t_n, x_n)$ ,  $\frac{dx_n}{dt} \equiv \frac{dx}{dt}(x_n)$  κ.ο.κ. Ορίζουμε

$$\begin{aligned}
 k_1 &= f(t_n, x_n) = f_n \\
 k_2 &= f(t_n + ah, x_n + bhk_1) \\
 x_{n+1} &= x_n + h(c_1 k_1 + c_2 k_2). \tag{4.44}
 \end{aligned}$$

και θα προσδιορίσουμε τις συνθήκες, έτσι ώστε στο σφάλμα οι όροι  $\mathcal{O}(h^2)$  της τελευταίας εξίσωσης να ταυτίζονται με αυτούς της (4.43). Αναπτύσσοντας την  $k_2$ , παίρνουμε

$$\begin{aligned}
 k_2 &= f(t_n + ah, x_n + bhk_1) \\
 &= f(t_n, x_n + bhk_1) + ha \frac{\partial f}{\partial t}(t_n, x_n + bhk_1) + \mathcal{O}(h^2) \\
 &= f(t_n, x_n) + hb k_1 \frac{\partial f}{\partial x}(t_n, x_n) + ha \frac{\partial f}{\partial t}(t_n, x_n) + \mathcal{O}(h^2) \\
 &= f_n + h \left\{ a \frac{\partial f_n}{\partial t} + b k_1 \frac{\partial f_n}{\partial x} \right\} + \mathcal{O}(h^2) \\
 &= f_n + h \left\{ a \frac{\partial f_n}{\partial t} + b f_n \frac{\partial f_n}{\partial x} \right\} + \mathcal{O}(h^2) \tag{4.45}
 \end{aligned}$$

Αντικαθιστώντας στην (4.44), παίρνουμε

$$\begin{aligned}
 x_{n+1} &= x_n + h(c_1 k_1 + c_2 k_2) \\
 &= x_n + h \left\{ c_1 f_n + c_2 f_n + c_2 h \left( a \frac{\partial f_n}{\partial t} + b f_n \frac{\partial f_n}{\partial x} \right) + \mathcal{O}(h^2) \right\} \\
 &= x_n + h(c_1 + c_2) f_n + \frac{h^2}{2} \left( (2c_2 a) \frac{\partial f_n}{\partial t} + (2c_2 b) f_n \frac{\partial f_n}{\partial x} \right) \\
 &\quad + \mathcal{O}(h^3). \tag{4.46}
 \end{aligned}$$

Αρκεί λοιπόν να επιλέξουμε

$$\begin{aligned}c_1 + c_2 &= 1 \\2c_2a &= 1 \\2c_2b &= 1.\end{aligned}\tag{4.47}$$

Η επιλογή  $c_1 = 0$ ,  $c_2 = 1$ ,  $a = b = 1/2$  μας δίνει τη μέθοδο (4.19). Άλλες επιλογές στη βιβλιογραφία είναι  $c_2 = 1/2$  και  $c_2 = 3/4$ .

## 4.9 Ασκήσεις

- 4.1 Αποδείξτε ότι το συνολικό σφάλμα στη μέθοδο Euler-Cromer είναι τάξης  $\Delta t$
- 4.2 Αναπαράγετε τα αποτελέσματα των σχημάτων 4.11–4.18
- 4.3 Βελτιώστε τον υπολογισμό του χρόνου στα προγράμματα των μεθόδων Euler, Euler-Cromer, Euler-Verlet, Runge-Kutta, ώστε να μην έχουμε προσθετική συσσώρευση σφαλμάτων στο χρόνο, όταν η παράμετρος  $h$  είναι πολύ μικρή (λ.χ. στις εντολές  $T(i)=T(i-1)+h$ ). Επαναλάβετε την ανάλυση της προηγούμενης άσκησης.
- 4.4 Μεταβάλετε τα προγράμματα των μεθόδων Euler, Euler-Cromer, Euler-Verlet, Runge-Kutta ώστε  $REAL \rightarrow REAL(8)$  (προσοχή στις σταθερές: να τις ορίσετε ρητά να είναι διπλής ακρίβειας προσθέτοντας τον εκθέτη D0). Επαναλάβετε την ανάλυση της προηγούμενης άσκησης.
- 4.5 Να επαναλάβετε τη σύγκριση των μεθόδων Euler, Euler-Cromer, Euler-Verlet, Runge-Kutta για τα παρακάτω συστήματα των οποίων η αναλυτική λύση είναι γνωστή:
- (α') Σωματίδιο που πέφτει ελεύθερα στο ομογενές πεδίο βαρύτητας. Θεωρήστε  $v(0) = 0$ ,  $m = 1$ ,  $g = 10$ .
- (β') Σωματίδιο που πέφτει στο ομογενές πεδίο βαρύτητας μέσα σε ρευστό από το οποίο δέχεται δύναμη  $F = -kv$ . Θεωρήστε  $v(0) = 0$ ,  $m = 1$ ,  $g = 10$ ,  $k = 0.1, 1.0, 2.0$ . Υπολογίστε την οριζική ταχύτητα του σωματιδίου αριθμητικά και συγκρίνετέ τη με τη θεωρητική της τιμή.
- (γ') Επαναλάβετε για δύναμη αντίστασης μέτρου  $|F| = kv^2$ .
- 4.6 Μελετήστε το σύστημα του αρμονικού ταλαντωτή με απόσβεση

$$\frac{d^2x}{dt^2} + \gamma \frac{dx}{dt} + \omega_0^2 x = 0. \quad (4.48)$$

Πάρτε  $\omega_0 = 3.145$ ,  $\gamma = 0.5$  και υπολογίστε την ενέργεια συναρτήσει του χρόνου. Είναι η τιμή της μονοτονική; Γιατί; (Δείξτε ότι  $d(E/m)/dt = -\gamma v^2$ ). Επαναλάβετε για  $\gamma = 4, 5, 6, 7, 8$ . Πότε το σύστημα ταλαντώνεται και πότε όχι; Υπολογίστε αριθμητικά την κρίσιμη τιμή του  $\gamma$  για την οποία το σύστημα μετατρέπεται από ταλαντούμενο σε μη ταλαντούμενο. Συγκρίνετε το αποτέλεσμα σας με την αναλυτική λύση.

4.7 Αναπαράγετε τα αποτελέσματα που οδηγούν στα σχήματα 4.19–4.22.

4.8 Αναπαράγετε τα αποτελέσματα που οδηγούν στα σχήματα 4.23–4.26. Υπολογίστε αριθμητικά τη φάση  $\delta(\omega)$  και επιβεβαιώστε τη Σχέση (4.33).

4.9 Θεωρήστε ότι το μοντέλο για μια κούνια είναι ο αρμονικός ταλαντωτής με απόσβεση ο οποίος υπόκειται σε περιοδική δύναμη η οποία είναι στιγμιαία ώθηση συχνότητας  $\omega$ . Ορίστε το “στιγμιαία” ότι είναι η ώθηση που δίνει επιτάχυνση  $a_0$  το κατάλληλο χρονικό διάστημα διάρκειας  $\Delta t$  και 0 στα υπόλοιπα βήματα. Υπολογίστε το πλάτος  $x_0(\omega)$  για  $\omega_0 = 3.145$  και  $\gamma = 0.5$ .

4.10 Θεωρήστε ότι η εξωτερική δύναμη είναι το “μισό συνημίτονο” δίνοντας επιτάχυνση στον ο αρμονικό ταλαντωτή με απόσβεση

$$a(t) = \begin{cases} a_0 \cos \omega t & \cos \omega t > 0 \\ 0 & \cos \omega t \leq 0 \end{cases}$$

Μελετήστε τη μετάβαση του συστήματος στη σταθερή κατάσταση και υπολογίστε το πλάτος  $x_0(\omega)$  για  $\omega_0 = 3.145$  και  $\gamma = 0.5$ .

4.11 Θεωρήστε ότι η εξωτερική δύναμη δίνει επιτάχυνση στον αρμονικό ταλαντωτή με απόσβεση

$$a(t) = \frac{1}{\pi} + \frac{1}{2} \cos \omega + \frac{2}{3\pi} \cos 2\omega t - \frac{2}{15\pi} \cos 4\omega t$$

Μελετήστε τη μετάβαση του συστήματος στη σταθερή κατάσταση και υπολογίστε το πλάτος  $x_0(\omega)$  για  $\omega_0 = 3.145$  και  $\gamma = 0.5$ . Συγκρίνετε τα αποτελέσματά σας με αυτά της προηγούμενης άσκησης. Τι συμπεραίνετε;

4.12 Γράψτε ένα πρόγραμμα που να προσομοιώνει ταυτόχρονα  $N$  όμοιους αρμονικούς ταλαντωτές. Πάρτε  $N = 20$  και δώστε τυχαίες αρχικές συνθήκες σε κάθε έναν από αυτούς και παρακολουθήστε τις τροχιές τους στο χώρο των φάσεων. Παρατηρήστε αν οι τροχιές τέμνονται και εξηγήστε τα αποτελέσματά σας.

4.13 Στην προηγούμενη άσκηση τοποθετήστε τους  $N = 20$  αρμονικούς ταλαντωτές στο χώρο των φάσεων σε ένα μικρό τετράγωνο με κέντρο την αρχή των αξόνων. Παρακολουθήστε την εξέλιξη του συστήματος στο χρόνο. Αλλάζει το σχήμα με το χρόνο; Αλλάζει το εμβαδόν; Εξηγήστε...

- 4.14 Επαναλάβετε την προηγούμενη άσκηση, όταν υπάρχει απόσβεση με  $\gamma = 0.5$ . Πάρτε  $\omega_0 = 3.145$ .
- 4.15 Στην περίπτωση του εκκρεμούς με απόσβεση και εξωτερική δύναμη, πάρτε  $\omega = 2$ ,  $\omega_0 = 1.0$ ,  $\gamma = 0.2$  μελετήστε την παροδική φάση στα διαγράμματα  $\theta(t)$ ,  $\dot{\theta}(t)$  για  $A = 0.1, 0.5, 0.79, 0.85, 1.03, 1.4$ .
- 4.16 Στην περίπτωση του εκκρεμούς με απόσβεση και εξωτερική δύναμη, πάρτε  $\omega = 2$ ,  $\omega_0 = 1.0$ ,  $\gamma = 0.2$  μελετήστε τις τροχιές στο χώρο της φάσης για  $A = 0.1, 0.19, 0.21, 0.25, 0.5, 0.71, 0.79, 0.85, 1.02, 1.031, 1.033, 1.05, 1.08, 1.1, 1.4, 1.8, 3.1, 3.5, 3.8, 4.2, 4.42, 4.44, 4.445, 4.447, 4.4488$ .
- 4.17 Αναπαράγετε τα σχήματα 4.30.
- 4.18 Αναπαράγετε τα σχήματα 4.31.
- 4.19 Στο πρόβλημα του εκκρεμούς με απόσβεση και εξωτερική περιοδική δύναμη να πάρετε:

$$\omega_0 = 1, \quad \omega = 2, \quad \gamma = 0.2$$

Η κίνηση του συστήματος για  $A = 0.60$ ,  $A = 0.75$  και  $A = 0.85$  είναι περιοδική μετά από τη μεταβατική συμπεριφορά (transient behavior). Να μετρήσετε την περίοδο της κίνησης με ακρίβεια 3 σημαντικών δεκαδικών ψηφίων σε κάθε περίπτωση και να τη συγκρίνετε με την φυσική περίοδο του εκκρεμούς και την περίοδο της εξωτερικής δύναμης. Ως αρχικές συνθήκες να πάρετε  $(\theta_0, \dot{\theta}_0) = (3.1, 0.0), (2.5, 0.0), (2.0, 0.0), (1.0, 0.0), (0.2, 0.0), (0.0, 1.0), (0.0, 3.0), (0.0, 6.0)$  και να επιβεβαιώσετε πως η περίοδος είναι ανεξάρτητη των αρχικών συνθηκών.

- 4.20 Στο πρόβλημα του εκκρεμούς με απόσβεση και εξωτερική περιοδική δύναμη να πάρετε:

$$\omega_0 = 1, \quad \omega = 2, \quad \gamma = 0.2$$

και να μελετήσετε την κίνηση του εκκρεμούς, όταν το πλάτος (ανάλογο της δύναμης)  $A$  μεταβάλλεται στο διάστημα  $[0.2, 5.0]$ . Να πάρετε διακριτές τιμές του  $A$  χωρίζοντας το παραπάνω διάστημα σε διαστήματα πλάτους  $\delta A = 0.002$ . Για κάθε τιμή του  $A$ , να καταχωρήσετε σε ένα αρχείο την τιμή του  $A$ , της γωνιακής



θέσης και γωνιακής ταχύτητας του εκκρεμούς, όταν  $t_k = k\pi$  με  $k = k_{trans}, k_{trans} + 1, k_{trans} + 2, \dots, k_{max}$ :

$$A \quad \theta(t_k) \quad \dot{\theta}(t_k)$$

Η επιλογή του  $k_{trans}$  γίνεται έτσι, ώστε να παραλειφθεί η μεταβατική συμπεριφορά (transient behavior) και να είστε βέβαιοι πως μελετάτε τη μόνιμη κατάσταση του εκκρεμούς. Μπορείτε να πάρετε  $k_{max} = 500$ ,  $k_{trans} = 400$ ,  $t_i = 0$ ,  $t_f = 500\pi$ , και να χωρίσετε τα διαστήματα  $[t_k, t_k + \pi]$  σε 50 υποδιαστήματα. Διαλέξτε  $\theta_0 = 3.1$ ,  $\dot{\theta}_0 = 0$ .

- (α') Φτιάξτε τη γραφική παράσταση του διαγράμματος διακλάδωσης που προκύπτει τοποθετώντας σε διάγραμμα τα σημεία  $(A, \theta(t_k))$ .
- (β') Επαναλάβετε τοποθετώντας σε διάγραμμα τα σημεία  $(A, \dot{\theta}(t_k))$ .
- (γ') Εξετάστε αν τα αποτελέσματά σας εξαρτώνται από την επιλογή των  $\theta_0$ ,  $\dot{\theta}_0$  επαναλαμβάνοντας για διαφορετικές τιμές, λ.χ.  $\theta_0 = 0$ ,  $\theta_0 = 1$ .
- (δ') Μελετήστε την περιοχή που ξεκινάει η χαοτική συμπεριφορά: Πάρτε  $A \in [1.0000, 1.0400]$  με  $\delta A = 0.0001$  και  $A \in [4.4300, 4.4500]$  με  $\delta A = 0.0001$  και βρείτε με τη δεδομένη ακρίβεια την τιμή  $A_c$  που ξεκινάει η χαοτική συμπεριφορά.
- (ε') Στη συνέχεια να αναπαραστήσετε γραφικά τα σημεία  $(\theta(t_k), \dot{\theta}(t_k))$  για  $A = 1.034, 1.040, 1.080, 1.400, 4.450, 4.600$ . Τοποθετήστε 2000 σημεία για κάθε τιμή του  $A$  και σχολιάστε πότε η χαοτική συμπεριφορά είναι εντονότερη.



# ΚΕΦΑΛΑΙΟ 5

## Κίνηση στο Επίπεδο

Στο κεφάλαιο αυτό θα επεκτείνουμε τη μελέτη του προηγούμενου κεφαλαίου στη μελέτη κίνησης σωματιδίου υπό την επίδραση δύναμης στο επίπεδο. Ιδιαίτερο ενδιαφέρον παρουσιάζει το πρόβλημα της κίνησης σε κεντρικό δυναμικό πεδίο, όπως η πλανητική κίνηση και το πρόβλημα της σκέδασης. Τα προβλήματα αυτά μπορούν να μελετηθούν με απλές μεθόδους Runge–Kutta, οπότε, από άποψη αριθμητικής ανάλυσης, το κεφάλαιο αυτό είναι απλή εφαρμογή των μεθόδων που έχουν ήδη μελετηθεί.

### 5.1 Runge–Kutta για την Κίνηση στο Επίπεδο

Στις δύο διαστάσεις, το πρόβλημα αρχικών τιμών που έχουμε να λύσουμε δίνεται από το σύστημα (4.6)

$$\begin{aligned}\frac{dx}{dt} &= v_x & \frac{dv_x}{dt} &= a_x(t, x, v_x, y, v_y) \\ \frac{dy}{dt} &= v_y & \frac{dv_y}{dt} &= a_y(t, x, v_x, y, v_y).\end{aligned}\tag{5.1}$$

Ο κώδικας που θα τρέχει τη μέθοδο Runge–Kutta 4ης τάξης προκύπτει με μικρές μετατροπές του κώδικα `rk.f90`. Κατ' αρχήν, για διευκόλυνση της μελέτης διαφορετικών δυνάμεων ξεχωρίζουμε τον κοινό κώδικα με το `user interface` και τον αλγόριθμο της μεθόδου από τις συναρτήσεις της επιτάχυνσης που αλλάζουν ανάλογα με τη δύναμη σε ξεχωριστά αρχεία. Στο αρχείο `rk2.f90` τοποθετούμε τα πρώτα και σε αρχείο `rk_XXX.f90` τα δεύτερα. XXX είναι ακολουθία χαρακτήρων που ταυτοποιούν τη δύναμη λ.χ. `rk2_hoc.f90` έχει την επιτάχυνση του αρμονικού ταλαντωτή, `rk2_g.f90` την επιτάχυνση από ομογενές πεδίο βαρύτητας  $\vec{g} = -g \hat{y}$  κ.ο.κ.

Στον κώδικα στο rk2.f90 κάνουμε μερικές μικροαλλαγές στο user interface. Τοποθετούμε δυο απροσδιόριστες σταθερές σύζευξης  $k_1$ ,  $k_2$  που μπορούν να δοθούν διαδραστικά από το χρήστη και να καθορίσουν το μέγεθος της δύναμης που ασκείται κάθε φορά στο σώμα. Τοποθετούνται σε common block

```
real(8) ::      k1,k2
common /couplings/k1,k2
```

το οποίο θα “φαίνεται” και από τις συναρτήσεις επιτάχυνσης  $f_3$ ,  $f_4$  και της ενέργειας energy. Ο χρήστης πρέπει τώρα να παρέχει τις αρχικές συνθήκες και για τις δύο συντεταγμένες στο επίπεδο  $x$ ,  $y$ . Αυτές αντιστοιχούν στις μεταβλητές  $X10 \leftrightarrow x_0$ ,  $X20 \leftrightarrow y_0$ ,  $V10 \leftrightarrow v_{x0}$ ,  $V20 \leftrightarrow v_{y0}$ , ενώ οι συναρτήσεις του χρόνου αντιστοιχούν στα arrays  $X1(P) \leftrightarrow x(t)$ ,  $X2(P) \leftrightarrow y(t)$ ,  $V1(P) \leftrightarrow v_x(t)$ ,  $V2(P) \leftrightarrow v_y(t)$ . Η ολοκλήρωση γίνεται όπως και πριν καλώντας

```
call RK(T,X1,X2,V1,V2,Ti,Tf,X10,X20,V10,V20,Nt)
```

και στο αρχείο rk2.dat αποθηκεύουμε τα αποτελέσματα μαζί με τη συνολική μηχανική ενέργεια που υπολογίζεται από τη συνάρτηση energy( $t, x_1, x_2, v_1, v_2$ ) η οποία βρίσκεται στο ίδιο αρχείο με τις επιταχύνσεις μια και η μορφή της εξαρτάται από τον τύπο της δύναμης:

```
open(unit=11,file='rk2.dat')
do i=1,Nt
  write(11,*)T(i),X1(i),X2(i),V1(i),V2(i),&
    energy(T(i),X1(i),X2(i),V1(i),V2(i))
enddo
```

Τέλος, πρέπει να γίνουν αλλαγές στον κώδικα της βασικής υπορουτίνας RKSTEP( $t, x_1, x_2, x_3, x_4, dt$ ) λόγω του μεγαλύτερου αριθμού μεταβλητών στο πρόβλημα. Παραθέτουμε ολόκληρο τον κώδικα για να διευκολύνουμε τον αναγνώστη:

```
!=====
!Program to solve a 4 ODE system using Runge-Kutta Method
!User must supply derivatives
!dx1/dt=f1(t,x1,x2,x3,x4) dx2/dt=f2(t,x1,x2,x3,x4)
!dx3/dt=f3(t,x1,x2,x3,x4) dx4/dt=f4(t,x1,x2,x3,x4)
!as real(8) functions
!Output is written in file rk2.dat
```

```

!=====
program rk2_solve
  implicit none
  integer ,parameter :: P=1010000
  real(8),dimension(P):: T,X1,X2,V1,V2
  real(8) :: Ti,Tf,X10,X20,V10,V20
  integer :: Nt, i
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: energy,E0,EF,DE
!Input:
  print *, 'Runge-Kutta Method for 4-ODEs Integration '
  print *, 'Enter coupling constants:'
  read *, k1,k2
  print *, 'k1= ',k1, ' k2= ',k2
  print *, 'Enter Nt,Ti,Tf,X10,X20,V10,V20:'
  read *, Nt,Ti,TF,X10,X20,V10,V20
  print *, 'Nt = ',Nt
  print *, 'Time: Initial Ti =',Ti, ' Final Tf=',Tf
  print *, '          X1(Ti)=',X10, ' X2(Ti)=',X20
  print *, '          V1(Ti)=',V10, ' V2(Ti)=',V20
!The Calculation:
  call RK(T,X1,X2,V1,V2,Ti,Tf,X10,X20,V10,V20,Nt)
!Output:
  open(unit=11,file='rk2.dat')
  do i=1,Nt
    write(11,*)T(i),X1(i),X2(i),V1(i),V2(i),&
      energy(T(i),X1(i),X2(i),V1(i),V2(i))
  enddo
  close(11)
!Rutherford scattering angles:
  print *, 'v-angle: ',atan2(V2(Nt),V1(Nt))
  print *, 'b-angle: ',2.0D0*atan(k1/(V10*V10*X20))
  E0 = energy(Ti,X10,X20,V10,V20)
  EF = energy(T(Nt),X1(Nt),X2(Nt),V1(Nt),V2(Nt))
  DE = ABS(0.5D0*(EF-E0)/(EF+E0))
  print *, 'E0,EF, DE/E= ',E0,EF,DE
end program rk2_solve
!=====
!The velocity functions f1,f2(t,x1,x2,v1,v2)
!=====
real(8) function f1(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  f1=v1 !dx1/dt= v1
end function f1
!
real(8) function f2(t,x1,x2,v1,v2)
  implicit none

```

```

real(8) :: t,x1,x2,v1,v2
f2=v2          !dx2/dt= v2
end function f2
!=====
!RK(T,X1,X2,V1,V2,Ti,Tf,X10,X20,V10,V20,Nt) is the driver
!for the Runge–Kutta integration routine RKSTEP
!Input: Initial and final times Ti,Tf
!       Initial values at t=Ti  X10,X20,V10,V20
!       Number of steps of integration: Nt-1
!       Size of arrays T,X1,X2,V1,V2
!Output: real arrays T(Nt),X1(Nt),X2(Nt),
!        V1(Nt),V2(Nt) where
!T(1) = Ti  X1(1) = X10  X2(1) = X20  V1(1) = V10  V2(1) = V20
!        X1(k) = X1(at t=T(k))  X2(k) = X2(at t=T(k))
!        V1(k) = V1(at t=T(k))  V2(k) = V2(at t=T(k))
!T(Nt)= Tf
!=====
subroutine RK(T,X1,X2,V1,V2,Ti,Tf,X10,X20,V10,V20,Nt)
implicit none
integer :: Nt
real(8),dimension(Nt)::T,X1,X2,V1,V2
real(8) :: Ti ,Tf
real(8) :: X10,X20
real(8) :: V10,V20
real(8) :: dt
real(8) :: TS,X1S,X2S !values of time and X1,X2 at given step
real(8) :: V1S,V2S
integer :: i
!Initialize variables:
dt      = (Tf-Ti)/(Nt-1)
T (1)   = Ti
X1(1)   = X10; X2(1) = X20
V1(1)   = V10; V2(1) = V20
TS      = Ti
X1S     = X10; X2S   = X20
V1S     = V10; V2S   = V20
!Make RK steps: The arguments of RKSTEP are
!replaced with the new ones
do i=2,Nt
  call RKSTEP(TS,X1S,X2S,V1S,V2S,dt)
  T(i) = TS
  X1(i) = X1S; X2(i) = X2S
  V1(i) = V1S; V2(i) = V2S
enddo
end subroutine RK
!=====
!Subroutine RKSTEP(t,x1,x2,dt)
!Runge–Kutta Integration routine of ODE
!dx1/dt=f1(t,x1,x2,x3,x4) dx2/dt=f2(t,x1,x2,x3,x4)

```

```

!dx3/dt=f3(t,x1,x2,x3,x4) dx4/dt=f4(t,x1,x2,x3,x4)
!User must supply derivative functions:
!real function f1(t,x1,x2,x3,x4)
!real function f2(t,x1,x2,x3,x4)
!real function f3(t,x1,x2,x3,x4)
!real function f4(t,x1,x2,x3,x4)
!Given initial point (t,x1,x2) the routine advances it
!by time dt.
!Input : Initial time t and function values x1,x2,x3,x4
!Output: Final time t+dt and function values x1,x2,x3,x4
!Careful!: values of t,x1,x2,x3,x4 are overwritten...
!=====
subroutine RKSTEP(t,x1,x2,x3,x4,dt)
  implicit none
  real(8) :: t,x1,x2,x3,x4,dt
  real(8) :: f1,f2,f3,f4
  real(8) :: k11,k12,k13,k14,k21,k22,k23,k24
  real(8) :: k31,k32,k33,k34,k41,k42,k43,k44
  real(8) :: h,h2,h6

  h =dt          !h =dt, integration step
  h2=0.5D0*h     !h2=h/2
  h6=h/6.0D0    !h6=h/6

  k11=f1(t,x1,x2,x3,x4)
  k21=f2(t,x1,x2,x3,x4)
  k31=f3(t,x1,x2,x3,x4)
  k41=f4(t,x1,x2,x3,x4)

  k12=f1(t+h2,x1+h2*k11,x2+h2*k21,x3+h2*k31,x4+h2*k41)
  k22=f2(t+h2,x1+h2*k11,x2+h2*k21,x3+h2*k31,x4+h2*k41)
  k32=f3(t+h2,x1+h2*k11,x2+h2*k21,x3+h2*k31,x4+h2*k41)
  k42=f4(t+h2,x1+h2*k11,x2+h2*k21,x3+h2*k31,x4+h2*k41)

  k13=f1(t+h2,x1+h2*k12,x2+h2*k22,x3+h2*k32,x4+h2*k42)
  k23=f2(t+h2,x1+h2*k12,x2+h2*k22,x3+h2*k32,x4+h2*k42)
  k33=f3(t+h2,x1+h2*k12,x2+h2*k22,x3+h2*k32,x4+h2*k42)
  k43=f4(t+h2,x1+h2*k12,x2+h2*k22,x3+h2*k32,x4+h2*k42)

  k14=f1(t+h ,x1+h *k13,x2+h *k23,x3+h *k33,x4+h2*k43)
  k24=f2(t+h ,x1+h *k13,x2+h *k23,x3+h *k33,x4+h2*k43)
  k34=f3(t+h ,x1+h *k13,x2+h *k23,x3+h *k33,x4+h2*k43)
  k44=f4(t+h ,x1+h *k13,x2+h *k23,x3+h *k33,x4+h2*k43)

  t =t+h
  x1=x1+h6*(k11+2.0D0*(k12+k13)+k14)
  x2=x2+h6*(k21+2.0D0*(k22+k23)+k24)
  x3=x3+h6*(k31+2.0D0*(k32+k33)+k34)
  x4=x4+h6*(k41+2.0D0*(k42+k43)+k44)

```

```
end subroutine RKSTEP
```

## 5.2 Βολές στο Βαρυτικό Πεδίο της Γης

Θεωρούμε αρχικά σωματίδιο υπό την επίδραση δύναμης που του προσδίδει επιτάχυνση  $\vec{g} = -g \hat{y}$

$$\begin{aligned} x(t) &= x_0 + v_{0x}t, & y(t) &= y_0 + v_{0y}t - \frac{1}{2}gt^2 \\ v_x(t) &= v_{0x}, & v_y(t) &= v_{0y} - gt \\ a_x(t) &= 0, & a_y(t) &= -g \end{aligned} \quad (5.2)$$

Το σωματίδιο, όπως γνωρίζουμε καλά, κινείται πάνω σε μια παραβολή στην οποία εμείς απλά διαλέγουμε το σημείο στο οποίο τοποθετείται αρχικά το σωματίδιο:

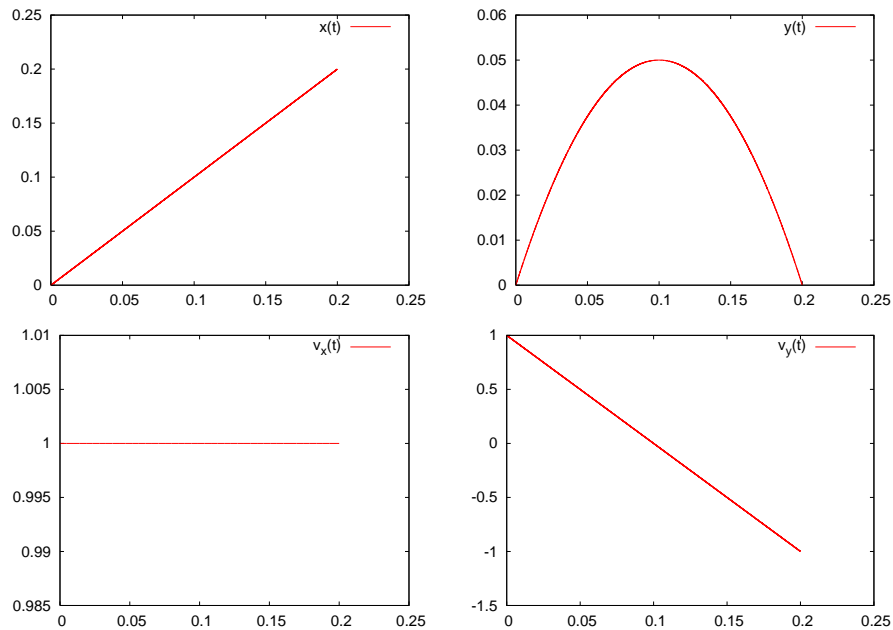
$$\begin{aligned} (y - y_0) &= \left( \frac{v_{0y}}{v_{0x}} \right) (x - x_0) - \frac{1}{2} \frac{g}{v_{0x}^2} (x - x_0)^2 \\ &= \tan \theta (x - x_0) - \frac{\tan^2 \theta}{4h_{\max}} (x - x_0)^2, \end{aligned} \quad (5.3)$$

όπου  $\tan \theta = v_{0y}/v_{0x}$  και  $h_{\max}$  η γωνία υπό την οποία βάλλεται το σωματίδιο και το μέγιστο ύψος που φτάνει το σωματίδιο ως προς το αρχικό σημείο βολής.

Κωδικοποιούμε την επιτάχυνση  $a_x(t) = 0$   $a_y(t) = -g$  ( $a_x \leftrightarrow f3$ ,  $a_y \leftrightarrow f4$ ) καθώς και τη συνολική μηχανική ενέργεια στο αρχείο `rk2_g.f90`:

```
!=====
!The acceleration functions f3,f4(t,x1,x2,v1,v2) provided
!by the user
!=====
!Free fall in constant gravitational field with
!g = -k2
real(8) function f3(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/ k1,k2
  f3=0.0D0      !dx3/dt=dv1/dt=a1
end function f3
!-----
real(8) function f4(t,x1,x2,v1,v2)
  implicit none
```





Σχήμα 5.1: Βολή στο βαρυτικό πεδίο με επιτάχυνση  $\vec{g} = -10.0\hat{y}$  και αρχική ταχύτητα  $\vec{v}_0 = \hat{x} + \hat{y}$ . Δίνονται τα διαγράμματα  $x(t)$ ,  $y(t)$ ,  $v_x(t)$ ,  $v_y(t)$ .

```

real(8) :: t,x1,x2,v1,v2
real(8) :: k1,k2
common /couplings/k1,k2
f4=-k1 !dx4/dt=dv2/dt=a2
end function f4
!
real(8) function energy(t,x1,x2,v1,v2)
implicit none
real(8) :: t,x1,x2,v1,v2
real(8) :: k1,k2
common /couplings/k1,k2
energy = 0.5D0*(v1*v1+v2*v2) + k1*x2
end function energy

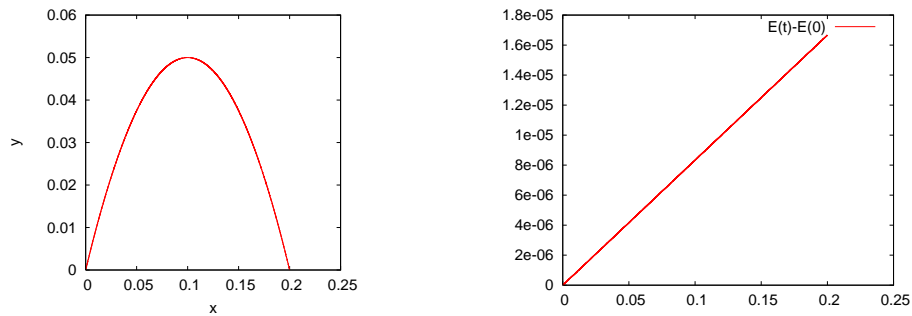
```

Στη συνέχεια παραθέτουμε τη σειρά εντολών που δίνει ο χρήστης για να υπολογίσει την τροχιά

```

> gfortran -O2 rk2.f90 rk2_g.f90 -o rk2
> ./rk2
Runge-Kutta Method for 4-ODEs Integration
Enter coupling constants:
10.0 0.0

```



Σχήμα 5.2: Βολή στο βαρυτικό πεδίο με επιτάχυνση  $\vec{g} = -10.0\hat{y}$  και αρχική ταχύτητα  $\vec{v}_0 = \hat{x} + \hat{y}$ . Φαίνεται η παραβολική τροχιά που ακολουθεί το σωματίο. Στο διπλανό σχήμα παρακολουθούμε την απόκλιση της μηχανικής ενέργειας του σωματιδίου από την αρχική της τιμή.

```

k1=      10.000000      k2=      0.000000
Enter Nt,Ti,Tf,X10,X20,V10,V20:
20000 0.0 0.2 0.0 0.0 1.0 1.0
Nt=      20000
Time: Initial Ti =      0.000000      Final Tf=      0.200000
      X1(Ti)=      0.000000      X2(Ti)=      0.000000
      V1(Ti)=      1.000000      V2(Ti)=      1.000000

```

Στη συνέχεια, επεξεργαζόμαστε τα αποτελέσματά μας αναλύοντας τα δεδομένα από το αρχείο `rk2.dat` με το πρόγραμμα `gnuplot`:

```

gnuplot> set terminal x11 1
gnuplot> plot "rk2.dat" using 1:2 with lines title "x(t)"
gnuplot> set terminal x11 2
gnuplot> plot "rk2.dat" using 1:3 with lines title "y(t)"
gnuplot> set terminal x11 3
gnuplot> plot "rk2.dat" using 1:4 with lines title "vx(t)"
gnuplot> set terminal x11 4
gnuplot> plot "rk2.dat" using 1:5 with lines title "vy(t)"
gnuplot> set terminal x11 5
gnuplot> plot "rk2.dat" using 1:($6-1.0) w lines t "E(t)E-(0)"
gnuplot> set terminal x11 6
gnuplot> set size square
gnuplot> set title "Trajectory"
gnuplot> plot "rk2.dat" using 2:3 with lines notit

```

Τα αποτελέσματα φαίνονται στο σχήματα 5.1 και 5.2. Παρατηρούμε μικρή αύξηση της ενέργειας που μας δίνει και το μέτρο της ακρίβειας της μεθόδου.

Με τη βοήθεια του `gnuplot` μπορούμε να φτιάξουμε κινούμενα σχέ-

δια της τροχιάς. Για το λόγο αυτό ομαδοποιούμε μερικές εντολές του gnuplot σε αρχείο σεναρίου, έστω στο `rk2_animate.gpl`

```
icount = icount+skip
plot "<cat -n rk2.dat" \
    using 3:($1<= icount ? $4: 1/0) with lines notitle
# pause 1
if(icount < nlines ) reread
```

Το παραπάνω αρχείο υποθέτει ότι, όταν τρέξουμε το gnuplot, έχουμε αρχικοποιήσει τις μεταβλητές `icount`, `skip`, `nlines` να είναι οι τιμές του αρχικού αριθμού γραμμών του αρχείου `rk2.dat` που θα μπουν στο διάγραμμα, ο αριθμός γραμμών που θα προστίθενται σε κάθε καινούργιο πλαίσιο που σχεδιάζεται στα κινούμενα σχέδια και ο συνολικός αριθμός γραμμών που περιέχει το αρχείο, ώστε να σταματήσει η διαδικασία. Η ιδέα είναι ότι οι εντολές του αρχείου διαβάζονται από το gnuplot κάνοντας ένα `plot` και αν πληρείται το κριτήριο του `if` το αρχείο ξαναδιαβάζεται με την εντολή `reread`. Ας εξηγήσουμε την γραμμή με την εντολή `plot`: Το “αρχείο” `"<cat -n rk2.dat"` είναι το standard output της εντολής `cat -n rk2.dat` η οποία τυπώνει στο standard output το αρχείο `rk2.dat` βάζοντας στην πρώτη στήλη τον αριθμό γραμμής που διαβάζεται. Έτσι η εντολή `plot` διαβάζει δεδομένα στα οποία η πρώτη στήλη είναι ο αριθμός γραμμής, η δεύτερη ο χρόνος, η τρίτη η συντεταγμένη  $x$ , η τέταρτη η συντεταγμένη  $y$  κ.ο.κ. Η γραμμή

```
using 3:($1<= icount ? $4: 1/0)
```

λέει να χρησιμοποιηθεί η 3η στήλη στον οριζόντιο άξονα και αν η πρώτη στήλη είναι μικρότερη από την τιμή της μεταβλητής `icount` να μπει στον κατακόρυφο άξονα η τιμή της 4ης στήλης αλλιώς τίποτα (βάζοντας κάτι που δεν είναι νόμιμο, όπως διαίρεση με το 0 κάνει το gnuplot να αγνοήσει το συγκεκριμένο σημείο). Με τον τρόπο αυτό, καθώς η τιμή της μεταβλητής `icount` αυξάνει, τοποθετούμε στο διάγραμμα περισσότερα σημεία της τροχιάς δημιουργώντας την ψευδαίσθηση της κίνησης. Τη γραμμή με την εντολή `pause` την έχουμε βάλει ως σχόλιο. Αν τα κινούμενα σχέδια είναι πολύ γρήγορα για σας, βγάλτε το χαρακτήρα του σχολίου `#` και αντικαταστήστε τη μονάδα με τον αριθμό δευτερολέπτων που θέλετε να σταματάει κάθε πλαίσιο. Για να χρησιμοποιήσουμε το σενάριο αυτό από το gnuplot δίνουμε τις εντολές

```
gnuplot> icount = 10
```

```
gnuplot> skip    = 200
gnuplot> nlines  = 20000
gnuplot> load    "rk2_animate.gpl"
```

Τα παραπάνω σενάρια θα τα βρείτε στο συνοδευτικό λογισμικό του κεφαλαίου. Εκεί θα βρείτε και σενάρια φλοιού τα οποία θα σας βοηθήσουν να αυτοματοποιήσετε πολλές από τις εντολές που περιγράψαμε παραπάνω. Περιγράφουμε τη χρήση δύο από αυτών. Πρώτα το σενάριο `rk2_animate.csh`:

```
> rk2_animate.csh -h
Usage: rk2_animate.csh -t [sleep time] -d [skip points] <file>
Default file is rk2.dat
Other options:
  -x: set lower value in xrange
  -X: set lower value in xrange
  -y: set lower value in yrange
  -Y: set lower value in yrange
  -r: automatic determination of x-y range
> rk2_animate.csh -r -d 500 rk2.dat
```

Η τελευταία γραμμή πραγματοποιεί τα κινούμενα σχέδια με πλαίσια που κάθε φορά έχουν 500 παραπάνω σημεία, ενώ τα όρια των πλαισίων υπολογίζονται αυτόματα από το σενάριο με το διακόπτη `-r`. Ο διακόπτης `-h` δίνει σύντομες οδηγίες για τη χρήση του σεναρίου, μια σύμβαση που την ακολουθούμε συχνά στα σενάρια/προγράμματα που γράφουμε.

Ένα πιο πλήρες σενάριο που κάνει όλες δουλειές είναι το `rk2.csh`. Οδηγίες χρήσης παίρνουμε με την εντολή

```
> ./rk2.csh -h
Usage: rk2.csh -f <force> k1 k2 x10 x20 v10 v20 STEPS t0 tf
Other Options:
  -n Do not animate trajectory
Available forces (value of <force>):
1: ax=-k1          ay= -k2 y          Harmonic oscillator
2: ax= 0           ay= -k1           Free fall
3: ax= -k2         vx          ay= -k2   vy - k1   Free fall + \
                                     air resistance ~ v
4: ax= -k2 |v| vx   ay= -k2 |v|vy - k1   Free fall + \
                                     air resistance ~ v^2
5: ax= k1*x1/r^3    ay= k1*x2/r^3      Coulomb Force
....
```

όπου φαίνεται ότι έχουμε την επιλογή να τρέξουμε το πρόγραμμα με διαφορετικές δυνάμεις που επιλέγονται με το διακόπτη `-f`. Στην υπό-

λοιπη γραμμή εντολών δίνουμε τα δεδομένα εισόδου για το πρόγραμμα rk2.f90, τις σταθερές ζεύξης  $k_1$ ,  $k_2$ , τις αρχικές συνθήκες  $x_{10}$ ,  $x_{20}$ ,  $v_{10}$ ,  $v_{20}$  και τις συνθήκες ολοκλήρωσης STEPS,  $t_0$ ,  $t_f$ . Έτσι για παράδειγμα οι εντολές

```
> rk2.csh -f 2 -- 10.0 0.0 0.0 0.0 1.0 1.0 20000 0.0 0.2
> rk2.csh -f 1 -- 16.0 1.0 0.0 1.0 1.0 0.0 20000 0.0 6.29
> rk2.csh -f 5 -- 10.0 0.0 -10 0.2 10. 0.0 20000 0.0 3.00
```

μας δίνουν την κίνηση του σωματιδίου στο πεδίο βαρύτητας που μελετήσαμε ως τώρα, την κίνηση ανομοιογενούς αρμονικού ταλαντωτή ( $k_1 = a_x = -\omega_1^2 x$ ,  $k_2 = a_y = -\omega_2^2 y$ ) και τη σκέδαση φορτίου σε πεδίο Coulomb - δοκιμάστε τα! Ελπίζω να σας δημιουργηθεί και η περιέργεια να δείτε “μέσα” στα σενάρια, έτσι ώστε να τα τροποποιείτε και δημιουργείτε από μόνοι/ες σας. Από μένα μερικές οδηγίες για τους τεμπέληδες: Αν θελήσετε να προσθέσετε μια δική σας δύναμη στο ρεπερτόριο του σεναρίου ακολουθήστε τη συνταγή: Προγραμματίστε τη δύναμή σας σε ένα αρχείο με όνομα rk2\_myforce.f90 σύμφωνα με τις προδιαγραφές του rk2\_g.f90. Επεξεργαστείτε το αρχείο rk2.csh και αλλάξτε τη γραμμή

```
set forcecode = (hoc g vg v2g cb)
```

σε

```
set forcecode = (hoc g vg v2g cb myforce)
```

(φυσικά μπορεί η μεταβλητή \$forcecode να έχει και άλλες εγγραφές στο σενάριο αλλά αυτό δεν θα σας εμποδίσει). Μετρήστε σε ποια σειρά έχετε βάλει το myforce, εδώ την 6η, και τρέξτε την εντολή με το διακόπτη -f 6 όπου το 6 αντικαταστήστε το με τη σειρά στο δικό σας σενάριο (οι τελίτσες είναι οι δικές σας σταθ. ζεύξης και αρχικές συνθήκες):

```
> rk2.csh -f 6 — .....
```

Ας μελετήσουμε τώρα την επίδραση της αντίστασης του αέρα ή ενός ρευστού στην πτώση/βολή του σωματιδίου. Για μικρές ταχύτητες η αντίσταση γίνεται ανάλογη της ταχύτητας και έχουμε  $\vec{F}_r = -mk\vec{v}$  οπότε

$$\begin{aligned} a_x &= -kv_x \\ a_y &= -kv_y - g. \end{aligned} \quad (5.4)$$

Παίρνοντας

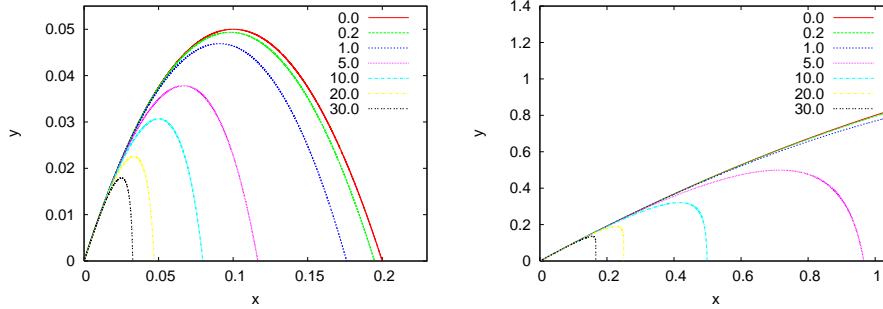
$$\begin{aligned}x(t) &= x_0 + \frac{v_{0x}}{k} (1 - e^{-kt}) \\y(t) &= y_0 + \frac{1}{k} \left( v_{0y} + \frac{g}{k} \right) (1 - e^{-kt}) - \frac{g}{k} t \\v_x(t) &= v_{0x} e^{-kt} \\v_y(t) &= \left( v_{0y} + \frac{g}{k} \right) e^{-kt} - \frac{g}{k},\end{aligned}\tag{5.5}$$

προκύπτει η κίνηση του σωματιδίου με ορισκή ταχύτητα  $v_y(+\infty) = -g/k$  ( $x(+\infty) = \text{σταθ.}, y(+\infty) \sim t$ ).

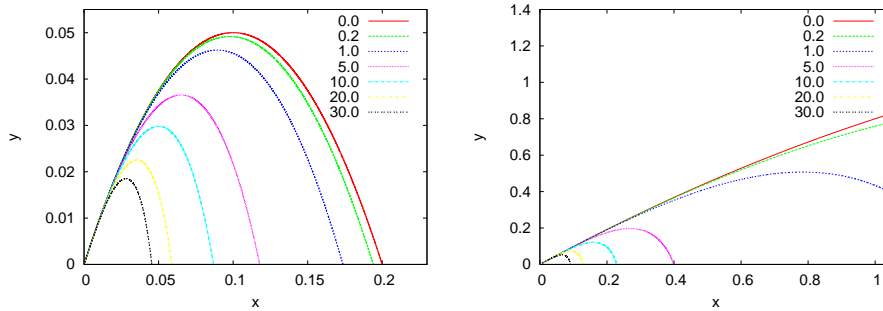
Ο προγραμματισμός της επιτάχυνσης καταγράφεται στο αρχείο (k1  $\leftrightarrow g$ , k2  $\leftrightarrow k$ ) rk2\_vg.f90:

```
!=====
!The acceleration functions f3,f4(t,x1,x2,v1,v2) provided
!by the user
!=====
!Free fall in constant gravitational field with
!ax = -k2 vx    ay = -k2 vy - k1
real(8) function f3(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  f3=-k2*v1    !dx3/dt=dv1/dt=a1
end function f3
!-----
real(8) function f4(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  f4=-k2*v2-k1    !dx4/dt=dv2/dt=a2
end function f4
```

Τα αποτελέσματα καταγράφονται στα σχήματα 5.3 όπου φαίνεται η επίδραση της αυξανόμενης αντίστασης στην τροχιά του σωματιδίου. Στο σχήμα 5.4 δίνεται για σύγκριση η επίδραση δύναμης  $\vec{F}_r = -mkv^2\hat{v}$ .



Σχήμα 5.3: Τροχιά σωματιδίου που βάλλεται στο σταθερό βαρυτικό πεδίο της γης  $\vec{g} = -10\hat{y}$  υπό την επίδραση αντίστασης ρευστού  $\vec{a}_r = -k\vec{v}$  για  $k = 0, 0.2, 1, 5, 10, 20, 30$ . Αριστερά έχουμε  $\vec{v}(0) = \hat{x} + \hat{y}$ , ενώ δεξιά  $\vec{v}(0) = 5\hat{x} + 5\hat{y}$ .



Σχήμα 5.4: Τροχιά σωματιδίου που βάλλεται στο σταθερό βαρυτικό πεδίο της γης  $\vec{g} = -10\hat{y}$  υπό την επίδραση αντίστασης ρευστού  $\vec{a}_r = -k\vec{v}^2\hat{v}$  για  $k = 0, 0.2, 1, 5, 10, 20, 30$ . Αριστερά έχουμε  $\vec{v}(0) = \hat{x} + \hat{y}$ , ενώ δεξιά  $\vec{v}(0) = 5\hat{x} + 5\hat{y}$ .

### 5.3 Κίνηση Πλανητών

Θα θεωρήσουμε το απλό πλανητικό μοντέλο του “ήλιου” με μάζα  $M$  και ενός πλανήτη “γη” με μάζα  $m$  έτσι ώστε  $m \ll M$ . Ο νόμος του Νεύτωνα μας δίνει ότι η επιτάχυνση της “γης” δίνεται από τη σχέση

$$\vec{a} = \vec{g} = -\frac{GM}{r^2}\hat{r} = -\frac{GM}{r^3}\vec{r}. \quad (5.6)$$

Θυμίζουμε στον αναγνώστη ότι  $G = 6.67 \times 10^{-11} \frac{\text{m}^3}{\text{kgr} \cdot \text{sec}^2}$ ,  $M = 1.99 \times 10^{30} \text{kgr}$ ,  $m = 5.99 \times 10^{24} \text{kgr}$ . Επίσης, όταν η υπόθεση  $m \ll M$  δεν είναι ικανοποιητική, τότε το πρόβλημα των δύο σωμάτων ανάγεται σε αυτό του ενός χρησιμοποιώντας την ανηγμένη μάζα

$$\frac{1}{\mu} = \frac{1}{m} + \frac{1}{M}.$$

Η δύναμη της βαρύτητας είναι κεντρική με αποτέλεσμα να διατηρείται η στροφορμή  $\vec{L} = \vec{r} \times \vec{p}$ . Αυτό σημαίνει ότι η κίνηση γίνεται πάνω σε ένα επίπεδο και μπορούμε να πάρουμε τον άξονα των  $z$ , έτσι ώστε

$$\vec{L} = L_z \hat{k} = m(xv_y - yv_x)\hat{k}. \quad (5.7)$$

Η δύναμη είναι διατηρητική και η ενέργεια

$$E = \frac{1}{2}mv^2 - \frac{GmM}{r} \quad (5.8)$$

διατηρείται. Αν πάρουμε την αρχή των αξόνων να είναι το κέντρο της δύναμης, τότε οι εξισώσεις κίνησης (5.6) γίνονται

$$\begin{aligned} a_x &= -\frac{GM}{r^3}x \\ a_y &= -\frac{GM}{r^3}y \end{aligned} \quad (5.9)$$

με  $r^2 = x^2 + y^2$ . Οι εξισώσεις αυτές είναι ένα σύστημα δύο συζευγμένων διαφορικών εξισώσεων ως προς τις συναρτήσεις  $x(t)$ ,  $y(t)$ . Οι λύσεις είναι κωνικές τομές που είναι είτε έλλειψη (δεσμευμένη τροχιά - “πλανήτης”), είτε παραβολή (για τη λεγόμενη “ταχύτητα διαφυγής”), είτε υπερβολή (σκέδαση).

Για την περίοδο περιστροφής των πλανητών ισχύει ο τρίτος νόμος του Κέπλερ

$$T^2 = \frac{4\pi^2}{GM}a^3 \quad (5.10)$$

όπου εδώ  $a$  είναι ο μεγάλος ημιάξονας της ελλειπτικής τροχιάς και  $b$  ο μικρός ημιάξονας. Το πόσο “πλατιά” είναι η έλλειψη χαρακτηρίζεται από την εκκεντρότητα της τροχιάς

$$e = \sqrt{1 - \frac{b^2}{a^2}}, \quad (5.11)$$

η οποία είναι 0 για τον κύκλο και τείνει προς τη 1 όταν την “πατάμε” να γίνει ευθεία. Σε απόσταση  $ea$  από το κέντρο της έλλειψης βρίσκονται οι εστίες της  $F_1$  και  $F_2$ . Αυτές έχουν την ιδιότητα ότι κάθε σημείο  $P$  της τροχιάς έχει

$$PF_1 + PF_2 = 2a. \quad (5.12)$$

Για να προγραμματίσουμε τη δύναμη του Νεύτωνα γράφουμε στο αρχείο rk2\_cb.f90:



```

=====
!The acceleration functions f3,f4(t,x1,x2,v1,v2) provided
!by the user
=====
!Motion in Coulombic potential:
!ax= k1*x1/r^3 ay= k1*x2/r^3
real(8) function f3(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: r2,r3
  r2=x1*x1+x2*x2
  r3=r2*sqrt(r2)
  if(r3.gt.0.0D0)then
    f3=k1*x1/r3          !dx3/dt=dv1/dt=a1
  else
    f3=0.0D0
  endif
end function f3
!
real(8) function f4(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: r2,r3
  r2=x1*x1+x2*x2
  r3=r2*sqrt(r2)
  if(r3.gt.0.0D0)then
    f4=k1*x2/r3          !dx4/dt=dv2/dt=a2
  else
    f4=0.0D0
  endif
end function f4
!
real(8) function energy(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: r
  r=sqrt(x1*x1+x2*x2)
  if( r .gt. 0.0D0)then
    energy = 0.5D0*(v1*v1+v2*v2) + k1/r
  else
    energy = 0.0D0
  endif
end function

```

`end function energy`

Στο παραπάνω πρόγραμμα  $k_1 = -GM$  και έχουμε προσέξει την περίπτωση το σωματίο να προσκρούσει στο ιδιάζον σημείο  $(0, 0)$ , το κέντρο της δύναμης. Προφανώς ο ίδιος κώδικας μπορεί να χρησιμοποιηθεί και για το ηλεκτροστατικό πεδίο Coulomb με  $k_1 = qQ/4\pi\epsilon_0 m$ .

Κατ' αρχήν μελετάμε τροχιές οι οποίες είναι δέσμιες. Διαλέγουμε  $GM = 10$ ,  $x(0) = 1.0$ ,  $y(0) = 0$ ,  $v_{0x} = 0$  και  $v_{0y}$  μεταβλητό. Μετράμε την περίοδο και το μήκος των ημιαξόνων της έλλειψης. Προκύπτει ο πίνακας 5.1. Μερικές από τις τροχιές φαίνονται στο σχήμα 5.5 όπου φαίνεται η

$v_{0x}$	$T/2$	$2a$
3.2	1.032	2.051
3.4	1.282	2.371
3.6	1.682	2.841
3.8	2.398	3.598
4.0	3.9288	5.002
4.1	5.5164	6.272
4.2	8.6952	8.481
4.3	16.95	13.256
4.35	28.168	18.6
4.38	42.81	24.58
4.40	61.8	31.393
4.42	99.91	43.252

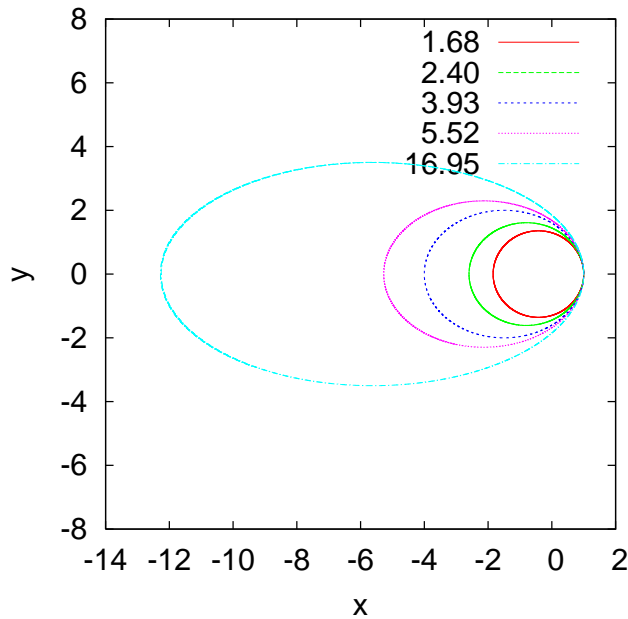
Πίνακας 5.1: Τα αποτελέσματα για την περίοδο και τον μεγάλο ημιάξονα της ελλειπτικής τροχιάς πλανητικής κίνησης για  $GM = 10$ ,  $x(0) = 1.0$ ,  $y(0) = 0$ ,  $v_{0y} = 0$ .

εξάρτηση του μεγέθους της έλλειψης από την περίοδο. Στο σχήμα 5.6, επιβεβαιώνουμε τον 3ο νόμο του Κέπλερ, Σχέση (5.10).

Πώς θα μπορούσαμε να προβλέψουμε το νόμο του Κέπλερ χωρίς να γνωρίζαμε το αποτέλεσμα εκ των προτέρων; Αν πάρουμε το λογάριθμο και στα δύο μέλη της εξίσωσης (5.10) προκύπτει:

$$\ln T = \frac{3}{2} \ln a + \frac{1}{2} \ln \left( \frac{4\pi^2}{GM} \right) \quad (5.13)$$

Άρα σε ένα διάγραμμα των σημείων  $(\ln a, \ln T)$  τα σημεία πρέπει να βρίσκονται πάνω σε μια ευθεία. Με τη μέθοδο των ελαχίστων τετραγώνων μπορούμε να υπολογίσουμε το συντελεστή κατεύθυνσης και το σημείο τομής των αξόνων που θα πρέπει να είναι  $\frac{3}{2}$  και  $\frac{1}{2} \ln (4\pi^2/GM)$  αντίστοιχα. Το αφήνουμε σαν άσκηση για τον αναγνώστη.



Σχήμα 5.5: Τροχιές πλανήτη για  $GM = 10$ ,  $x(0) = 1.0$ ,  $y(0) = 0$ ,  $v_{0y} = 0$  και  $v_{0x} = 3.6, 3.8, 4.0, 4.1, 4.3$ . Αναγράφονται οι αντίστοιχες ημιπερίοδοι.

Σε περίπτωση που η αρχική ταχύτητα του σωματιδίου υπερβεί την ταχύτητα διαφυγής  $v_e$ , το σωματίο ξεφεύγει από την επίδραση του πεδίου βαρύτητας. Αυτό γίνεται όταν η μηχανική του ενέργεια (5.8) είναι 0 ή όταν

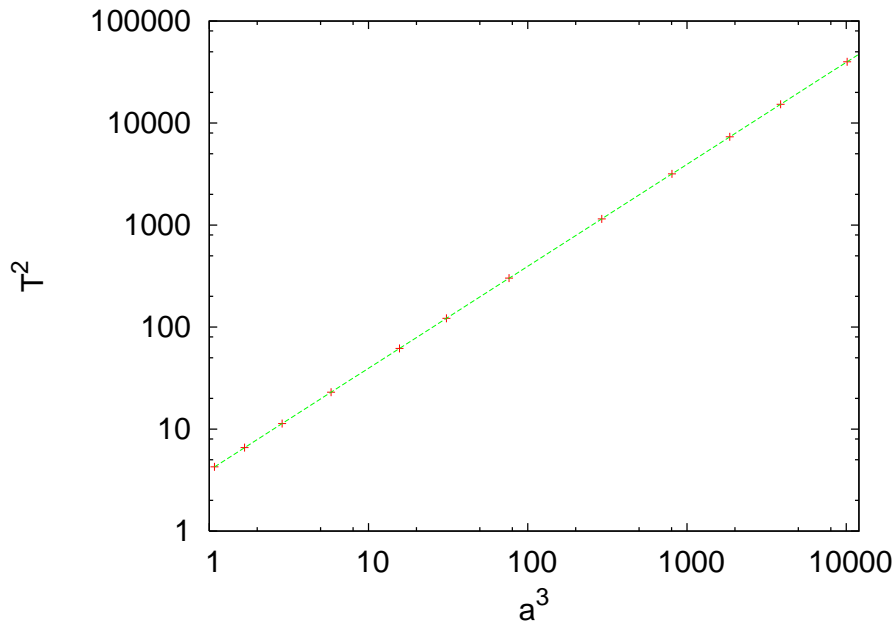
$$v_e^2 = \frac{2GM}{r}, \quad (5.14)$$

που στην περίπτωση που εξετάζουμε με  $GM = 10$ ,  $x(0) = 1.0$ ,  $y(0) = 0$ , παίρνουμε  $v_e \approx 4.4721 \dots$ . Αφήνουμε για άσκηση στον αναγνώστη τον αριθμητικό προσδιορισμό της  $v_e$ .

## 5.4 Σκέδαση

Στην παράγραφο αυτή θεωρούμε σκέδαση σωματιδίων από ένα κεντρικό δυναμικό <sup>1</sup>. Υποθέτουμε ότι στο δυναμικό αυτό υπάρχουν τροχιές που ξεκινούν από το άπειρο και καταλήγουν στο άπειρο, στο οποίο τα σωματίδια κινούνται σχεδόν ελεύθερα από την επίδραση της δύναμης. Έτσι αρχικά τα σωματίδια κινούνται ελεύθερα προς την περιοχή της

<sup>1</sup>Διαβάστε το κεφάλαιο 4 του [37]



Σχήμα 5.6: Ο τρίτος νόμος του Κέπλερ για  $GM = 10$ . Τα σημεία είναι οι μετρήσεις από τον πίνακα 5.1 και η γραμμή που τα συνδέει είναι η αναλυτική λύση (5.10).

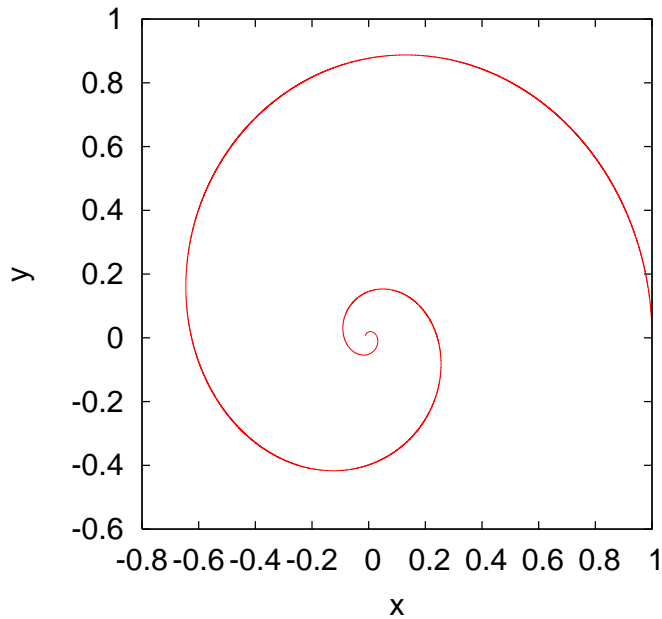
αλληλεπίδρασης μέσα στην οποία αλλάζουν κατεύθυνση και κινούνται πάλι έξω από αυτή σε διαφορετική διεύθυνση. Λέμε τότε ότι το σωματίο σκεδάστηκε και ότι η γωνία μεταξύ της αρχικής και τελικής διεύθυνσης της ταχύτητας είναι η γωνία σκέδασης  $\theta$ . Το ενδιαφέρον στην περίπτωση αυτή έγκειται στο γεγονός ότι από την κατανομή της γωνίας σκέδασης μιας δέσμης σωματιδίων μπορούμε να πάρουμε χρήσιμη πληροφορία για το δυναμικό σκέδασης. Αυτή η τεχνική χρησιμοποιείται κατά κόρον στους σημερινούς επιταχυντές για την μελέτη των θεμελιωδών αλληλεπιδράσεων των στοιχειωδών σωματιδίων.

Για να κατανοήσουμε τους ορισμούς είναι χρήσιμο να θεωρήσουμε τη σκέδαση μικρών σκληρών σφαιρών ακτίνας  $r_1$  από άλλες σκληρές σφαίρες ακτίνας  $R_2$ . Το δυναμικό αλληλεπίδρασης<sup>2</sup> είναι δηλαδή:

$$V(r) = \begin{cases} 0 & r > R_2 + r_1 \\ \infty & r < R_2 + r_1 \end{cases}, \quad (5.15)$$

όπου  $r$  είναι η απόσταση του κέντρου της  $r_1$  από το κέντρο της  $R_2$ . Υποθέτουμε ότι τα σωματίδια της δέσμης δεν αλληλεπιδρούν μεταξύ τους και ότι κατά τη σκέδαση κάθε σωματίο αλληλεπιδρά μόνο με ένα

<sup>2</sup>Λέγεται δυναμικό σκληρού πυρήνα (hard core potential).



Σχήμα 5.7: Σπειροειδής τροχιά σωματιδίου που κινείται υπό την επίδραση κεντρικής δύναμης  $\vec{F} = -k/r^3\hat{r}$ .

κέντρο σκέδασης του στόχου. Έστω  $J$  η πυκνότητα ροής ή ένταση της δέσμης<sup>3</sup> και  $A$  η διατομή της δέσμης. Έστω ότι ο στόχος έχει  $n$  σωματίδια ανά μονάδα επιφάνειας. Η ενεργός διατομή της αλληλεπίδρασης είναι  $\sigma = \pi(r_1 + R_2)^2$  όπου  $r_1$  και  $R_2$  οι ακτίνες των σκεδαζομένων σφαιρών και των στόχων αντίστοιχα [βλ. σχήμα (5.8)]: όλες οι σφαίρες έξω από την επιφάνεια αυτή στη δέσμη δεν σκεδάζονται από το συγκεκριμένο στόχο. Η συνολική ενεργός διατομή που παρουσιάζουν όλα τα κέντρα αλληλεπίδρασης του στόχου είναι

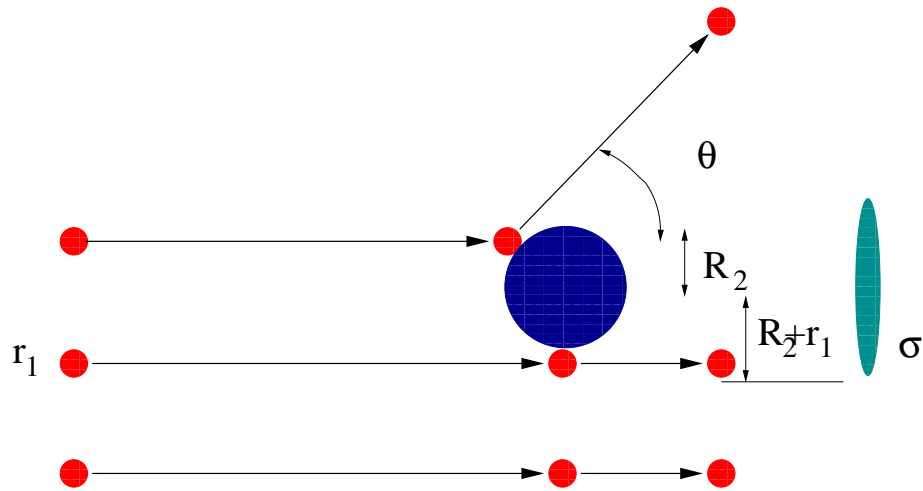
$$\Sigma = nA\sigma, \quad (5.16)$$

όπου  $nA$  είναι ο συνολικός αριθμός των κέντρων του στόχου που βρίσκονται μέσα στην δέσμη. Κατά μέσο όρο, ο ρυθμός σκέδασης, δηλ. ο αριθμός των σκεδάσεων ανά μονάδα χρόνου θα είναι

$$N = J\Sigma = JnA\sigma. \quad (5.17)$$

Η παραπάνω εξίσωση αποτελεί και τον ορισμό της συνολικής ενεργούς διατομής  $\sigma$  της αλληλεπίδρασης για οποιαδήποτε άλλη περίπτωση σκέδασης που πληρεί τις βασικές υποθέσεις που κάναμε. Η ποσότητα αυτή

<sup>3</sup>Ο αριθμός των σωματιδίων που περνούν από μια επιφάνεια κάθετη στη δέσμη ανά μονάδα χρόνου και μονάδα επιφανείας.

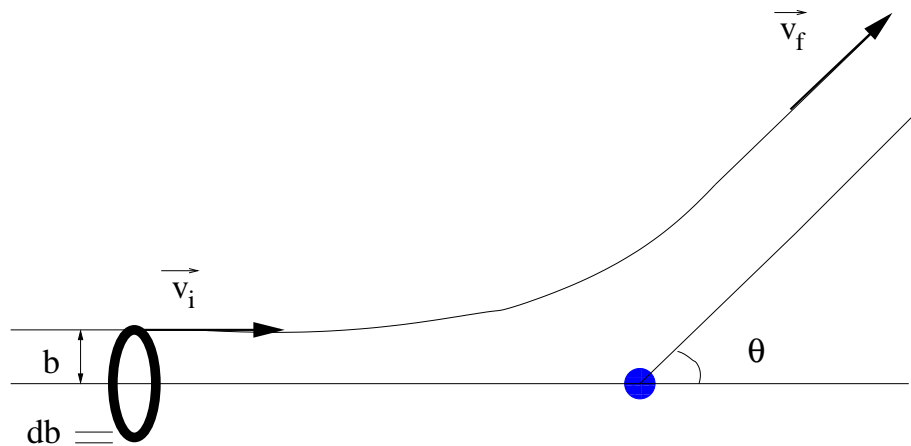


Σχήμα 5.8: Σκέδαση σκληρών σφαιρών.  $\theta$  είναι η γωνία σκέδασης. Δεξιά φαίνεται η συνολική ενεργός διατομή  $\sigma$ .

εξαρτάται από το είδος της αλληλεπίδρασης. Η διαφορική ενεργός διατομή  $\sigma(\theta)$  ορίζεται από τη σχέση

$$dN = JnA\sigma(\theta) d\Omega, \quad (5.18)$$

όπου  $dN$  ο αριθμός των σωματιδίων ανά μονάδα χρόνου που σκεδάζονται μέσα στη στερεά γωνία  $d\Omega$ . Η συνολική ενεργός διατομή είναι



Σχήμα 5.9: Σωματίδια της δέσμης που περνούν μέσα από το δακτύλιο  $2\pi b db$  σκεδάζονται μέσα στη στερεά γωνία  $d\Omega = 2\pi \sin\theta d\theta$ .

$$\sigma_{tot} = \int_{\Omega} \sigma(\theta) d\Omega = \int \sigma(\theta) \sin \theta d\theta d\phi = 2\pi \int \sigma(\theta) \sin \theta d\theta. \quad (5.19)$$

Στην τελευταία σχέση χρησιμοποιήσαμε την κυλινδρική συμμετρία της αλληλεπίδρασης ως προς τον άξονα της κρούσης. Καταλήγουμε στη σχέση

$$\sigma(\theta) = \frac{1}{nAJ} \frac{dN}{2\pi \sin \theta d\theta}. \quad (5.20)$$

Αυτή η σχέση μπορεί να χρησιμοποιηθεί πειραματικά για τη μέτρηση της διαφορικής ενεργούς διατομής μετρώντας το ρυθμό ανίχνευσης σωματιδίων μέσα σε δύο κώνους που ορίζονται από τις γωνίες  $\theta$  και  $\theta + d\theta$ . Τη σχέση αυτή θα χρησιμοποιήσουμε και στον αριθμητικό υπολογισμό της  $\sigma(\theta)$ .

Για να προσδιορίσουμε τη διαφορική ενεργό διατομή από μια θεωρία, μπορούμε να ακολουθήσουμε την εξής γενική διαδικασία. Έστω ότι σωματίο βάλλεται προς τον στόχο όπως φαίνεται στο σχήμα 5.9.  $b$  ονομάζεται η παράμετρος κρούσης και η τελική γωνία  $\theta$  εξαρτάται από αυτή. Άρα το μέρος της δέσμης που σκεδάζεται σε γωνίες μεταξύ  $\theta$  και  $\theta + d\theta$  βρίσκεται σε ένα κυκλικό δαχτυλίδι ακτίνας  $b(\theta)$ , πάχους  $db$  και εμβαδού  $2\pi b db$ . Αφού έχουμε ένα σωματίο στο στόχο  $nA = 1$ . Ο αριθμός των σωματιδίων ανά μονάδα χρόνου που περνούν μέσα από το δαχτυλίδι είναι  $J2\pi b db$ , άρα

$$2\pi b(\theta) db = -2\pi \sigma(\theta) \sin \theta d\theta \quad (5.21)$$

(το  $-$  οφείλεται στο γεγονός ότι, όταν το  $b$  αυξάνει, το  $\theta$  μικραίνει). Από το δυναμικό μπορούμε να υπολογίσουμε το  $b(\theta)$  οπότε προκύπτει η  $\sigma(\theta)$ . Αντίστροφα, αν μετρήσουμε τη  $\sigma(\theta)$ , μπορούμε να προσδιορίσουμε την  $b(\theta)$ .

#### 5.4.1 Σκέδαση Rutherford

Η σκέδαση φορτισμένου σωματιδίου φορτίου  $q$  (“ηλεκτρονίου”) μέσα σε δυναμικό Coulomb πολύ βαρύτερου σημειακού ηλεκτρικού φορτίου  $Q$  (“πυρήνας”) ονομάζεται σκέδαση Rutherford. Στην περίπτωση αυτή το δυναμικό αλληλεπίδρασης είναι

$$V(r) = \frac{1}{4\pi\epsilon_0} \frac{Q}{r}, \quad (5.22)$$

το οποίο προσδίδει επιτάχυνση  $\vec{a}$  στο σωματίδιο ίση με

$$\vec{a} = \frac{qQ}{4\pi\epsilon_0 m r^2} \hat{r} \equiv \alpha \frac{\vec{r}}{r^3}. \quad (5.23)$$

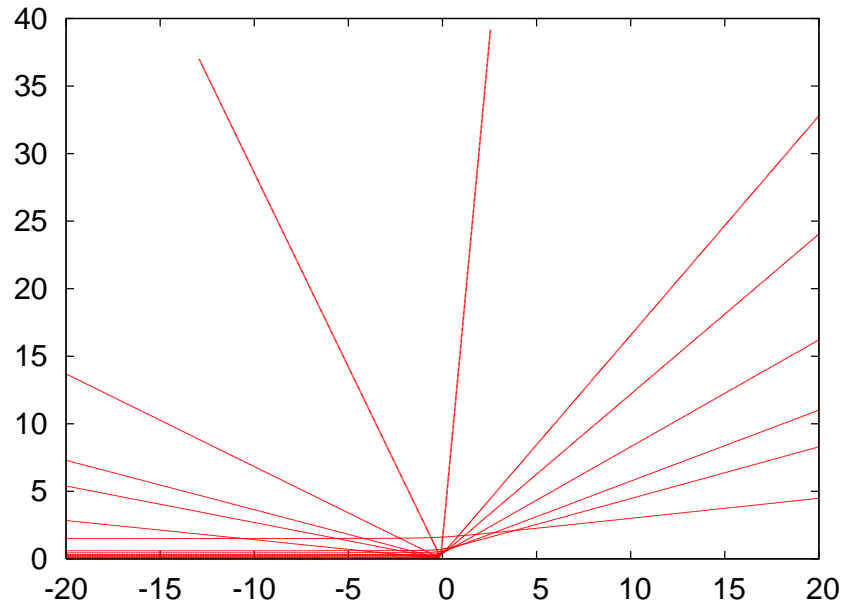
Η ενέργεια του σωματιδίου είναι  $E = \frac{1}{2}mv^2$  και το μέτρο της στροφορμής του είναι  $l = mvb$ , όπου εδώ  $v \equiv |\vec{v}|$ . Η σχέση μεταξύ της παραμέτρου κρούσης και της γωνίας σκέδασης βρίσκεται να είναι [37]

$$b(\theta) = \frac{\alpha}{v^2} \cot \frac{\theta}{2}, \quad (5.24)$$

όπου σε συνδυασμό με την (5.21) προκύπτει ότι

$$\sigma(\theta) = \frac{\alpha^2}{4} \frac{1}{v^4} \sin^{-4} \frac{\theta}{2}. \quad (5.25)$$

Αρχικά εξετάζουμε τις τροχιές σκέδασης. Τα αποτελέσματα φαίνονται



Σχήμα 5.10: Τροχιές σκέδασης Rutherford. Θέσαμε  $k_1 \equiv \frac{qQ}{4\pi\epsilon_0 m} = 1$  στο αρχείο `rk2_cb.f90` και μελετήσαμε τις τροχιές για  $b = 0.08, 0.015, 0.020, 0.035, 0.080, 0.120, 0.200, 0.240, 0.320, 0.450, 0.600, 1.500$ . Το σωματίο τοποθετήθηκε αρχικά στη θέση  $x(0) = -50$  και του δόθηκε αρχική ταχύτητα  $v = 3$ . Ο αριθμός των βημάτων στην ολοκλήρωση είναι 1000 για χρόνο από 0 έως 30.

ποιοτικά στο σχήμα 5.10 στην περίπτωση που τα φορτισμένα σωματρία έχουν ομώνυμα φορτία. Ανάλογο σχήμα προκύπτει και για ετερόνυμα φορτία. Στην περίπτωση αυτή πρέπει να δοθεί ιδιαίτερη προσοχή στην ακρίβεια της μεθόδου για μικρές παραμέτρους κρούσης  $b < 0.2$  (και τις υπόλοιπες παραμέτρους όπως στο σχήμα 5.10) όπου η γωνία σκέδασης γίνεται  $\approx 1$ . Πολύ μεγαλύτερος αριθμός βημάτων απαιτείται για



την επίτευξη ικανοποιητικής ακρίβειας. Βρίσκουμε ότι η ποσότητα που μπορεί να χρησιμοποιηθεί σαν δείκτης που δείχνει την σύγκλιση των αριθμητικών αποτελεσμάτων με αυτά της Σχέσης (5.24) είναι η ενέργεια, η οποία πρέπει να διατηρείται κατά την κρούση. Αυτό θα μας χρησιμεύσει όταν θα μελετήσουμε δυναμικά για τα οποία δεν έχουμε αναλυτική λύση.

Για να μελετήσουμε ποσοτικά τα αποτελέσματά μας αυξάνουμε την ακρίβεια, έτσι ώστε να πετύχουμε ικανοποιητική σύγκλιση των αναλυτικών και αριθμητικών αποτελεσμάτων. Καταρτίζουμε έτσι τον πίνακα 5.2. Θα περιγράψουμε τώρα ένα τρόπο για τον υπολογισμό

$b$	$\theta_n$	$\theta_a$	$\Delta E/E$	Nt
0.008	2.9982	2.9978	$1.06 \cdot 10^{-5}$	5000
0.020	2.7861	2.7854	$6.25 \cdot 10^{-5}$	5000
0.030	2.6152	2.6142	$1.29 \cdot 10^{-4}$	5000
0.043	2.4043	2.4031	$2.31 \cdot 10^{-4}$	5000
0.056	2.2092	2.2079	$3.27 \cdot 10^{-4}$	5000
0.070	2.0184	2.0172	$4.07 \cdot 10^{-4}$	5000
0.089	1.7918	1.7909	$4.70 \cdot 10^{-4}$	5000
0.110	1.5814	1.5808	$4.82 \cdot 10^{-4}$	5000
0.130	1.4147	1.4144	$4.59 \cdot 10^{-4}$	5000
0.160	1.2138	1.2140	$3.97 \cdot 10^{-4}$	5000
0.200	1.0137	1.0142	$3.08 \cdot 10^{-4}$	5000
0.260	0.8070	0.8077	$2.04 \cdot 10^{-4}$	5000
0.360	0.5979	0.5987	$1.07 \cdot 10^{-4}$	5000
0.560	0.3910	0.3917	$3.83 \cdot 10^{-5}$	5000
1.160	0.1905	0.1910	$5.58 \cdot 10^{-6}$	5000

Πίνακας 5.2: Γωνίες σκέδασης στη σκέδαση Rutherford. Θέσαμε  $k1 \equiv \frac{qQ}{4\pi\epsilon_0 m} = 1$  στο αρχείο rk2\_cb.f90 και μελετήσαμε τις τροχιές για τις τιμές του  $b$  που φαίνονται στη στήλη 1.  $\theta_n$  είναι η γωνία σκέδασης που υπολογίζεται αριθμητικά και  $\theta_a$  το αποτέλεσμα της Σχέσης (5.24).  $\Delta E/E$  είναι η ποσοστιαία μεταβολή της ενέργειας λόγω συστηματικών σφαλμάτων της μεθόδου και στην τελευταία στήλη ο αριθμός των βημάτων ολοκλήρωσης για χρόνο από 0 έως 30. Το σωματίδιο τοποθετήθηκε αρχικά στη θέση  $x(0) = -50$  και του δόθηκε αρχική ταχύτητα  $v = 3$ .

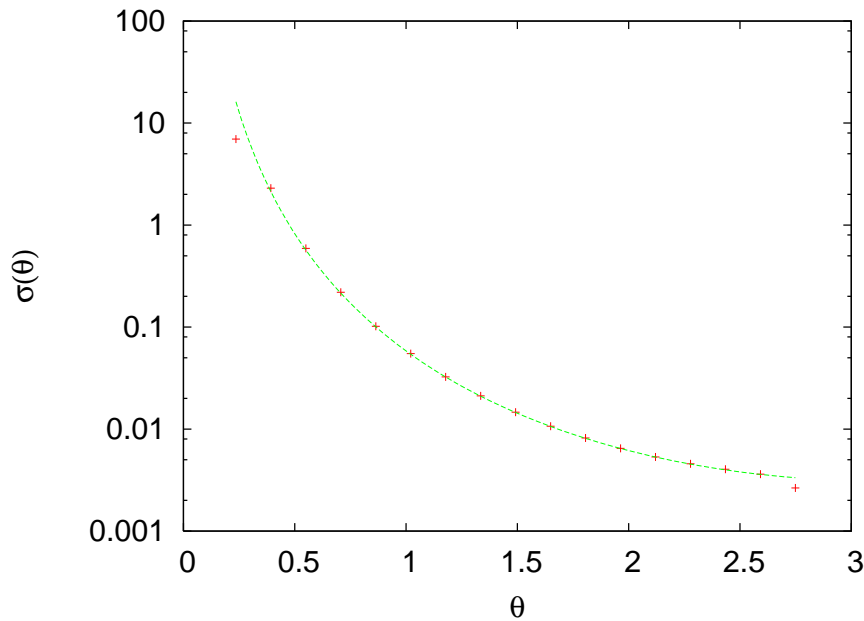
της ενεργούς διατομής χρησιμοποιώντας τη Σχέση (5.20). Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί η (5.21) υπολογίζοντας την κατάλληλη παράγωγο αριθμητικά. Αυτό όμως το αφήνουμε για άσκηση στον ανήσυχο και επιμελή αναγνώστη. Ο υπολογισμός που θα κάνουμε μοιάζει να είναι “πειραματικός”. Τοποθετούμε “ανιχνευτή” που “ανιχνεύει” τα

$b$	$\theta_n$	$\theta_a$	$\Delta E/E$	STEPS
0.020	0.577	2.785	0.47	150000
0.030	2.466	2.614	0.22	150000
0.043	2.333	2.403	$8.62 \cdot 10^{-2}$	150000
0.056	2.180	2.208	$2.88 \cdot 10^{-2}$	150000
0.070	2.006	2.017	$1.07 \cdot 10^{-2}$	150000
0.089	1.779	1.791	$8.94 \cdot 10^{-3}$	60000
0.110	1.570	1.581	$7.07 \cdot 10^{-3}$	30000
0.130	1.406	1.414	$5.25 \cdot 10^{-3}$	20000
0.160	1.198	1.214	$8.63 \cdot 10^{-3}$	5000
0.200	1.007	1.014	$3.71 \cdot 10^{-3}$	5000
0.260	0.8057	0.8077	$1.40 \cdot 10^{-3}$	5000
0.360	0.5988	0.5987	$4.32 \cdot 10^{-4}$	5000
0.560	0.3923	0.3917	$9.40 \cdot 10^{-5}$	5000
1.160	0.1913	0.1910	$8.61 \cdot 10^{-6}$	5000

Πίνακας 5.3: Αποτελέσματα όμοια με αυτά του πίνακα 5.2. Η μόνη διαφορά είναι ότι η σκέδαση είναι μεταξύ ετερόνυμων φορτίων με  $\frac{qQ}{4\pi\epsilon_0 m} = -1$ . Φαίνεται η δυσκολία που αντιμετωπίζει η μέθοδος για μικρές παραμέτρους κρούσης.

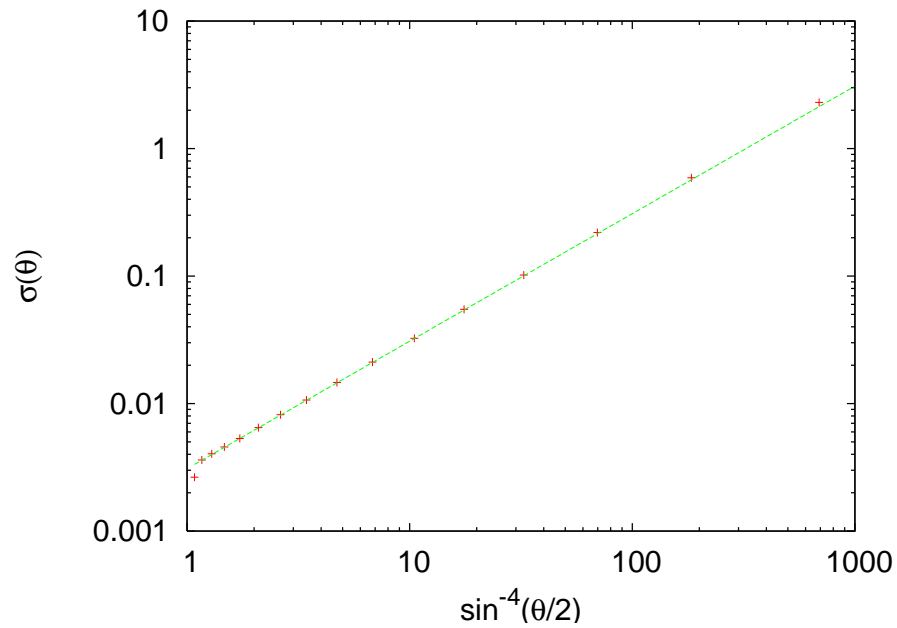
σωμάτια που σκεδάζονται από  $\theta$  μέχρι  $\theta + \delta\theta$ . Για το λόγο αυτό χωρίζουμε το διάστημα  $[0, \pi]$  σε  $N_b$  διαστήματα (bins), έτσι ώστε  $\delta\theta = \pi/N_b$ . Κάνουμε “πειράματα” σκέδασης μεταβάλλοντας την παράμετρο σκέδασης  $b \in [b_m, b_M]$  με βήμα  $\delta b$ . Λόγω της συμμετρίας το προβλήματος, κρατάμε το  $\phi$  σταθερό, οπότε δεδομένο  $\theta$  αντιστοιχεί σε κώνο ανοίγματος  $\theta$  και κορυφή το κέντρο σκέδασης. Παρατηρούμε σε ποια γωνία  $\theta$  σκεδάζεται το σωματίο με το συγκεκριμένο  $b$  και καταχωρούμε τον αριθμό των σωματιδίων ανά μονάδα χρόνου  $\delta N \propto b\delta b$ , επειδή αυτός είναι ανάλογος του εμβαδού του δακτυλιδιού ακτίνας  $b$ . Απομένει να υπολογίσουμε τη ροή  $J$  η οποία είναι ο συνολικός αριθμός των σωματιδίων ανά μονάδα χρόνου που δεν είναι άλλος από  $J \propto \sum_i b\delta b$  (στο λόγο  $\delta N/J$  η σταθερά αναλογίας και το  $\delta b$  απλοποιούνται) και τη στερεά γωνία  $2\pi \sin(\theta) \delta\theta$ . Τέλος, εύκολα χρησιμοποιείται η σχέση (5.19) για τον υπολογισμό της συνολικής ενεργούς διατομής  $\sigma_{tot}$ . Ο προγραμματισμός της διαδικασίας γίνεται μεταβάλλοντας απλά το κυρίως πρόγραμμα του rk2.f90 το οποίο καταγράφουμε στο αρχείο scatter.f90.

```
!=====
!Program that computes scattering cross-section of a central
!force on the plane. The user should first check that the
```



Σχήμα 5.11: Διαφορική ενεργός διατομή για τη σκέδαση Rutherford. Η συνεχής γραμμή είναι η συνάρτηση (5.25) για  $\alpha = 1$ ,  $v = 3$ . Θέσαμε  $\frac{qQ}{4\pi\epsilon_0 m} = 1$ . Το σωματίο τοποθετήθηκε αρχικά στη θέση  $x(0) = -50$  και του δόθηκε αρχική ταχύτητα  $v = 3$ . Γίνανε 5000 βήματα ολοκλήρωσης για χρόνο από 0 έως 30. Η παράμετρος κρούσης  $b$  μεταβλήθηκε από 0.02 έως 1 με βήμα 0.0002.

```
!parameters used, lead to a free state in the end.
! ** X20 is the impact parameter b **
!A 4 ODE system is solved using Runge-Kutta Method
!User must supply derivatives
!dx1/dt=f1(t,x1,x2,x3,x4) dx2/dt=f2(t,x1,x2,x3,x4)
!dx3/dt=f3(t,x1,x2,x3,x4) dx4/dt=f4(t,x1,x2,x3,x4)
!as real(8) functions
!Output is written in file scatter.dat
!=====
program scatter_cross_section
  implicit none
  integer,parameter :: P=1010000
  real(8),dimension(P):: T,X1,X2,V1,V2
  real(8) :: Ti,Tf,X10,X20,V10,V20
  real(8) :: X20F,dX20 !max impact parameter and step
  integer :: Nt
  integer :: i
  real(8) :: k1,k2
  common /couplings/k1,k2
  integer, parameter :: Nbins=20
```



Σχήμα 5.12: Διαφορική ενεργός διατομή για τη σκέδαση Rutherford όπως στο σχήμα 5.11. Η συνεχής ευθεία γραμμή είναι η  $1/(4 \times 3^4)x$  από όπου είναι εμφανής η συναρτησιακή μορφή της  $\sigma(\theta)$ .

```

integer :: index
real(8) :: angle,bins(Nbins),Npart
real(8),parameter :: PI      =3.14159265358979324D0
real(8),parameter :: rad2deg=180.0D0/PI
real(8),parameter :: dangle =PI/Nbins
real(8) R,density,dOmega,sigma,sigmatot
!Input:
print *, 'Runge-Kutta Method for 4-ODEs Integration'
print *, 'Enter coupling constants:'
read  *, k1,k2
print *, 'k1= ',k1, ' k2= ',k2
print *, 'Enter Nt,Ti,Tf,X10,X20,V10,V20:'
read  *, Nt,Ti,TF,X10,X20,V10,V20
print *, 'Enter final impact parameter X20F and step dX20:'
read  *, X20F,dX20
print *, 'Nt = ',Nt
print *, 'Time: Initial Ti =',Ti, ' Final Tf=',Tf
print *, '          X1(Ti)=',X10, ' X2(Ti)=',X20
print *, '          V1(Ti)=',V10, ' V2(Ti)=',V20
print *, 'Impact par X20F =',X20F, ' dX20 =',dX20

open(unit=11,file='scatter.dat')

```

```

bins      = 0.0d0
!The Calculation:
Npart     = 0.0D0
X20       = X20 + dX20/2.0D0 !starts in middle of first interval
do while (X20 .lt. X20F )
  call RK(T,X1,X2,V1,V2,Ti,Tf,X10,X20,V10,V20,Nt)
! Take absolute value due to symmetry:
  angle = DABS(atan2(V2(Nt),V1(Nt)))
!Output: The final angle. Check if almost constant
  write(11,*) '@ ', X20, angle,&
    DABS(atan2(V2(Nt-50),V1(Nt-50))),&
    k1/V10**2/tan(angle/2.0D0)
!Update histogram:
  index    = int(angle/dangle)+1
!Number of incoming particles per unit time
!is proportional to radius of ring
!of radius X20, the impact parameter:
!db is cancelled from density
  bins(index) = bins(index) + X20
  Npart       = Npart       + X20 !<-- i.e. from here
  X20         = X20         + dX20
enddo
!Print scattering cross section:
R           = X20           !beam radius
density     = Npart/(PI*R*R) !beam flux density J
sigmatot    = 0.0D0         !total cross section
do i=1,Nbins
  angle      = (i-0.5D0)*dangle
  d0omega    = 2.0D0*PI*sin(angle)*dangle !d(Solid Angle)
  sigma      = bins(i)/(density*d0omega)
  if(sigma.gt.0.0D0) &
    write(11,*) 'ds= ',angle,angle*rad2deg,sigma
  sigmatot = sigmatot + sigma*d0omega
enddo
write(11,*) 'sigmatot= ',sigmatot
close(11)
end program scatter_cross_section

```

Η μεταγλώττιση γίνεται όπως και με την περίπτωση το rk2.f90, ενώ τα αποτελέσματα βρίσκονται στο αρχείο scatter.dat. Έτσι, για να παράγουμε τα αποτελέσματα των Σχημάτων 5.11 και 5.12 εκτελούμε τις εντολές:

```

> gfortran scatter.f90 rk2_cb.f90 -o scatter
> ./scatter
Runge-Kutta Method for 4-ODEs Integration
Enter coupling constants:

```

```

1.0 0.0
k1= 1.00000 k2= 0.00000
Enter Nt,Ti,Tf,X10,X20,V10,V20:
5000 0 30 -50 0.02 3 0
Enter final impact parameter X20F and step dX20:
1 0.0002
Nt= 5000
Time: Initial T0 = 0.00000 Final TF= 30.00000
      X1(T0)= -50.00000 X2(T0)= 2.00000E-002
      V1(T0)= 3.00000 V2(T0)= 0.00000
Impact par X20F = 1.00000 dX20 = 2.00000E-004

```

και ακολούθως βλέπουμε τα αποτελέσματα με το gnuplot:

```

gnuplot> set log
gnuplot> plot [:1000] "<grep ds= scatter.dat" \
u ((sin($2/2))**(-4)):(($4) notit, \
(1./(4.*3.**4))*x notit
gnuplot> unset log
gnuplot> set log y
gnuplot> plot [:] "<grep ds= scatter.dat" u 2:4 notit, \
(1./(4.*3.**4))*(sin(x/2))**(-4) notit

```

Τα αποτελέσματα που παίρνουμε είναι σε πολύ καλή συμφωνία με τα αναμενόμενα από την αναλυτική έκφραση (5.25). Το επόμενο βήμα θα είναι να μελετήσουμε διαφορετικά δυναμικά για τα οποία δεν έχουμε αναλυτική λύση με την οποία θα μπορούσαμε να συγκρίνουμε τα αποτελέσματά μας.

### 5.4.2 Σκέδαση σε Άλλα Πεδία Δυνάμεων

Ας εξετάσουμε πρώτα τη σκέδαση από μία δύναμη της μορφής

$$\vec{F} = f(r) \hat{r}, \quad f(r) = \begin{cases} \frac{1}{r^2} - \frac{r}{a^3} & r \leq a \\ 0 & r > a \end{cases}, \quad (5.26)$$

η οποία είναι ένα απλό μοντέλο της σκέδασης ποζιτρονίου  $e^+$  (θετικό φορτίο  $+e$ ) με άτομο υδρογόνου που αποτελείται από θετικά φορτισμένο πυρήνα (θετικό φορτίο  $+e$ ) που περιβάλλεται από νέφος ηλεκτρονίου αντίθετου φορτίου. Φυσικά έχουμε θέσει τις κλίμακες, έτσι ώστε  $m_{e^+} = 1$  και  $e^2/4\pi\epsilon_0 = 1$ . Στην περίπτωση αυτή δεν έχουμε αναλυτική λύση, οπότε θα χρησιμοποιήσουμε αριθμητικές μεθόδους για τον υπολογισμό των συναρτήσεων  $b(\theta)$ ,  $\sigma(\theta)$ , καθώς και της συνολικής ενεργού διατομής  $\sigma_{tot}$ .

Η δυναμική ενέργεια δίνεται από τη σχέση:

$$f(r) = -\frac{dV(r)}{dr} \Rightarrow V(r) = \frac{1}{r} + \frac{r^2}{2a^2} - \frac{3}{2a}. \quad (5.27)$$

όπου επιλέξαμε  $V(r) = 0$  για  $r \geq a$ . Ο προγραμματισμός της δύναμης γίνεται εύκολα στο αρχείο `rk_hy.f90`:

```
!=====
!The acceleration functions f3,f4(t,x1,x2,v1,v2) provided
!by the user
!=====
!Motion in hydrogen atom + positron:
!f(r) = 1/r^2-r/k1^3
!ax= f(r)*x1/r ay= f(r)*x2/r
real(8) function f3(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: r2,r,fr
  r2=x1*x1+x2*x2
  r =sqrt(r2)
  if(r .le.k1 .and. r2.gt.0.0D0)then
    fr = 1/r2-r/k1**3
  else
    fr = 0.0D0
  endif

  if(fr.gt.0.0D0 .and. r .gt.0.0D0)then
    f3=fr*x1/r !dx3/dt=dv1/dt=a1
  else
    f3=0.0D0
  endif
end function f3
!=====
real(8) function f4(t,x1,x2,v1,v2)
  implicit none
  real(8) :: t,x1,x2,v1,v2
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: r2,r,fr
  r2=x1*x1+x2*x2
  r =sqrt(r2)
  if(r .le.k1 .and. r2.gt.0.0D0)then
    fr = 1/r2-r/k1**3
  else
    fr = 0.0D0
```

```

endif

if(fr.gt.0.0D0 .and. r .gt.0.0D0)then
  f4=fr*x2/r          !dx3/dt=dv1/dt=a1
else
  f4=0.0D0
endif
end function f4

!
real(8) function energy(t,x1,x2,v1,v2)
implicit none
real(8) :: t,x1,x2,v1,v2
real(8) :: k1,k2
common /couplings/k1,k2
real(8) :: r,Vr
r=sqrt(x1*x1+x2*x2)
if( r .le.k1 .and. r .gt.0.0D0)then
  Vr = 1/r + 0.5D0*r*r/k1**3 - 1.5D0 / k1
else
  Vr = 0.0D0
endif
energy = 0.5D0*(v1*v1+v2*v2) + Vr
end function energy

```

Τα αποτελέσματα δίνονται στα σχήματα 5.13–5.14. Βρίσκουμε ότι  $\sigma_{tot} = \pi a^2$  (Άσκηση 5.10).

Ένα άλλο δυναμικό που παρουσιάζει ενδιαφέρον είναι το δυναμικό Yukawa ως φαινομενολογικό μοντέλο πυρηνικών αλληλεπιδράσεων:

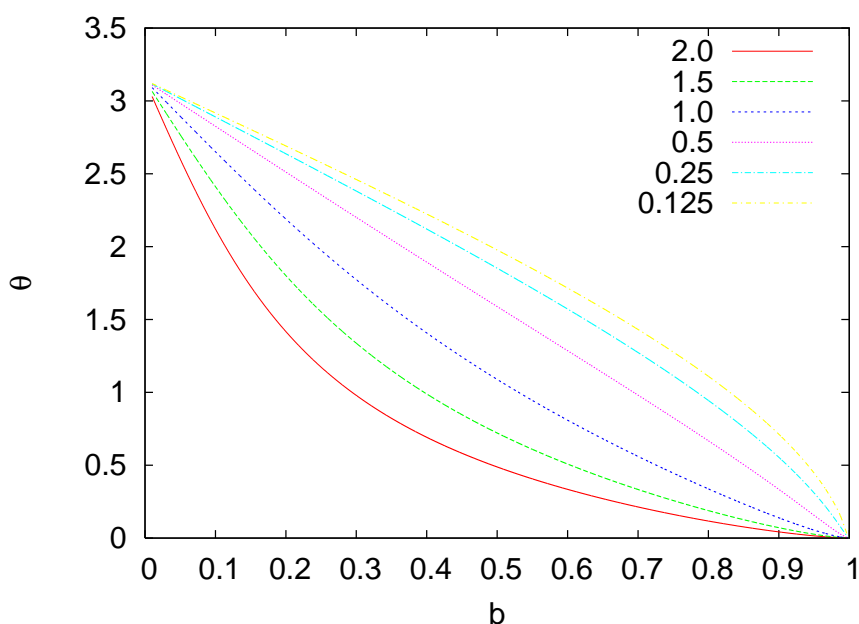
$$V(r) = k \frac{e^{-r/a}}{r}. \quad (5.28)$$

Το πεδίο αυτό μπορεί να χρησιμοποιηθεί και ως μοντέλο ενεργούς αλληλεπίδρασης των ηλεκτρονίων στα μέταλλα (Thomas–Fermi) ή ως το δυναμικό Debye στο κλασικό πλάσμα. Η δύναμη που ασκείται σε ένα σώμα υπό την επίδραση του δυναμικού αυτού είναι:

$$\vec{F}(r) = f(r) \hat{r}, \quad f(r) = k \frac{e^{-r/a}}{r^2} \left(1 + \frac{r}{a}\right) \quad (5.29)$$

Ο προγραμματισμός της δύναμης γίνεται στο αρχείο rk2\_yu.f90 κατά απόλυτη αναλογία με την προηγούμενη περίπτωση. Τα αποτελέσματα φαίνονται στα σχήματα 5.15–5.16.



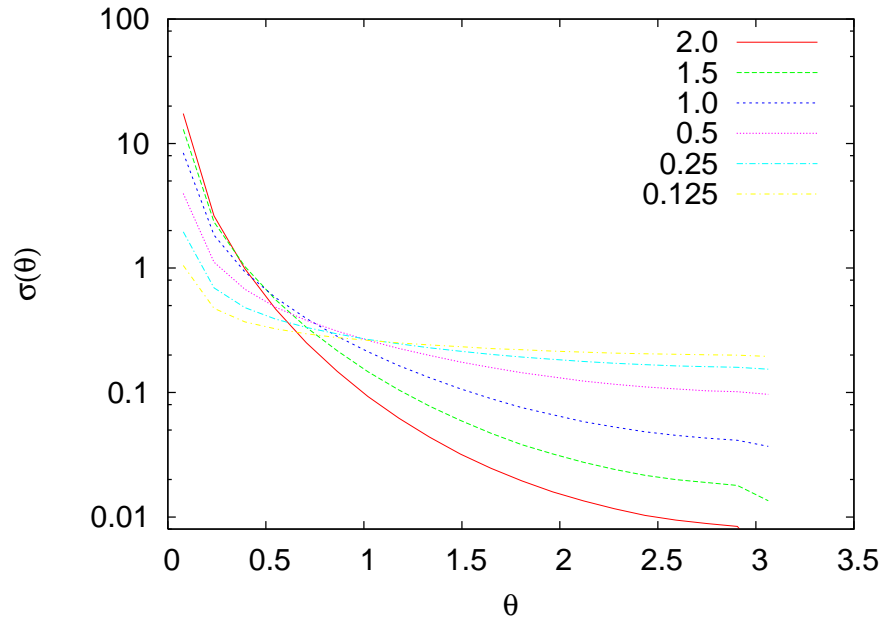


Σχήμα 5.13: Η συνάρτηση  $b(\theta)$  για το δυναμικό της Σχέσης (5.27) για διαφορετικές τιμές της αρχικής ταχύτητας  $v$ . Έχουμε επιλέξει  $a = 1$  και η ολοκλήρωση γίνεται σε 4000 βήματα από  $t_i = 0$  έως  $t_f = 40$  με  $x(0) = -5$ .

## 5.5 Περισσότερα Σωματίδια

Στην παράγραφο αυτή γενικεύουμε τη μελέτη των προηγούμενων παραγράφων προσθέτοντας στο δυναμικό σύστημα που μελετάμε περισσότερους βαθμούς ελευθερίας. Ο αριθμός των δυναμικών εξισώσεων που έχουμε να λύσουμε αυξάνει, οπότε θα γενικεύσουμε το πρόγραμμα ολοκλήρωσης με τη μέθοδο Runge–Kutta 4ης τάξης για αυθαίρετο αριθμό εξισώσεων NEQ. Θα εξηγήσουμε στον αναγνώστη πώς να κάνει *δυναμική εκχώρηση μνήμης* (dynamic memory allocation), έτσι ώστε να καθορίζεται δυναμικά όταν τρέχει το πρόγραμμα, ο χώρος στη μνήμη για τα arrays που είναι απαραίτητα για τις NEQ συναρτήσεις που ολοκληρώνουμε.

Μέχρι τώρα χρησιμοποιούσαμε *στατική εκχώρηση μνήμης* (static memory allocation). Αυτό σημαίνει ότι ο μεταγλωττιστής γνωρίζει τη στιγμή της μεταγλώττισης το μέγεθος των χρησιμοποιούμενων arrays. Για παράδειγμα, στο πρόγραμμα `rk2.f90` ορίζουμε την παράμετρο `P` να έχει μια δεδομένη τιμή, και τα arrays δηλώνονται με τις εντολές:

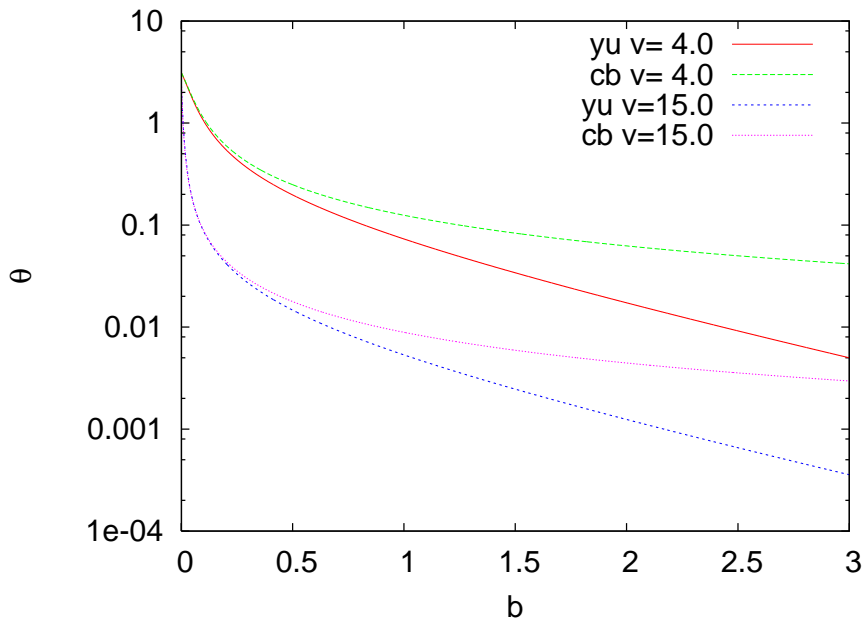


Σχήμα 5.14: Η συνάρτηση  $\sigma(\theta)$  για το δυναμικό της Σχέσης (5.27) για διαφορετικές τιμές της αρχικής ταχύτητας  $v$ . Έχουμε επιλέξει  $a = 1$  και η ολοκλήρωση γίνεται σε 4000 βήματα από  $t_i = 0$  έως  $t_f = 40$  με  $x(0) = -5$ .

```
integer , parameter :: P=1010000
real(8) , dimension(P) :: T, X1, X2, V1, V2
```

Αυτό σημαίνει πως αν θελήσουμε να αλλάξουμε το  $P$ , πρέπει να μεταβάλλουμε την τιμή του στον πηγαίο κώδικα και μετά να ξαναμεταγλωττίσουμε. Με τη δυναμική εκχώρηση της μνήμης, μπορούμε να πάρουμε την τιμή του αριθμού των χρονικών σημείων  $Nt$ , καθώς και τον αριθμό των εξισώσεων  $NEQ$ , όταν τρέχει το πρόγραμμα και να ζητήσουμε από το λειτουργικό σύστημα την απαραίτητη μνήμη, αφού οι τιμές τους γίνουν γνωστές. Για να γίνει αυτό, πρέπει να προσδιορίσουμε το *shape* των arrays (πόσους δείκτες έχουν) και να τους δώσουμε τον προσδιορισμό *allocatable*. Τότε μπορούμε σε οποιαδήποτε στιγμή να ζητήσουμε τη μνήμη που απαιτείται καλώντας τη συνάρτηση `ALLOCATE`. Ένα παράδειγμα φαίνεται παρακάτω:

```
integer Nt, NEQ
real(8) , allocatable :: T (:)      ! Rank-1 array
real(8) , allocatable :: X (:,:)    ! Rank-2 array
real(8) , allocatable :: X0 (:)     ! Rank-1 array
```



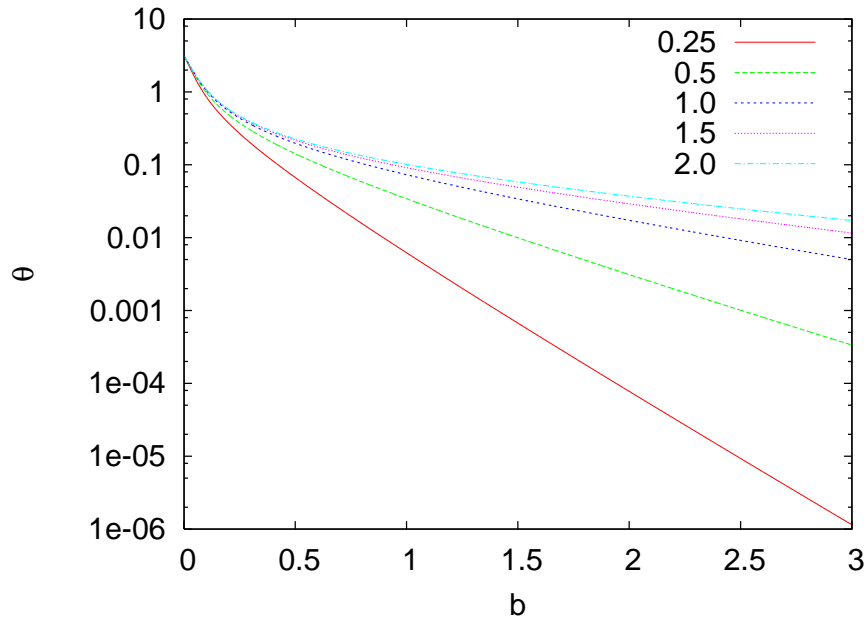
Σχήμα 5.15: Η συνάρτηση  $b(\theta)$  για το δυναμικό Yukawa για διαφορετικές τιμές της αρχικής ταχύτητας  $v$ . Έχουμε επιλέξει  $a = 1$ ,  $k = 1$  και η ολοκλήρωση γίνεται σε 5000 βήματα από  $t_i = 0$  έως  $t_f = 30$  με  $x(0) = -50$ . Δίνεται συγκριτικά η σχέση (5.24) της σκέδασης Rutherford από τις καμπύλες μαρκαρισμένες ως cb.

```

read *,Nt
call finit(NEQ)
allocate(X0(NEQ))
allocate(T(Nt))
allocate(X(Nt,NEQ))
...
(compute with X0,T,X)
...
deallocate(X0)
deallocate(X )
deallocate(T )
(X0,T,X are not usable anymore)
...
.....
subroutine finit(NEQ)
  NEQ = 4
end subroutine finit

```

Στον παραπάνω κώδικα, δηλώνουμε τα arrays να έχουν την ιδιότητα allocatable και για κάθε “:” έχουμε ένα δείκτη. Έτσι τα arrays T,X0 έχουν ένα δείκτη (rank-1 arrays) και το X έχει δύο δείκτες (rank-2



Σχήμα 5.16: Η συνάρτηση  $b(\theta)$  για το δυναμικό Yukawa για διαφορετικές τιμές της “έκτασης”  $a$  της δύναμης. Έχουμε επιλέξει  $v = 4.0$ ,  $k = 1$  και η ολοκλήρωση γίνεται σε 5000 βήματα από  $t_i = 0$  έως  $t_f = 30$  με  $x(0) = -50$ .

array). Στη συνέχεια, ζητάμε τις τιμές για τις μεταβλητές  $Nt$ ,  $NEQ$  από το χρήστη και από την υπορουτίνα `finit` και καλούμε τη συνάρτηση `ALLOCATE` για να ζητήσει από το λειτουργικό σύστημα την εκχώρηση της ζητούμενης μνήμης για τα στοιχεία των arrays<sup>4</sup>. Αν η εκχώρηση μνήμης γίνει με επιτυχία, μπορούμε να χρησιμοποιήσουμε τα arrays όπως κάναμε μέχρι τώρα. Όταν δε χρειαζόμαστε πια τα arrays, επιστρέφουμε τη μνήμη που ζητήσαμε στο σύστημα με την συνάρτηση `DEALLOCATE`. Αν το αμελήσουμε και ζητάμε μνήμη ανεξέλεγκτα χωρίς να την επιστρέφουμε, όταν αυτή δεν είναι πια αναγκαία (λ.χ. σε μια επαναλαμβανόμενη διαδικασία), το πρόγραμμά μας μπορεί να παρουσιάσει αυτό που λέγεται “διαρροές μνήμης” (memory leaks). Η δυναμική εκχώρηση μνήμης είναι πολύ βολική, αλλά θα πρέπει να σημειώσουμε ότι σε υπολογισμούς υψηλών επιδόσεων (high performance computing) η στατική εκχώρηση μνήμης μπορεί να οδηγήσει σε προγράμματα που τρέχουν πιο γρήγορα.

Μετά από αυτό, οι αλλαγές που πρέπει να κάνουμε στο πρόγραμμα

<sup>4</sup>Υποθέτουμε πως τα  $Nt$ ,  $NEQ$  έχουν θετικές τιμές και η μνήμη που ζητάμε είναι διαθέσιμη. Πιο σωστά θα έπρεπε να δώσουμε την εντολή `allocate(T(Nt), STAT=IERR)` και μετά να ελέγξουμε την εκχώρηση με την εντολή `IF(IERR .eq. 0) STOP 'Memory allocation for T failed'`.

δεν είναι σημαντικές. Θα γράψουμε το κυρίως πρόγραμμα στο αρχείο `rkA.f90`, ενώ θα κωδικοποιήσουμε τις δυναμικές εξισώσεις σε συνοδευτικό αρχείο με όνομα της μορφής `rkA_XXX.f90`. Στο τελευταίο, ο χρήστης θα πρέπει να προγραμματίσει μια υπορουτίνα  $f(t, X, dXdt)$  η οποία θα παίρνει στην είσοδο τη χρονική στιγμή  $t$  και τις τιμές των υπό ολοκλήρωση συναρτήσεων  $X(NEQ)$  και θα δίνει στην έξοδο τις τιμές των παραγώγων τους  $dXdt(NEQ)$  τη χρονική στιγμή  $t$ . Επίσης, η συνάρτηση `finit(NEQ)` θα θέτει τον αριθμό των εξισώσεων της  $f$  και η οποία θα καλείται μία φορά στην αρχή του κυρίως προγράμματος.

Το κυρίως πρόγραμμα, που βρίσκεται στο αρχείο `rkA.f90`<sup>5</sup> είναι:

```
!=====
!Program to solve an ODE system using the
!4th order Runge-Kutta Method
!NEQ: Number of equations
!User supplies two subroutines:
!f(t,x,xdot): with real(8) :: t,x(NEQ),xdot(NEQ) which
!given the time t and current values of functions x(NEQ)
!it returns the values of derivatives: xdot = dx/dt
!The values of two coupling constants k1,k2 may be used
!in f which are read in the main program and stored in
!common /couplings/k1,k2
!finit(NEQ) : sets the value of NEQ
!
!User Interface:
!k1,k2: real(8) coupling constants
!Nt,Ti,Tf: Nt-1 integration steps, initial/final time
!X0: real(8),dimension(NEQ): initial conditions
!Output:
!rkA.dat with Nt lines consisting of: T(Nt),X(Nt,NEQ)
!=====
program rk2_solve
  implicit none
  real(8),allocatable :: T (:)
  real(8),allocatable :: X (:,:)
  real(8),allocatable :: X0(:)
  real(8) :: Ti,Tf
  integer :: Nt, NEQ,i
  real(8) :: k1,k2
  common /couplings/k1,k2
  !We need explicit interface, since energy has
  !assumed-shape arrays as arguments.
  INTERFACE
```

<sup>5</sup>Στο συνοδευτικό λογισμικό θα βρείτε και τα αρχεία `rkN.f90` και `rkN_XXX.f90` που δείχνουν πώς μπορεί να γίνει ο προγραμματισμός με στατική εκχώρηση μνήμης.

```

real(8) function energy(t_intrf,x_intrf)
  implicit none
  real(8) :: t_intrf,x_intrf(:)
end function energy
END INTERFACE

!Input:
print *, 'Runge-Kutta Method for ODE Integration.'
!Get the number of equations:
call finit(NEQ);allocate(X0(NEQ))
print *, 'NEQ= ',NEQ
print *, 'Enter coupling constants:'
read *, k1,k2
print *, 'k1= ',k1, ' k2= ',k2
print *, 'Enter Nt,Ti,Tf,X0:'
read *, Nt,Ti,TF,X0
print *, 'Nt = ',Nt
print *, 'Time: Initial Ti =',Ti, ' Final Tf=',Tf
print '(A,2000G28.16)', ' X0 =',X0
allocate(T(Nt));allocate(X(Nt,NEQ))
!The Calculation:
call RK(T,X,Ti,Tf,X0,Nt,NEQ)
!Output:
open(unit=11,file='rkA.dat')
do i=1,Nt
  write(11,'(2000G28.16)')T(i),X(i,:),&
    energy(T(i),X(i,:))
enddo
close(11)
end program rk2_solve

!=====
!Driver of the RKSTEP routine
!=====

subroutine RK(T,X,Ti,Tf,X0,Nt,NEQ)
  implicit none
  integer :: Nt,NEQ
  real(8),dimension(Nt) :: T
  real(8),dimension(Nt,NEQ):: X
  real(8),dimension(NEQ) :: X0
  real(8) :: Ti ,Tf
  real(8) :: dt
  real(8) :: TS,XS(NEQ) !values of time and X at given step
  integer :: i
!Initialize variables:
dt = (Tf-Ti)/(Nt-1)
T (1) = Ti
X (1,:)= X0
TS = Ti
XS = X0
!Make RK steps: The arguments of RKSTEP are

```

```

!replaced with the new ones
do i=2,Nt
  call RKSTEP(TS,XS,dt,NEQ)
  T(i) = TS
  X(i,:)= XS
enddo
end subroutine RK

!=====
!Subroutine RKSTEP(t,X,dt)
!Runge-Kutta Integration routine of ODE
!=====
subroutine RKSTEP(t,x,dt,NEQ)
  implicit none
  integer :: NEQ
  real(8),dimension(NEQ) :: x
  real(8) :: t,dt,tt
  real(8),dimension(NEQ) :: k1,k2,k3,k4,xx
  real(8) :: h,h2,h6
!We need explicit interface, since f has assumed-shape
!arrays as arguments.
  INTERFACE
    subroutine f(t_intrf,x_intrf,xdot_intrf)
      implicit none
      real(8) :: t_intrf
      real(8),dimension(:) :: x_intrf,xdot_intrf
    end subroutine f
  END INTERFACE

  h =dt          !h =dt, integration step
  h2=0.5D0*h     !h2=h/2
  h6=h/6.0D0    !h6=h/6

  call f(t ,x ,k1); xx = x + h2*k1; tt =t+h2
  call f(tt,xx,k2); xx = x + h2*k2; tt =t+h2
  call f(tt,xx,k3); xx = x + h *k3; tt =t+h
  call f(tt,xx,k4)

  t =t+h
  x =x +h6*(k1+2.0D0*(k2+k3)+k4)
end subroutine RKSTEP

```

Να επεξηγήσουμε μερικές λεπτομέρειες. Παρατηρήστε τη χρήση sections των arrays:

```

write(11, '(2000G28.16)')T(i),X(i,:)
X(1,:)= X0
X(i,:)= XS

```

Παραπάνω  $X(1,:)$  είναι ολόκληρη η πρώτη γραμμή του array  $X$ , η οποία είναι *conformable* με το array  $X0$ , και η οποία τίθεται στοιχείο προς στοιχείο ίση με το array  $X0$ . Δηλαδή  $X(1,1)=X0(1)$ ,  $X(1,2)=X0(2)$ ,  $\dots$ ,  $X(1,NEQ)=X0(NEQ)$ . Παρομοίως, η εντολή `write(...)`  $X(i,:)$  τυπώνει ολόκληρη τη γραμμή  $i$  του  $X$ , ενώ η εντολή  $X(i,:)=XS$  θέτει  $X(i,1)=XS(1)$ ,  $X(i,2)=XS(2)$ ,  $\dots$ ,  $X(i,NEQ)=XS(NEQ)$ .

Επίσης παρατηρήστε τις διανυσματικές πράξεις

```
xx = x + h2* k1
x  = x + h6*(k1+2.0D0*(k2+k3)+k4)
```

που είναι ισοδύναμες με τα do loops

```
do i=1,NEQ
  xx(i) = x(i) + h2* k1(i)
enddo
do i=1,NEQ
  x(i)  = x(i) + h6*(k1(i)+2.0D0*(k2(i)+k3(i))+k4(i))
enddo
```

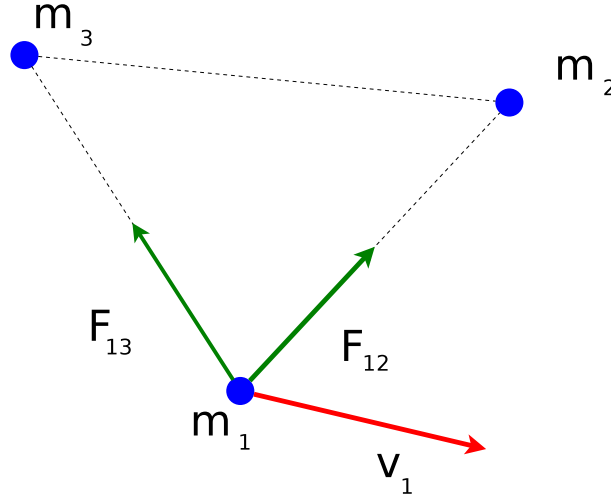
Τέλος, θα εξηγήσουμε την έννοια του INTERFACE block. Τις συναρτήσεις μέχρι τώρα τις δηλώνουμε στο καλούν πρόγραμμα, δηλώνοντας απλά τον τύπο της τιμής που επέστρεφαν. Σε μερικές περιπτώσεις, όπως εδώ όπου έχουν ως ορίσματα (dummy arguments) arrays που δεν είναι γνωστό παρά μόνο το shape τους (assumed-shape arrays), ο μεταγλωττιστής χρειάζεται περισσότερη πληροφορία. Πρέπει να δώσουμε τα ορίσματα, τους τύπους τους καθώς και το shape τους αν είναι arrays. Έτσι, σε κάθε πρόγραμμα που καλεί τις συναρτήσεις αυτές, πρέπει να βάλουμε ένα INTERFACE block που να δίνει τις παραπάνω πληροφορίες. Εδώ, για τις συναρτήσεις `f` και `energy` το INTERFACE block είναι

```
INTERFACE
!-----
  subroutine f(t_intrf,x_intrf,xdot_intrf)
    implicit none
    real(8) :: t_intrf
    real(8),dimension(:) :: x_intrf,xdot_intrf
  end subroutine f
!-----
  real(8) function energy(t_intrf,x_intrf)
    implicit none
    real(8) :: t_intrf,x_intrf(:)
  end function energy
```



```
!
END INTERFACE
```

Μπορείτε να δημιουργήσετε αρχεία με ονόματα όπως λ.χ. `interfaces.inc` που να έχει ομάδες από `INTERFACE` blocks και να το εισάγετε σε κάθε ρουτίνα που τις χρησιμοποιεί με την εντολή `include "interfaces.inc"`.



Σχήμα 5.17: Τρία σωματίδια ίδιας μάζας κινούνται υπό την επίδραση της βαρυτικής δύναμης που ασκεί το ένα στο άλλο (ή ίδιου φορτίου με την επίδραση απωστικής ηλεκτροστατικής δύναμης). Το πρόβλημα λύνεται αριθμητικά στο πρόγραμμα που βρίσκεται στα αρχεία `rkA.f90`, `rkA_3pcb.f90`.

Προχωράμε τώρα να λύσουμε ένα συγκεκριμένο πρόβλημα. Θεωρήστε ότι τρία σωματίδια ίδιας μάζας κινούνται υπό την επίδραση της βαρυτικής δύναμης που ασκεί το ένα στο άλλο (ή ίδιου φορτίου με την επίδραση απωστικής ηλεκτροστατικής δύναμης) όπως φαίνεται στο σχήμα 5.17. Η δύναμεις που ασκούν το ένα στο άλλο είναι

$$\vec{F}_{ij} = \frac{mk_1}{r_{ij}^3} \vec{r}_{ij}, \quad i, j = 1, 2, 3 \quad (5.30)$$

όπου  $k_1 = -Gm$  και οι εξισώσεις κίνησης γίνονται ( $i = 1, 2, 3$ )

$$\begin{aligned} \frac{dx_i}{dt} &= v_{ix} & \frac{dv_{ix}}{dt} &= k_1 \sum_{j=1, j \neq i}^3 \frac{x_i - x_j}{r_{ij}^3} \\ \frac{dy_i}{dt} &= v_{iy} & \frac{dv_{iy}}{dt} &= k_1 \sum_{j=1, j \neq i}^3 \frac{y_i - y_j}{r_{ij}^3}, \end{aligned} \quad (5.31)$$

με  $r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$ . Η συνολική ενέργεια του συστήματος είναι

$$E/m = \frac{1}{2}(v_1^2 + v_2^2) + \sum_{i,j=1,j<i}^3 \frac{k_1}{r_{ij}}. \quad (5.32)$$

Προγραμματίζουμε τις παραπάνω σχέσεις στο αρχείο `rkA_3pcb.f90`, τα περιεχόμενα του οποίου δίνονται παρακάτω:

```
!=====
!Sets number of equations
!=====
subroutine finit(NEQ)
  NEQ = 12
end subroutine finit
!=====
!Three particles of the same
!mass on the plane interacting
!via Coulombic force
!=====
subroutine f(t,X,dXdt)
  implicit none
  real(8) :: k1,k2
  common /couplings/k1,k2
  real(8) :: t,X(:),dXdt(:)
!-----
  real(8) :: x11,x12,x21,x22,x31,x32
  real(8) :: v11,v12,v21,v22,v31,v32
  real(8) :: r12,r13,r23
!-----
  x11 = X(1);x21 = X(5);x31 = X(9)
  x12 = X(2);x22 = X(6);x32 = X(10)
  v11 = X(3);v21 = X(7);v31 = X(11)
  v12 = X(4);v22 = X(8);v32 = X(12)
!-----
  r12 = ((x11-x21)*(x11-x21)+(x12-x22)*(x12-x22))**(-3.D0/2.D0)
  r13 = ((x11-x31)*(x11-x31)+(x12-x32)*(x12-x32))**(-3.D0/2.D0)
  r23 = ((x21-x31)*(x21-x31)+(x22-x32)*(x22-x32))**(-3.D0/2.D0)
!-----
  dXdt(1) = v11
  dXdt(2) = v12
  dXdt(3) = k1*(x11-x21)*r12+k1*(x11-x31)*r13 ! a11=dv11/dt
  dXdt(4) = k1*(x12-x22)*r12+k1*(x12-x32)*r13 ! a12=dv12/dt
!-----
  dXdt(5) = v21
  dXdt(6) = v22
  dXdt(7) = k1*(x21-x11)*r12+k1*(x21-x31)*r23 ! a21=dv21/dt
  dXdt(8) = k1*(x22-x12)*r12+k1*(x22-x32)*r23 ! a22=dv22/dt
```

```

!
dXdt(9) = v31
dXdt(10) = v32
dXdt(11) = k1*(x31-x11)*r13+k1*(x31-x21)*r23 ! a31=dv31/dt
dXdt(12) = k1*(x32-x12)*r13+k1*(x32-x22)*r23 ! a32=dv32/dt
end subroutine f
=====
real(8) function energy(t,X)
implicit none
real(8) :: k1,k2
common /couplings/k1,k2
real(8) :: t,X(:)
!
real(8) :: x11,x12,x21,x22,x31,x32
real(8) :: v11,v12,v21,v22,v31,v32
real(8) :: r12,r13,r23
!
x11 = X(1);x21 = X(5);x31 = X(9)
x12 = X(2);x22 = X(6);x32 = X(10)
v11 = X(3);v21 = X(7);v31 = X(11)
v12 = X(4);v22 = X(8);v32 = X(12)
!
r12 = ((x11-x21)*(x11-x21)+(x12-x22)*(x12-x22))*(-0.5D0)
r13 = ((x11-x31)*(x11-x31)+(x12-x32)*(x12-x32))*(-0.5D0)
r23 = ((x21-x31)*(x21-x31)+(x22-x32)*(x22-x32))*(-0.5D0)
!
energy = &
0.5D0*(v11*v11+v12*v12+v21*v21+v22*v22+v31*v31+v32*v32)
energy = energy + k1*(r12+r13+r23)
end function energy

```

Για να το τρέξετε και για να δείτε τα αποτελέσματα μελετήστε τις εντολές στο σενάριο φλοιού rkA\_3pcb.csh. Ενδεικτικά εκτελέστε την εντολή

```
> rkA_3pcb.csh -0.5 4000 1.5 -1 0.1 1 0 1 -0.1 -1 0 0.05 1 0 -1
```

που θα τρέξει το πρόγραμμα για  $k_1 = -0.5$ ,  $\vec{r}_1(0) = -\hat{x} + 0.1\hat{y}$ ,  $\vec{v}_1(0) = \hat{x}$ ,  $\vec{r}_2(0) = \hat{x} - 0.1\hat{y}$ ,  $\vec{v}_2(0) = -\hat{x}$ ,  $\vec{r}_3(0) = 0.05\hat{x} + \hat{y}$ ,  $\vec{v}_3(0) = -\hat{y}$ ,  $Nt = 4000$  και  $t_f = 1.5$ .

## 5.6 Ασκήσεις

- 5.1 Αναπαράγετε τα αποτελέσματα των Σχημάτων 5.3 και 5.4. Συγκρίνετε τα αποτελέσματά σας με την γνωστή αναλυτική λύση.
- 5.2 Προγραμματίστε τη δύναμη που αισθάνεται φορτισμένο σωματίδιο σε ομογενές μαγνητικό πεδίο  $\vec{B} = B\hat{k}$  και μελετήστε την τροχιά του για  $\vec{v}(0) = v_{0x}\hat{x} + v_{0y}\hat{y}$ . Για  $x(0) = 1, y(0) = 0, v_{0y} = 0$  υπολογίστε την ακτίνα της τροχιάς και φτιάξτε γράφημα της σχέσης ακτίνας τροχιάς και  $v_{0x}$ . Συγκρίνετε με αυτό που αναμένετε από τον αναλυτικό υπολογισμό. (Μη σχετικιστικός υπολογισμός)
- 5.3 Μελετήστε τον ανομοιογενή αρμονικό ταλαντωτή  $a_x = -\omega_1^2 x, a_y = -\omega_2^2 y$ . Αναπαράγετε τις καμπύλες Lissajous παίρνοντας  $x(0) = 0, y(0) = 1, v_x(0) = 1, v_y(0) = 0, t_f = 2\pi, \omega_2^2 = 1, \omega_1^2 = 1, 2, 4, 9, 16, \dots$ . Τι γίνεται όταν  $\omega_1^2 \neq n\omega_2^2$ ;
- 5.4 Αναπαράγετε τα αποτελέσματα του πίνακα 5.1 και των Σχημάτων 5.5 και 5.6. Φτιάξτε διάγραμμα  $(\ln a, \ln T)$  και υπολογίστε την κλίση της ευθείας που θα προκύψει με τη μέθοδο των ελαχίστων τετραγώνων. Είναι αυτή που περιμένετε; Υπολογίστε το σημείο τομής με τον κατακόρυφο άξονα και συγκρίνετε το αποτέλεσμα σας με το αναμενόμενο.
- 5.5 Υπολογίστε τη στροφορμή ως προς το κέντρο της δύναμης σε κάθε βήμα ολοκλήρωσης στην πλανητική κίνηση και μελετήστε αν πράγματι διατηρείται. Δείξτε αναλυτικά, ότι η διατήρηση της στροφορμής συνεπάγεται ότι το διάνυσμα θέσης του πλανήτη σαρώνει επιφάνεια με σταθερό ρυθμό.
- 5.6 Υπολογίστε αριθμητικά την ταχύτητα διαφυγής  $v_e$  για  $GM = 10.0, y(0) = 0.0, x_0 = x(0) = 1$  με τον ακόλουθο τρόπο. Δείξτε ότι  $v_0^2 = -GM(1/a) + v_e^2$ . Πάρτε  $v_x(0) = 0, v_y(0) = v_0$ . Μεταβάλετε την  $v_y(0) = v_0$  και μετρήστε τον αντίστοιχο  $a$ . Από το σημείο που τέμνει τον άξονα των  $y$  υπολογίστε την  $v_e$ .
- 5.7 Επαναλάβετε την προηγούμενη άσκηση για  $x_0 = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0$ . Από τη γραφική παράσταση  $v_e = f(1/x_0)$  επιβεβαιώστε τη Σχέση (5.14).
- 5.8 Εξετάστε αν για την κλειστή τροχιά του πλανήτη με  $GM = 10.0, x(0) = 1, y(0) = 0.0, v_x(0) = 0, v_y(0) = 4$  ισχύει η οριζουσα ιδιότητα της έλλειψης  $F_1P + F_2P = 2a$ . Ως σημείο  $F_1$  θα πάρετε το κέντρο της

δύναμης και αφού προσδιορίσετε αριθμητικά το  $a$  θα πάρετε ως  $F_2$  το σημείο που είναι συμμετρικό ως προς το κέντρο της έλλειψης.

- 5.9 Θεωρήστε την κίνηση πλανητών σύμφωνα με την προηγούμενη άσκηση. Εφαρμόστε στιγμιαία ώθηση στην εφαπτόμενη διεύθυνση της τροχιάς, αφού ο πλανήτης εκτελέσει περίπου  $1/4$  της τροχιάς του. Πόσο ευσταθής είναι η τροχιά στην ώθηση (δηλ. ποια είναι η εξάρτηση της τροχιάς από το μέγεθος/διάρκεια της ώθησης); Επαναλάβετε την ανάλυση όταν η ώθηση είναι στην κάθετη διεύθυνση.
- 5.10 Θεωρήστε το δυναμικό σκέδασης ποζιτρονίου-υδρογόνου της σχέσης (5.26). Φτιάξτε τη γραφική παράσταση της συνάρτησης  $f(r)$  καθώς και της  $V(r)$  για διαφορετικές τιμές του  $a$ . Υπολογίστε αριθμητικά τη συνολική ενεργό διατομή  $\sigma_{tot}$  και δείξτε ότι είναι ίση με  $\pi a^2$ .
- 5.11 Θεωρήστε το δυναμικό Morse που χρησιμοποιείται σε μοντέλα διατομικών μορίων:

$$V(r) = D (\exp(-2\alpha r) - 2 \exp(-\alpha r)) \quad (5.33)$$

με  $D, \alpha > 0$ . Λύστε αριθμητικά το πρόβλημα αρχικά στη μία διάσταση και συγκρίνετε με τις γνωστές αναλυτικές λύσεις για ενέργεια  $E < 0$ :

$$x(t) = \frac{1}{\alpha} \ln \left\{ \frac{D - \sqrt{D(D - |E|)} \sin(\alpha t \sqrt{2|E|/m} + C)}{|E|} \right\} \quad (5.34)$$

με τη σταθερά ολοκλήρωσης να δίνεται ως συνάρτηση της αρχικής θέσης και της ενέργειας από

$$C = \sin^{-1} \left[ \frac{D - |E|e^{\alpha x_0}}{\sqrt{D(D - |E|)}} \right]. \quad (5.35)$$

Η κίνηση είναι περιοδική με περίοδο που εξαρτάται από την ενέργεια  $= (\pi/\alpha)\sqrt{2m/|E|}$ . Για  $E > 0$  έχουμε

$$x(t) = \frac{1}{\alpha} \ln \left\{ \frac{\sqrt{D(D + E)} \cosh(\alpha t \sqrt{2E/m} + C) - D}{|E|} \right\} \quad (5.36)$$

ενώ για  $E = 0$

$$x(t) = \frac{1}{\alpha} \ln \left\{ \frac{1}{2} + \frac{D\alpha^2}{m}(t + C)^2 \right\}. \quad (5.37)$$

Στις τελευταίες σχέσεις η σταθερά ολοκλήρωσης  $C$  δίνεται από άλλη σχέση και όχι από την (5.35). Μελετήστε την κίνηση στο χώρο των φάσεων  $(x, \dot{x})$  και μελετήστε τη μετάβαση από ανοιχτές σε κλειστές τροχιές του συστήματος.

- 5.12 Στην προηγούμενη άσκηση θεωρήστε τον όρο του ενεργού δυναμικού  $V_{eff}(r) = l^2/2mr^2$  ( $l \equiv |\vec{L}|$ ). Κάνετε τη γραφική παράσταση της συνάρτησης  $V_{tot}(r) = V(r) + V_{eff}(r)$  για  $D = 20$ ,  $\alpha = 1$ ,  $m = 1$ ,  $l = 1$ , φυσικά για  $r > 0$ . Προσδιορίστε τη θέση ισορροπίας και την ενέργεια ιονισμού.

Μελετήστε αριθμητικά τις λύσεις  $x(t)$ ,  $y(t)$ ,  $y(x)$ ,  $r(t)$  στο επίπεδο για  $E > 0$ ,  $E = 0$ , και  $E < 0$ . Στην τελευταία περίπτωση μελετήστε και το πρόβλημα σκέδασης, υπολογίζοντας αριθμητικά τη συνάρτηση  $b(\theta)$ ,  $\sigma(\theta)$  και τη συνολική ενεργό διατομή  $\sigma_{tot}$ .

- 5.13 Θεωρήστε τη δύναμη  $\vec{F}(r) = f(r) \hat{r}$  όπου  $f(r) = 24(2/r^{13} - 1/r^7)$  η οποία είναι μοντέλο μοριακού δυναμικού. Υπολογίστε το δυναμικό  $V(r)$  και κάνετε τη γραφική παράσταση της συνάρτησης  $V_{tot}(r) = V(r) + V_{eff}(r)$ . Προσδιορίστε τη θέση ισορροπίας και την ενέργεια ιονισμού.

Μελετήστε το πρόβλημα σκέδασης, υπολογίζοντας αριθμητικά τη συνάρτηση  $b(\theta)$ ,  $\sigma(\theta)$  και τη συνολική ενεργό διατομή  $\sigma_{tot}$ . Πόσο εξαρτάται ο υπολογισμός σας από την ελάχιστη γωνία σκέδασης;

- 5.14 Μελετήστε την κίνηση σωματιδίου υπό την επίδραση ελκτικής κεντρικής δύναμης  $\vec{F} = -k/r^3 \hat{r}$ . Εξετάστε με ποιες αρχικές συνθήκες παίρνετε σπειροειδή τροχιά.

- 5.15 Υπολογίστε τη συνολική διαφορική διατομή  $\sigma_{tot}$  αναλυτικά και υπολογιστικά για την σκέδαση Rutherford. Τι παρατηρείτε στην αριθμητικά υπολογισμένη τιμή, καθώς μεταβάλετε τα όρια της ολοκλήρωσης;

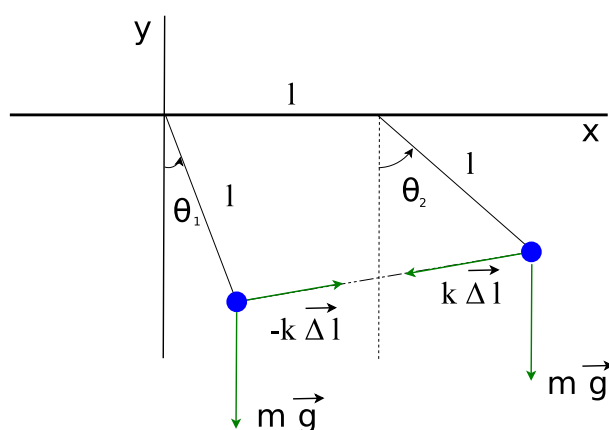
- 5.16 Γράψτε πρόγραμμα που θα υπολογίζει την τροχιά φορτισμένου σωματιδίου, όταν αυτό κινείται σε ηλεκτρικό πεδίο Coulomb που δημιουργείται από  $N$  ακίνητα σημειακά ηλεκτρικά φορτία.

- 5.17 Λύστε το πρόβλημα των τριών σωμάτων που λύνεται στο κείμενο στο αρχείο rkA\_3cb.f90 στην περίπτωση που έχουμε τρία φορτία διαφορετικού μεγέθους και πρόσημου

- 5.18 Δύο σωματίδια ίσης μάζας και φορτίου ίσου μέτρου κινούνται πάνω στο επίπεδο  $xy$  μέσα σε σταθερό μαγνητικό πεδίο  $\vec{B} = B\hat{z}$ .

Επιλύστε τις εξισώσεις κίνησης του συστήματος με τη μέθοδο Runge–Kutta 4ης τάξης. Αναπαραστήστε γραφικά τις τροχιές των σωματιδίων για αρχικές συνθήκες που θα επιλέξετε.

- 5.19 Τρία σωματίδια ίσης μάζας  $m$  συνδέονται με πανομοιότυπα ελατήρια σταθεράς  $k$  και φυσικού μήκους  $l$  και κινούνται χωρίς τριβές πάνω σε οριζόντιο επίπεδο. Επιλύστε τις εξισώσεις κίνησης του συστήματος με τη μέθοδο Runge–Kutta 4ης τάξης. Αναπαραστήστε γραφικά τις τροχιές των σωματιδίων για αρχικές συνθήκες που θα επιλέξετε. (Υπόδειξη: Δείτε τα αρχεία `rkA_3hoc.f90`, `rkA_3hoc.csh`)



Σχήμα 5.18: Δύο σωματίδια ίδιας μάζας στηριγμένα στα άκρα λεπτών και αβαρών ράβδων μήκους  $l$  συνδέονται με αβαρές ιδανικό ελατήριο σταθεράς  $k$  και φυσικού μήκους  $l$ . Τα σημεία ανάρτησης των ράβδων είναι σταθερά και απέχουν απόσταση  $l$  (Άσκηση 5.20).

- 5.20 Δύο σωματίδια ίδιας μάζας είναι στηριγμένα στα άκρα λεπτών και αβαρών ράβδων μήκους  $l$  και συνδέονται με αβαρές ιδανικό ελατήριο σταθεράς  $k$  και φυσικού μήκους  $l$  όπως στο σχήμα 5.18. Τα σημεία ανάρτησης των ράβδων είναι σταθερά και απέχουν απόσταση  $l$ . Υπολογίστε τη Lagrangian του συστήματος και από αυτή τις εξισώσεις κίνησης για τις γωνίες  $\theta_1$  και  $\theta_2$ . Λύστε αριθμητικά τις εξισώσεις αυτές με τη μέθοδο Runge–Kutta 4ης τάξης και στη συνέχεια αναπαραστήστε τις θέσεις των δύο εκκρεμών σε καρτεσιανές συντεταγμένες και κάνετε τη γραφική παράσταση της τροχιάς. Να τα κάνετε αυτά για τους κανονικούς τρόπους ταλάντωσης του συστήματος για μικρές ταλαντώσεις  $\theta_1 \lesssim 0.1$  και στη συνέχεια

παρατηρήστε τι γίνεται καθώς η γωνία αυξάνει. (Υπόδειξη: Δείτε τα αρχεία `rk_cpend.f90`, `rk_cpend.csh`)

- 5.21 Επαναλάβετε την προηγούμενη άσκηση όταν τα σημεία ανάρτησης των δύο εκκρεμών μπορούν να ολισθαίνουν χωρίς τριβές πάνω στον άξονα των  $x$ .
- 5.22 Επαναλάβετε την άσκηση 5.20 προσθέτοντας ένα τρίτο εκκρεμές δεξιά του δεύτερου σε ίδια απόσταση  $l$ .



## ΚΕΦΑΛΑΙΟ 6

### Κίνηση στο Χώρο

Στο κεφάλαιο αυτό θα μελετήσουμε την κίνηση σωματιδίου όταν αυτό κινείται στο χώρο (τρεις διαστάσεις). Θα εξετάσουμε την περίπτωση της μη σχετικιστικής αλλά και της σχετικιστικής κίνησης όταν η ταχύτητα του σωματιδίου πλησιάζει την ταχύτητα του φωτός στο κενό. Θα χρησιμοποιήσουμε τη μέθοδο Runge-Kutta μέθοδο με προσαρμοζόμενο βήμα 4ης–5ης τάξης (εν συντομία RK45) χρησιμοποιώντας λογισμικό που διατίθεται σε αποθετήρια λογισμικού. Στη συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε λογισμικό ελεύθερα διαθέσιμο από το αποθετήριο (repository) [www.netlib.org](http://www.netlib.org), και πιο συγκεκριμένα τη σουίτα rksuite των R.W. Brankin, I. Gladwell, and L.F. Shampine [38].

Στόχος του κεφαλαίου είναι να μάθετε να αναζητάτε λύσεις στα προβλήματα σας από λογισμικό που διατίθεται από άλλους προγραμματιστές και να αποκτήσετε εμπειρία στο πώς να βρίσκετε την πληροφορία, πώς να το χρησιμοποιήσετε και πώς να το συνδέσετε με το δικό σας πρόγραμμα. Στο τέλος, θα έχετε και ένα εργαλείο καλής ποιότητας για να λύνετε προβλήματα με τη μέθοδο αριθμητικής ολοκλήρωσης Runge-Kutta με προσαρμοζόμενο βήμα, μέθοδος που χρησιμοποιείται από πολλούς ερευνητές.

## 6.1 Runge–Kutta στις τρεις διαστάσεις.

Στις τρεις διαστάσεις, το πρόβλημα αρχικών τιμών που έχουμε να λύσουμε δίνεται από το σύστημα (4.6)

$$\begin{aligned}\frac{dx}{dt} &= v_x & \frac{dv_x}{dt} &= a_x(t, x, v_x, y, v_y, z, v_z) \\ \frac{dy}{dt} &= v_y & \frac{dv_y}{dt} &= a_y(t, x, v_x, y, v_y, z, v_z) \\ \frac{dz}{dt} &= v_z & \frac{dv_z}{dt} &= a_z(t, x, v_x, y, v_y, z, v_z).\end{aligned}\quad (6.1)$$

Στην περίπτωση αυτή θα χρησιμοποιήσουμε για αυξημένη ακρίβεια και υψηλότερη απόδοση έναν αλγόριθμο της οικογένειας Runge–Kutta με προσαρμοζόμενο έλεγχο βήματος (adaptive stepsize control). Για λεπτομέρειες παραπέμπουμε τον αναγνώστη στο βιβλίο [30]. Σκοπός μας εδώ δεν είναι να αναλύσουμε το συγκεκριμένο αλγόριθμο, αλλά να εξασκηθούμε στη χρήση προγραμμάτων που έχουν γράψει άλλοι για το συγκεκριμένο πρόβλημα που έχουμε να λύσουμε.

Φυσικά, το πρώτο που έχουμε να κάνουμε είναι να προσδιορίσουμε το κατάλληλο λογισμικό για το προς λύση πρόβλημα. Για το λόγο αυτό, ανάλογα με τη δυσκολία του προβλήματος αναζητούμε πληροφορίες στο δίκτυο, βιβλία σχετικά με το πρόβλημα και φυσικά, αν το πρόβλημά μας είναι ερευνητικού επιπέδου, αναζητούμε πληροφορίες στις ερευνητικές εργασίες και τους ειδικούς. Στη συγκεκριμένη περίπτωση, το πρόβλημά μας είναι σχετικά απλό και έχει πολλές και καλές λύσεις. Αναζητώντας λύση στο χώρο του ποιοτικού ελεύθερου λογισμικού αριθμητικών εφαρμογών, η πρώτη στάση που κάνουμε είναι στο [www.netlib.org](http://www.netlib.org) repository. Από τη λίστα του διαθέσιμου λογισμικού<sup>1</sup> επιλέγουμε τη βιβλιοθήκη ode και από αυτή τη σουίτα rk suite. Στο σύνδεσμο <http://www.netlib.org/ode/> διαβάζουμε

```
lib  rk suite
alg  Runge–Kutta
for  initial value problem for first order ordinary ↔
      differential
      equations. A suite of codes for solving IVPs in ODEs. A
      choice of RK methods, is available. Includes an error
      assessment facility and a sophisticated stiffness checker.
      Template programs and example results provided.
```

<sup>1</sup>Δυστυχώς το πακέτο diffpack ...αλλαξοπίστησε και πέρασε στο χώρο του εμπορικού λογισμικού.

```

Supersedes RKF45, DDERKF, D02PAF.
ref RKSUITE, Softreport 92-S1, Dept of Math, SMU, Dallas, ←
Texas
by R.W. Brankin (NAG), I. Gladwell and L.F. Shampine (SMU)
lang Fortran
prec double

```

από όπου μαθαίνουμε ότι το πακέτο έχει αλγόριθμους τύπου Runge-Kutta γραμμένους σε γλώσσα Fortran και αφορά πραγματικές μεταβλητές διπλής ακρίβειας (double precision). Κατεβάζουμε τα αρχεία `rksuite.f`, `rksuite.doc`, `details.doc`, `templates`, `readme`.

Για να χρησιμοποιήσουμε υπορουτίνες στο πρόγραμμά μας το πρώτο βήμα είναι να διαβάσουμε προσεκτικά τα εγχειρίδια χρήσης του πακέτου. Αυτό μπορεί να βρίσκεται (ανάλογα με την περίπτωση φυσικά) σε τυπωμένα έγγραφα, σε ηλεκτρονικά έγγραφα [html, pdf, ..., σε συνοδευτικά αρχεία με ονόματα τύπου README, INSTALL, ... ή αρχεία που βρίσκονται σε υποκαταλόγους με ονόματα όπως doc/..., σε online αρχεία βοήθειας (man pages) κ.ο.κ.]. Το καλό λογισμικό έχει όλη τη χρησιμη πληροφορία στα αρχεία που περιέχουν τον πηγαίο κώδικα, κάτι που ισχύει και στην περίπτωσή μας.

Για να συνδέσουμε υποπρογράμματα στο δικό μας πρόγραμμα χρειαζόμαστε τις εξής βασικές πληροφορίες:

- **INPUT DATA:** Δηλ. πώς παρέχουμε στο πρόγραμμά μας τις απαραίτητες πληροφορίες για να εκτελεστεί ο υπολογισμός. Είναι σαφές πως στην περίπτωσή μας χρειάζεται τουλάχιστον να δώσουμε τις αρχικές συνθήκες, το χρόνο ολοκλήρωσης και τον αριθμό βημάτων. Επίσης, ο χρήστης πρέπει να παρέχει τις συναρτήσεις στο δεξί μέλος της (6.1). Άλλες πληροφορίες που είναι δυνατόν να ζητούνται είναι λ.χ. επιθυμητή ακρίβεια, πληροφορίες για το hardware όπως διαθέσιμη αριθμητική ακρίβεια κλπ.
- **OUTPUT DATA:** Δηλ. πώς και πού το πρόγραμμα μας δίνει τα αποτελέσματα του υπολογισμού, αν αυτός έγινε ομαλά κλπ.
- **WORKSPACE:** Ειδικά σε ρουτίνες FORTRAN 77 που η μνήμη δεν ζητείται δυναμικά (αλλά όχι αναγκαστικά μόνο τότε) μπορεί να χρειαστεί να παρέχουμε στην υπορουτίνα χώρο στη μνήμη για τους ενδιάμεσους υπολογισμούς.

Η εγκατάσταση του λογισμικού είναι απλή. Όλος ο κώδικας είναι μέσα στο αρχείο `rksuite.f` και όπως μαθαίνουμε από το αρχείο

`rksuite.doc`<sup>2</sup> αρκεί να δώσουμε στο πρόγραμμα την τιμή τριών μεταβλητών που καθορίζουν την ακρίβεια υπολογισμού με υποδιαστολή στον υπολογιστή μας. Διαβάζουμε:

```
...
RKSUITE requires three environmental constants OUTCH, MCHEPS,
DWARF. When you use RKSUITE, you may need to know their
values. You can obtain them by calling the subroutine ENVIRN
in the suite:

    CALL ENVIRN(OUTCH, MCHPES, DWARF)

returns values

    OUTCH  — INTEGER
              Standard output channel on the machine being used.
    MCHEPS — DOUBLE PRECISION
              The unit of roundoff, that is, the largest
              positive number such that 1.0D0 + MCHEPS = 1.0D0.
    DWARF  — DOUBLE PRECISION
              The smallest positive number on the machine being
              used.
...
***** Installation Details *****

All machine-dependent aspects of the suite have been
isolated in the subroutine ENVIRN in the rksuite.for file.
Certain environmental parameters must be specified in this
subroutine. The values in the distribution version are
those appropriate to the IEEE arithmetic standard. They
must be altered, if necessary, to values appropriate to the
computing system you are using before calling the codes of
the suite. If the IEEE arithmetic standard values are not
appropriate for your system, appropriate values can often
be obtained by calling routines named in the Comments of
ENVIRN.
...
```

Δηλ. οι μεταβλητές `OUTCH`, `MCHEPS`, `DWARF` ορίζονται στην υπορουτίνα `ENVIRN` την οποία μπορεί να χρησιμοποιήσει για να τις χρησιμοποιήσει οπουδήποτε θέλει στο πρόγραμμα. Παρακάτω διαβάζουμε ότι το πρόγραμμα τις προ-ορίζει σε μάλλον ασφαλείς τιμές, αλλά αν ο προγραμματιστής πρέπει να τις αλλάξει<sup>3</sup>, τότε πρέπει να επέμβει στην υπο-

<sup>2</sup>Είναι ένα απλό text αρχείο που μπορείτε να διαβάσετε με την εντολή `less rksuite.doc` ή με τον `emacs`.

<sup>3</sup>Εδώ φαίνεται και η επαγγελματικότητα του συγγραφέα του κώδικα που προ-

ρουτίνα ENVIRN και να τις αλλάξει. Άρα, πρέπει να κοιτάξουμε μέσα στο αρχείο rksuite.f για να διαβάσουμε τα σχόλια της εν λόγω ρουτίνας:

```
...
      SUBROUTINE ENVIRN(OUTCH,MCHEPS,DWARF)
...
C The following six statements are to be Commented out
C after verification that the machine and installation
C dependent quantities are specified correctly.
...
      WRITE(*,*) ' Before using RKSUITE, you must verify that the '
      WRITE(*,*) ' machine- and installation-dependent quantities '
      WRITE(*,*) ' specified in the subroutine ENVIRN are correct, '
      WRITE(*,*) ' and then Comment these WRITE statements and the '
      WRITE(*,*) ' STOP statement out of ENVIRN. '
      STOP
...
C The following values are appropriate to IEEE
C arithmetic with the typical standard output channel.
C
      OUTCH = 6
      MCHEPS = 1.11D-16
      DWARF = 2.23D-308
```

Άρα, αρκεί να αφαιρέσουμε τις έξι εντολές WRITE και STOP κάνοντας τις σχόλια της Fortran<sup>4</sup>, αφού βεβαιωθούμε ότι οι επιλογές για τις τιμές των μεταβλητών OUTCH, MCHEPS, DWARF είναι ικανοποιητικές:

```
...
C      WRITE(*,*) ' Before using RKSUITE, you must verify that the '
C      WRITE(*,*) ' machine- and installation-dependent quantities '
C      WRITE(*,*) ' specified in the subroutine ENVIRN are correct, '
C      WRITE(*,*) ' and then Comment these WRITE statements and the '
C      WRITE(*,*) ' STOP statement out of ENVIRN. '
C      STOP
...
```

Ένας τρόπος να το ελέγξουμε με τη βοήθεια ελεύθερα διαθέσιμου λογισμικού είναι να αναζητήσουμε και να κατεβάσουμε τα αρχεία dlmach.f, ilmach.f από τη netlib.org, να τις τοποθετήσουμε στον υποκατάλογο blas/<sup>5</sup> και να γράψουμε το μικρό πρόγραμμα test\_envirn\_blas.f90

```
program testme
```

βλέπει ότι το πρόγραμμά του μπορεί να χρησιμοποιηθεί πολύ αργότερα κάτω από άγνωστες εξελίξεις στο hardware.

<sup>4</sup>Παρατηρήστε ότι στη μορφή κώδικα “fixed width” της παλιότερης Fortran, οι γραμμές που έχουν ως πρώτο χαρακτήρα το “C” θεωρούνται σχόλια και αγνοούνται από το μεταγλωττιστή.

<sup>5</sup>Σε επόμενο κεφάλαιο θα μάθουμε περισσότερα για τη βασική βιβλιοθήκη γραμμικής άλγεβρας blas.

```

implicit none
integer :: OUTCH
real(8) :: DWARF, MCHEPS
real(8) :: x
integer :: I1MACH !blas routines
real(8) :: D1MACH
OUTCH = I1MACH(2)
MCHEPS = D1MACH(3)
DWARF = D1MACH(1)
write(6,101)OUTCH,MCHEPS,DWARF

MCHEPS = epsilon(x)/2.0D0
DWARF = tiny(x)
write(6,101)OUTCH,MCHEPS,DWARF

101 format(I4,2E30.18)
end program testme

```

Όπως βλέπετε και από το παραπάνω πρόγραμμα, οι μεταβλητές MCHEPS και DWARF μπορούν να υπολογιστούν και από τις intrinsic συναρτήσεις της Fortran EPSILON() και TINY() χωρίς να χρειαστούμε τις συναρτήσεις της blas. Μεταγλωττίζουμε και τρέχουμε

```

> gfortran test_envirn_blas.f90 blas/*1mach.f -o test_envirn_blas
> ./test_envirn_blas
6      0.111022302462515654E-15      0.222507385850720138-307

```

και προκύπτει ότι οι επιλογές μας είναι ικανοποιητικές.

Το επόμενο βήμα είναι να μάθουμε να χρησιμοποιούμε τη ρουτίνα. Για το λόγο αυτό διαβάζουμε προσεκτικά το αρχείο rksuite.doc από το οποίο περιληπτικά μαθαίνουμε τα εξής: Το πρόγραμμα χρησιμοποιεί τη ρουτίνα UT (UT = “Usual Task”) για να κάνει την ολοκλήρωση με τη μέθοδο Runge-Kutta με προσαρμοζόμενο βήμα. Το βήμα προσαρμόζεται χρησιμοποιώντας Runge-Kutta 2ης-3ης τάξης (METHOD=1), 4ης-5ης τάξης (METHOD=2) ή 7ης-8ης τάξης (METHOD=3) από τις οποίες θα διαλέξουμε METHOD=2. Πριν καλέσουμε την UT, πρέπει να καλέσουμε μια υπορουτίνα αρχικοποίησης, την SETUP. Τέλος, ο χρήστης παρέχει την υπορουτίνα F η οποία ορίζει τις παραγώγους των συναρτήσεων, δηλ. στην περίπτωσή μας το δεξί μέλος των 6.1. Ένας γρήγορος τρόπος για να μάθουμε να χρησιμοποιούμε ένα πρόγραμμα είναι “by example”. Στην περίπτωσή μας στο πακέτο παρέχονται δοκιμαστικά προγράμματα για εκμάθηση και έλεγχο ορθότητας. Αυτά βρίσκονται στο αρχείο templates το οποίο ανοίγει από μόνο του με το πρόγραμμα φλοιού sh:

```
> sh templates
tmpl1.out
tmpl1a.f
...
```

Το πρόγραμμα `tmpl1a.f` έχει τη λύση για τον αρμονικό ταλαντωτή και φαίνεται με πολλά επεξηγηματικά σχόλια η απλή χρήση του προγράμματος, την οποία επιλέγουμε τελικά στη δικιά μας περίπτωση. Έτσι καταλήγουμε στο παρακάτω πρόγραμμα για την οδήγηση της ολοκλήρωσης, το οποίο αποθηκεύουμε στο αρχείο `rk3.f90`

```
!=====
!Program to solve a 6 ODE system using Runge-Kutta Method
!Output is written in file rk3.dat
!=====
program rk3_solve
  include 'rk3.inc'
  real(8) :: T0,TF,X10,X20,X30,V10,V20,V30
  real(8) :: t,dt,tstep
  integer :: STEPS
  integer :: i
  real(8) :: energy
!Arrays/variables needed by rksuite:
  real(8) TOL,THRES(NEQ),WORK(LENWRK),Y(NEQ),YMAX(NEQ),&
    YP(NEQ),YSTART(NEQ),HSTART
  logical ERRASS, MESSAGE
  integer UFLAG
!.. External Subroutines ..
  EXTERNAL F, SETUP, STAT, UT
!Input:
  print *, 'Runge-Kutta Method for 6-ODEs Integration'
  print *, 'Enter coupling constants k1,k2,k3,k4:'
  read *, k1,k2,k3,k4
  print *, 'k1= ',k1, ' k2= ',k2, ' k3= ',k3, ' k4= ',k4
  print *, 'Enter STEPS,T0,TF,X10,X20,X30,V10,V20,V30:'
  read *, STEPS,T0,TF,X10,X20,X30,V10,V20,V30
  print *, 'No. Steps= ',STEPS
  print *, 'Time: Initial T0 =',T0, ' Final TF=',TF
  print *, ' X1(T0)=',X10, ' X2(T0)=',X20, ' X3(T0)=',X30
  print *, ' V1(T0)=',V10, ' V2(T0)=',V20, ' V3(T0)=',V30
!Initial Conditions
  dt = (TF-T0)/STEPS
  YSTART(1) = X10
  YSTART(2) = X20
  YSTART(3) = X30
  YSTART(4) = V10
  YSTART(5) = V20
```

```

YSTART(6) = V30
!
! Set error control parameters.
!
TOL = 5.0D-6
do i = 1, NEQ
  THRES(i) = 1.0D-10
enddo
MESSAGE = .TRUE.
ERRASS = .FALSE.
HSTART = 0.0D0
! Initialization:
call SETUP(NEQ, T0, YSTART, TF, TOL, THRES, METHOD, 'Usual Task', &
  ERRASS, HSTART, WORK, LENWRK, MESSAGE)
open(unit=11, file='rk3.dat')
write(11,100) T0, YSTART(1), YSTART(2), YSTART(3), YSTART(4), &
  YSTART(5), YSTART(6), energy(T0, YSTART)
! Calculation:
do i=1, STEPS
  t = T0 + i*dt
  call UT(F, t, tstep, Y, YP, YMAX, WORK, UFLAG)
  if(UFLAG.GT.2) exit !exit the loop: go after enddo
  write(11,100) tstep, Y(1), Y(2), Y(3), Y(4), Y(5), Y(6), &
    energy(tstep, Y)
enddo
close(11)
100 format(8E25.15)
end program rk3_solve

```

Παρατηρούμε ότι τους κοινούς ορισμούς τους τοποθετήσαμε στο ξεχωριστό αρχείο rk3.inc για να χρησιμοποιηθούν και από τη συνάρτηση των παραγώγων. Τα περιεχόμενα του αρχείου αυτού τοποθετούνται στη σειρά include 'rk3.inc':

```

!Basic definitions of variables for the suite rksuite
implicit none
!NEQ is the number of equations, 6 in 3 dimensions
!METHOD=2 is for RK45.
INTEGER          NEQ,   LENWRK,          METHOD
PARAMETER        (NEQ=6, LENWRK=32*NEQ, METHOD=2)
REAL *8          k1, k2, k3, k4 !force couplings
COMMON /COUPLINGS/ k1, k2, k3, k4

```

Παρατηρούμε ότι εδώ θέτουμε τον αριθμό των διαφορικών εξισώσεων NEQ=6, καθώς και τη μέθοδο ολοκλήρωσης METHOD=2. Η μεταβλητή LENWRK καθορίζει το μέγεθος της μνήμης που χρειάζεται το πρόγραμμα για τους ενδιάμεσους υπολογισμούς. Το πρόγραμμα αρχίζει ακριβώς όπως και



τα προηγούμενα, αφήνοντας έτσι το interface με το χρήστη αναλλοίωτο. Οι αρχικές θέσεις και ταχύτητες αποθηκεύονται στο array YSTART στις θέσεις 1...6. Στις 3 πρώτες θέσεις έχουμε τις συντεταγμένες χώρου, ενώ στις 3 τελευταίες τις συντεταγμένες της ταχύτητας. Αφού καθορίσουμε μερικές μεταβλητές που καθορίζουν τη συμπεριφορά του προγράμματος (δες αρχείο rksuite.doc για λεπτομέρειες), καλούμε την υπορουτίνα SETUP. Στη συνέχεια, τυπώνουμε τις αρχικές συνθήκες στο αρχείο εξόδου rk3.dat και φτάνουμε στην καρδιά του προγράμματος, την ολοκλήρωση:

```
do i=1,STEPS
  t = T0 + i*dt
  call UT(F,t,tstep,Y,YP,YMAX,WORK,UFLAG)
  if(UFLAG.GT.2) exit !exit the loop: go after enddo
  write(11,100) tstep,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),&
    energy(tstep,Y)
enddo
```

F είναι η συνάρτηση που υπολογίζει τις παραγώγους, γραμμένη από εμάς παρακάτω. t είναι ο χρόνος στον οποίο επιθυμούμε να έχουμε το αποτέλεσμα της ολοκλήρωσης. Λόγω προσαρμοζόμενου βήματος, αυτός μπορεί να μην είναι ακριβώς ο ίδιος με αυτόν που τελικά μας επιστρέφει η υπορουτίνα, δηλ. τον tstep. Y είναι οι τιμές των συναρτήσεων, δηλ.  $x = Y(1)$ ,  $y = Y(2)$ ,  $z = Y(3)$  και  $v_x = Y(4)$ ,  $v_y = Y(5)$ ,  $v_z = Y(6)$ . energy(t,Y) είναι η συνάρτηση που υπολογίζει τη μηχανική ενέργεια του συστήματος την οποία θα γράψουμε στο ίδιο αρχείο με τη συνάρτηση F. Τέλος, η μεταβλητή UFLAG είναι “σημαία” που δείχνει ότι η UT επιστρέφει με σφάλμα, οπότε το πρόγραμμα τερματίζει. Παρακάτω, παραθέτουμε τον κώδικα δοκιμής, την κίνηση βλήματος στο πεδίο βαρύτητας με δύναμη αντίστασης από ρευστό ανάλογη της ταχύτητας του βλήματος  $\vec{F}_r = -mk\vec{v}$ . Ο κώδικας αποθηκεύεται στο αρχείο rk3\_g.f90. Παίρνουμε  $\vec{g} = -k1 \hat{k}$  και  $k = k2$ .

```
!-----
subroutine F(T,Y,YP)
  include 'rk3.inc'
  real(8) :: t
  real(8) :: Y(*),YP(*)
  real(8) :: x1,x2,x3,v1,v2,v3
  x1 = Y(1);v1 = Y(4)
  x2 = Y(2);v2 = Y(5)
  x3 = Y(3);v3 = Y(6)
  !Velocities:  dx_i/dt = v_i
```

```

YP(1) = v1
YP(2) = v2
YP(3) = v3
! Acceleration: dv_i/dt = a_i
YP(4) = -k2*v1
YP(5) = -k2*v2
YP(6) = -k2*v3-k1
end subroutine F
!
real(8) function energy(T,Y)
include 'rk3.inc'
real(8) :: t,e
real(8) :: Y(*)
real(8) :: x1,x2,x3,v1,v2,v3
x1 = Y(1);v1 = Y(4)
x2 = Y(2);v2 = Y(5)
x3 = Y(3);v3 = Y(6)
! Kinetic Energy
e = 0.5*(v1*v1+v2*v2+v3*v3)
! Potential Energy
e = e + k1*x3
energy = e
end function energy

```

Για ευκολία, “μεταφράσαμε” τις τιμές του array Y(NEQ) σε μεταβλητές θέσης και ταχύτητας και μετά χρησιμοποιήσαμε τους γνωστούς τύπους<sup>6</sup>. Η μεταγλώττιση, υποθέτοντας ότι τη σουίτα rksuite.f την τοποθετήσαμε στον υποκατάλογο rksuite/, το τρέξιμο και η επισκόπηση των αποτελεσμάτων με το gnuplot γίνεται με τις εντολές:

```

> gfortran rk3.f90 rk3_g.f90 rksuite/rksuite.f -o rk3
> ./rk3
Runge-Kutta Method for 6-ODEs Integration
Enter coupling constants k1,k2,k3,k4:
10 0 0 0
k1= 10.0000 k2= 0.0000E+000 k3=
0.0000E+000 k4= 0.0000E+000
Enter STEPS,T0,TF,X10,X20,X30,V10,V20,V30:
10000 0 3 0 0 0 1 1 1
No. Steps= 10000

```

<sup>6</sup>Παρατηρήστε πώς δηλώσαμε τα arrays Y, YP: `real(8) :: Y(*),YP(*)`. Τα arrays αυτά για τις F, energy είναι “assumed-size” arrays, δηλ. arrays των οποίων το μέγεθος είναι άγνωστο στη διαδικασία. Αν έχουμε arrays με περισσότερους δείκτες, μόνο η τελευταία διάσταση επιτρέπεται να είναι \*. Γενικά είναι καλύτερο να αποφεύγεται σε καινούργια προγράμματα η χρήση assumed-size arrays και να προτιμώνται τα assumed-shape όπως κάναμε στο πρόγραμμα rkA.f90 στη σελ. 293. Η δήλωση στην περίπτωση αυτή θα ήταν `real(8) :: Y(:),YP(:)`

```

Time: Initial T0 = 0.0000E+000 Final TF= 3.0000
      X1(T0)= 0.0000E+000 X2(T0)= 0.0000E+000
      X3(T0)= 0.0000E+000
      V1(T0)= 1.0000 V2(T0)= 1.0000
      V3(T0)= 1.0000

> gnuplot
gnuplot> plot "rk3.dat" using 1:2 with lines title "x1(t)"
gnuplot> plot "rk3.dat" using 1:3 with lines title "x2(t)"
gnuplot> plot "rk3.dat" using 1:4 with lines title "x3(t)"
gnuplot> plot "rk3.dat" using 1:5 with lines title "v1(t)"
gnuplot> plot "rk3.dat" using 1:6 with lines title "v2(t)"
gnuplot> plot "rk3.dat" using 1:7 with lines title "v3(t)"
gnuplot> plot "rk3.dat" using 1:8 with lines title "E(t)"
gnuplot> set title "trajectory"
gnuplot> splot "rk3.dat" using 2:3:4 with lines notitle

```

Την παραπάνω εργασία την έχουμε κωδικοποιήσει σε σενάριο φλοιού (shell script) με όνομα rk3.csh. Από το αρχείο αυτό χρησιμοποιείται το ανάλογο αρχείο για animation με το όνομα rk3\_animate.csh. Έτσι, η παραπάνω δουλειά συνοψίζεται στην παρακάτω εντολή:

```
./rk3.csh -f 1 — 10 0. 0 0 0 0 0 1 1 1 10000 0 3
```

## 6.2 Κίνηση Σωματίου σε ΗΜ πεδίο.

Μετά την ανάλυση της προηγούμενης παραγράφου, έχουμε τα απαραίτητα εργαλεία να μελετήσουμε τη μη-σχετικιστική κίνηση φορτισμένου σωματίου μέσα σε ηλεκτρομαγνητικό (ΗΜ) πεδίο. Αυτό υπόκειται στην επίδραση της δύναμης Lorentz:

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B}). \quad (6.2)$$

Ας θεωρήσουμε πρώτα την απλή περίπτωση σταθερού ΗΜ πεδίου της μορφής  $\vec{E} = E_x \hat{x} + E_y \hat{y} + E_z \hat{z}$ ,  $\vec{B} = B \hat{z}$ . Οι συνιστώσες της επιτάχυνσης του σωματίου θα είναι:

$$\begin{aligned} a_x &= (qE_x/m) + (qB/m)v_y \\ a_y &= (qE_y/m) - (qB/m)v_x \\ a_z &= (qE_z/m). \end{aligned} \quad (6.3)$$

Προγραμματίζουμε το παραπάνω δυναμικό πεδίο στο αρχείο rk3\_B.f90, θέτοντας  $k1 = qB/m$ ,  $k2 = qE_x/m$ ,  $k3 = qE_y/m$ ,  $k4 = qE_z/m$ :

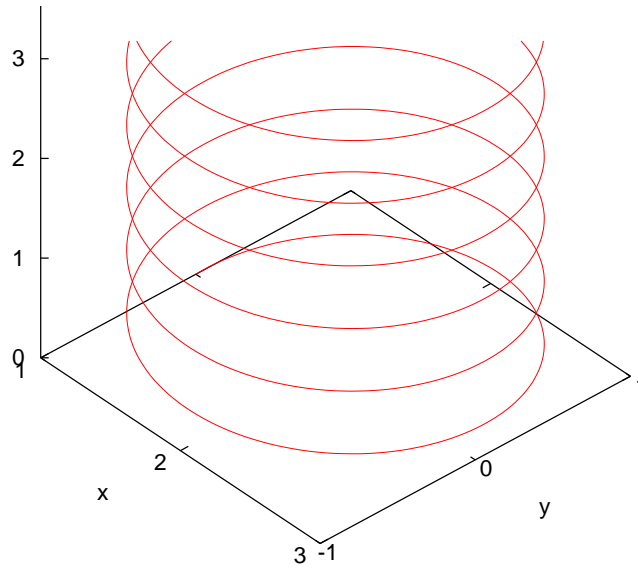
```

!-----
!Particle in constant Magnetic and electric field
!q B/m = k1 z    q E/m = k2 x + k3 y + k4 z
subroutine F(T,Y,YP)
  include 'rk3.inc'
  real(8) :: t
  real(8) :: Y(*),YP(*)
  real(8) :: x1,x2,x3,v1,v2,v3
  x1 = Y(1);v1 = Y(4)
  x2 = Y(2);v2 = Y(5)
  x3 = Y(3);v3 = Y(6)
!Velocities:  dx_i/dt = v_i
  YP(1) = v1
  YP(2) = v2
  YP(3) = v3
!Acceleration: dv_i/dt = a_i
  YP(4) = k2 + k1 * v2
  YP(5) = k3 - k1 * v1
  YP(6) = k4
end subroutine F
!-----
real(8) function energy(T,Y)
  include 'rk3.inc'
  real(8) :: t,e
  real(8) :: Y(*)
  real(8) :: x1,x2,x3,v1,v2,v3
  x1 = Y(1);v1 = Y(4)
  x2 = Y(2);v2 = Y(5)
  x3 = Y(3);v3 = Y(6)
!Kinetic Energy
  e = 0.5*(v1*v1+v2*v2+v3*v3)
!Potential Energy
  e = e - k2*x1 - k3*x2 - k4*x3
  energy = e
end function energy

```

Με παρόμοιο τρόπο μπορούμε να μελετήσουμε πεδία τα οποία είναι χωροεξαρτημένα. Οι επιλογές μας πρέπει να ικανοποιούν τις εξισώσεις του Maxwell! Για να δούμε τον περιορισμό στο χώρο φορτισμένου σωματιδίου με την επίδραση μαγνητικού πεδίου, παίρνουμε τις απλές περιπτώσεις  $\vec{B} = B_y \hat{y} + B_z \hat{z}$  με  $qB_y/m = -k_2 y$ ,  $qB_z/m = k_1 + k_2 z$  και  $qB_y/m = k_3 z$ ,  $qB_z/m = k_1 + k_2 y$ . Παρατηρούμε ότι ισχύει  $\vec{\nabla} \cdot \vec{B} = 0$ . Εσείς υπολογίστε την πυκνότητα ρεύματος από την εξίσωση  $\vec{\nabla} \times \vec{B} = \mu_0 \vec{j}$ .

Τα αποτελέσματά μας φαίνονται στα σχήματα 6.1–6.4.



Σχήμα 6.1: Τροχιά φορτισμένου σωματιδίου σε σταθερό μαγνητικό πεδίο  $\vec{B} = B\hat{z}$  με  $qB/m = 1.0$ ,  $\vec{v}(0) = 1.0\hat{y} + 0.1\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 1000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 40$ .

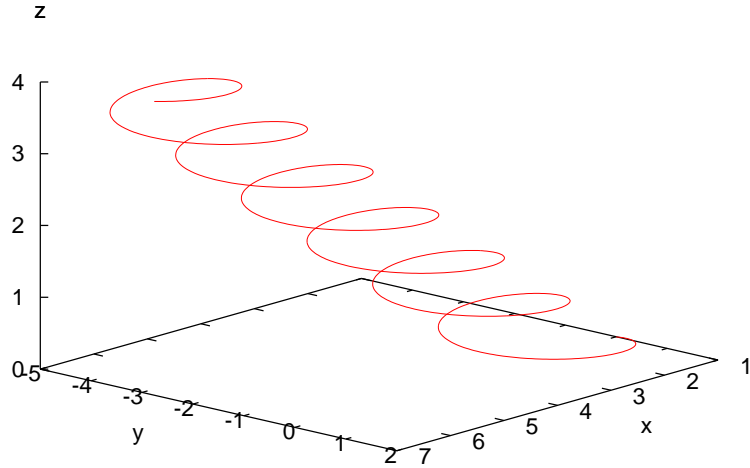
### 6.3 Σχετικιστική Κίνηση.

Στην παράγραφο αυτή θα συζητήσουμε τον υπολογισμό τροχιάς σωματιδίου μη μηδενικής μάζας ηρεμίας, όταν η ταχύτητά του γίνεται συγκρίσιμη με αυτή του φωτός. Παρακάτω, θα θέσουμε την ταχύτητα του φωτός στο κενό  $c = 1$ . Οι εξισώσεις κίνησης σωματιδίου μάζας ηρεμίας  $m_0 > 0$ , μάζας  $m = m_0/\sqrt{1-v^2}$ , ορμής  $\vec{p} = m\vec{v}$  και ενέργειας  $E = m = \sqrt{p^2 + m_0^2}$  μέσα σε δυναμικό πεδίο  $\vec{F}$  δίνονται από τις σχέσεις:

$$\frac{d\vec{p}}{dt} = \vec{F}. \quad (6.4)$$

Για να τις γράψουμε σε σύστημα διαφορικών εξισώσεων πρώτης τάξης χρησιμοποιούμε τις σχέσεις:

$$\vec{v} = \frac{\vec{p}}{m} = \frac{\vec{p}}{E} = \frac{\vec{p}}{\sqrt{p^2 + m_0^2}}, \quad (6.5)$$

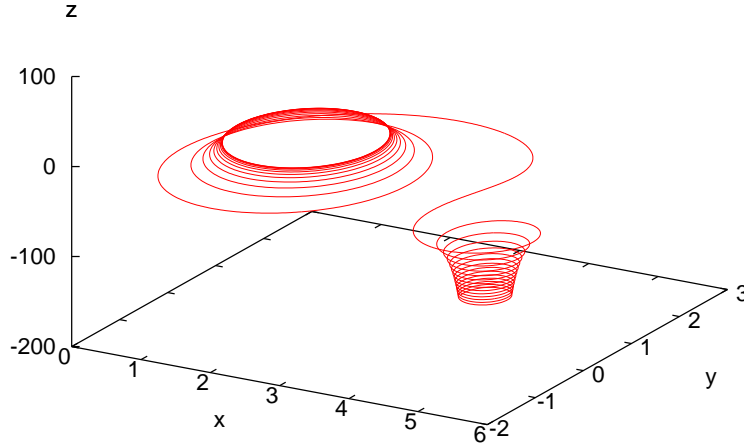


Σχήμα 6.2: Τροχιά φορτισμένου σωματιδίου σε σταθερό μαγνητικό πεδίο  $\vec{B} = B\hat{z}$  με  $qB/m = 1.0$ , και σταθερό ηλεκτρικό πεδίο  $\vec{E} = E_x\hat{x} + E_y\hat{y}$  με  $qE_x/m = qE_y/m = 0.1$ ,  $\vec{v}(0) = 1.0\hat{y} + 0.1\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 1000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 40$ . Η τροχιά δεν είναι υπό κλίμακα, προσέξτε τις διαφορετικές κλίμακες στους τρεις άξονες.

οι οποίες μαζί με την  $\vec{v} = d\vec{r}/dt$  μας δίνουν:

$$\begin{aligned} \frac{dx}{dt} &= \frac{(p_x/m_0)}{\sqrt{1 + (p/m_0)^2}}, & \frac{d(p_x/m_0)}{dt} &= \frac{F_x}{m_0} \\ \frac{dy}{dt} &= \frac{(p_y/m_0)}{\sqrt{1 + (p/m_0)^2}}, & \frac{d(p_y/m_0)}{dt} &= \frac{F_y}{m_0} \\ \frac{dz}{dt} &= \frac{(p_z/m_0)}{\sqrt{1 + (p/m_0)^2}}, & \frac{d(p_z/m_0)}{dt} &= \frac{F_z}{m_0}, \end{aligned} \quad (6.6)$$

που αποτελούν ένα σύστημα διαφορικών εξισώσεων πρώτης τάξης για τις συναρτήσεις  $(x(t), y(t), z(t), (p_x/m_0)(t), (p_y/m_0)(t), (p_z/m_0)(t))$ . Για τη λύση με τη μέθοδο Runge-Kutta προσαρμοσμένου βήματος 4ης – 5ης τάξης σύμφωνα με τις προηγούμενες παραγράφους, χρειαζόμαστε τις αρχικές συνθήκες  $(x(0), y(0), z(0), (p_x/m_0)(0), (p_y/m_0)(0), (p_z/m_0)(0))$ . Χρη-



Σχήμα 6.3: Τροχιά φορτισμένου σωματιδίου σε μαγνητικό πεδίο  $\vec{B} = B_y \hat{y} + B_z \hat{z}$  με  $qB_y/m = -0.02y$ ,  $qB_z/m = 1 + 0.02z$ ,  $\vec{v}(0) = 1.0\hat{y} + 0.1\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 10000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 500$ . Η τροχιά δεν είναι υπό κλίμακα, προσέξτε τις διαφορετικές κλίμακες στους τρεις άξονες.

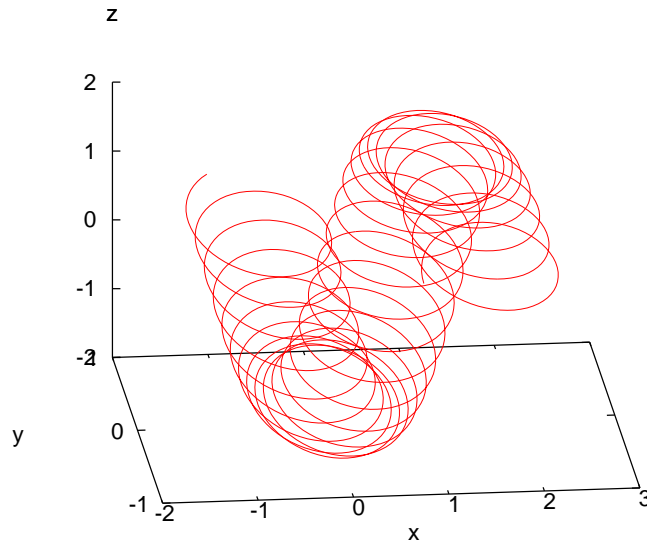
σιμοποιώντας τις σχέσεις

$$\begin{aligned} (p_x/m_0) &= \frac{v_x}{\sqrt{1-v^2}} & v_x &= \frac{(p_x/m_0)}{\sqrt{1+(p/m_0)^2}} \\ (p_y/m_0) &= \frac{v_y}{\sqrt{1-v^2}} & v_y &= \frac{(p_y/m_0)}{\sqrt{1+(p/m_0)^2}} \\ (p_z/m_0) &= \frac{v_z}{\sqrt{1-v^2}} & v_z &= \frac{(p_z/m_0)}{\sqrt{1+(p/m_0)^2}} \end{aligned} \quad (6.7)$$

μπορούμε να δώσουμε εναλλακτικά ως αρχικές συνθήκες  $(x(0), y(0), z(0), v_x(0), v_y(0), v_z(0))$ , καθώς και από τις λύσεις  $(x(t), y(t), z(t), (p_x/m_0)(t), (p_y/m_0)(t), (p_z/m_0)(t))$  να πάρουμε τις  $(x(t), y(t), z(t), v_x(t), v_y(t), v_z(t))$ . Προσοχή όμως να ελέγχουμε ότι ισχύει πάντα ( $m_0 > 0$ )

$$v^2 = (v_x)^2 + (v_y)^2 + (v_z)^2 < 1. \quad (6.8)$$

Για τον προγραμματισμό του παραπάνω προβλήματος χρειάζεται να



Σχήμα 6.4: Τροχιά φορτισμένου σωματιδίου σε μαγνητικό πεδίο  $\vec{B} = B_y\hat{y} + B_z\hat{z}$  με  $qB_y/m = 0.08z$ ,  $qB_z/m = 1.4 + 0.08y$ ,  $\vec{v}(0) = 1.0\hat{y} + 0.1\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 40000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 3000$ . Η τροχιά δεν είναι υπό κλίμακα, προσέξτε τις διαφορετικές κλίμακες στους τρεις άξονες.

μεταβάλλουμε ελαφρά το πρόγραμμα `rk3.f90`. Οι αλλαγές αφορούν το κυρίως πρόγραμμα μόνο στη σχέση ταχυτήτων ορμών που μπορεί να επιθυμεί να μελετήσει ο χρήστης. Όσο για το αρχείο με το οποίο προγραμματίζουμε το δυναμικό πεδίο, χρειάζεται να μεταβάλλουμε μόνο τις σχέσεις για την ταχύτητα, αφού η προς ολοκλήρωση συνάρτηση είναι τώρα η ορμή. Ας αρχίσουμε πρώτα με το κυρίως πρόγραμμα, `sr.f90`:

```
!=====
!Program to solve a 6 ODE system using Runge-Kutta Method
!Output is written in file sr.dat
!Interface to be used with relativistic particles.
!=====
program sr_solve
  include 'sr.inc'
  real(8) :: T0,TF,X10,X20,X30,V10,V20,V30
  real(8) :: P10,P20,P30
  real(8) :: P1,P2,P3,V1,V2,V3
  real(8) :: t,dt,tstep
```



```

integer :: STEPS
integer :: i
real(8) :: energy
!Arrays/variables needed by rksuite:
real(8) :: TOL,THRES(NEQ), WORK(LENWRK), Y(NEQ), YMAX(NEQ),&
        YP(NEQ), YSTART(NEQ),HSTART
logical :: ERRASS, MESSAGE
integer :: UFLAG
!.. External Subroutines ..
EXTERNAL          F, SETUP, STAT, UT
!Input:
print *, 'Runge-Kutta Method for 6-ODEs Integration '
print *, 'Special Relativistic Particle:'
print *, 'Enter coupling constants k1,k2,k3,k4:'
read  *, k1,k2,k3,k4
print *, 'k1= ',k1, ' k2= ',k2, ' k3= ',k3, ' k4= ',k4
print *, 'Enter STEPS,T0,TF,X10,X20,X30,V10,V20,V30:'
read  *, STEPS,T0,TF,X10,X20,X30,V10,V20,V30
call momentum(V10,V20,V30,P10,P20,P30)
print *, 'No. Steps= ',STEPS
print *, 'Time: Initial T0 =',T0, ' Final TF=',TF
print *, '          X1(T0)=',X10, ' X2(T0)=',X20, ' X3(T0)=',X30
print *, '          V1(T0)=',V10, ' V2(T0)=',V20, ' V3(T0)=',V30
print *, '          P1(T0)=',P10, ' P2(T0)=',P20, ' P3(T0)=',P30

!Initial Conditions
dt      = (TF-T0)/STEPS
YSTART(1) = X10
YSTART(2) = X20
YSTART(3) = X30
YSTART(4) = P10
YSTART(5) = P20
YSTART(6) = P30
!
!  Set error control parameters.
!
TOL = 5.0D-6
do i = 1, NEQ
    THRES(i) = 1.0D-10
enddo
MESSAGE = .TRUE.
ERRASS  = .FALSE.
HSTART  = 0.0D0
!Initialization:
call SETUP(NEQ,T0,YSTART,TF,TOL,THRES,METHOD,'Usual Task',&
        ERRASS,HSTART,WORK,LENWRK,MESSAGE)
open(unit=11,file='sr.dat')
call velocity(YSTART(4),YSTART(5),YSTART(6),V1,V2,V3)
write(11,100) T0,YSTART(1),YSTART(2),YSTART(3),&

```

```

        V1,V2,V3,&
        energy(T0,YSTART),&
        YSTART(4),YSTART(5),YSTART(6)
! Calculation:
do i=1,STEPS
    t = T0 + i*dt
    call UT(F,t,tstep,Y,YP,YMAX,WORK,UFLAG)
    if(UFLAG.GT.2) exit
    call velocity(Y(4),Y(5),Y(6),V1,V2,V3)
    write(11,100) tstep,Y(1),Y(2),Y(3),&
        V1,V2,V3,&
        energy(tstep,Y),&
        Y(4),Y(5),Y(6)
enddo
close(11)
100 format(11E25.15)
end program sr_solve
!=====
!momentum -> velocity transformation
!=====
subroutine velocity(p1,p2,p3,v1,v2,v3)
    implicit none
    real(8) :: v1,v2,v3,p1,p2,p3,v,p,vsq,psq

    psq = p1*p1+p2*p2+p3*p3

    v1 = p1/sqrt(1.0D0+psq)
    v2 = p2/sqrt(1.0D0+psq)
    v3 = p3/sqrt(1.0D0+psq)
end subroutine velocity
!=====
!velocity -> momentum transformation
!=====
subroutine momentum(v1,v2,v3,p1,p2,p3)
    implicit none
    real(8) :: v1,v2,v3,p1,p2,p3,v,p,vsq,psq

    vsq = v1*v1+v2*v2+v3*v3
    if(vsq .ge. 1.0D0) stop 'sub momentum: vsq >= 1'
    p1 = v1/sqrt(1.0D0-vsq)
    p2 = v2/sqrt(1.0D0-vsq)
    p3 = v3/sqrt(1.0D0-vsq)
end subroutine momentum

```

Παρατηρούμε το ρόλο που παίζουν οι υπορουτίνες `momentum` και `velocity` οι οποίες αναλαμβάνουν τους μετασχηματισμούς (6.7). Εκεί γίνεται και ο έλεγχος της συνθήκης (6.8). Θα τις χρησιμοποιήσουμε και στο αρχείο που θα γράψουμε το πρόγραμμα που θα δίνει τις παραγώγους των

συναρτήσεων για κάθε πεδίο δυνάμεων που θα θελήσουμε να μελετήσουμε.

Η πρώτη μας απόπειρα είναι να μελετήσουμε την κίνηση σχετικιστικού φορτισμένου σωματιδίου μέσα σε σταθερό ΗΜ πεδίο. Μέσα στο πεδίο αυτό η επιτάχυνση του σωματιδίου δίνεται από τις σχέσεις (6.3). Οι σχέσεις αυτές προγραμματίζονται μέσα στο αρχείο `sr_B.f90`. Πέρα από τις αλλαγές που αναφέραμε μέχρι τώρα, πρέπει να προσέξουμε και τον ορισμό της κινητικής ενέργειας:

$$T = \left( \frac{1}{\sqrt{1-v^2}} - 1 \right) m_0 = \left( \sqrt{1 + (p/m_0)^2} - 1 \right) m_0 \quad (6.9)$$

Το περιεχόμενο του `sr_B.f90` είναι:

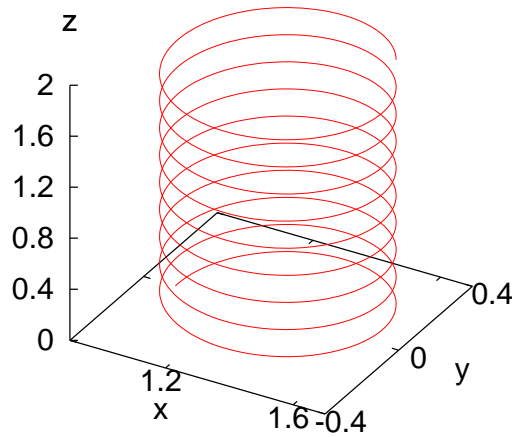
```
!=====
! Particle in constant Magnetic and electric field
! q B/m = k1 z    q E/m = k2 x + k3 y + k4 z
!=====
subroutine F(T,Y,YP)
  include 'sr.inc'
  real(8) :: t
  real(8) :: Y(*),YP(*)
  real(8) :: x1,x2,x3,v1,v2,v3,p1,p2,p3
  x1 = Y(1);p1 = Y(4)
  x2 = Y(2);p2 = Y(5)
  x3 = Y(3);p3 = Y(6)
  call velocity(p1,p2,p3,v1,v2,v3)
!now we can use all x1,x2,x3,p1,p2,p3,v1,v2,v3
  YP(1) = v1
  YP(2) = v2
  YP(3) = v3
! Acceleration:
  YP(4) = k2 + k1 * v2
  YP(5) = k3 - k1 * v1
  YP(6) = k4
end subroutine F
!=====
!Energy per unit rest mass
!=====
real(8) function energy(T,Y)
  include 'sr.inc'
  real(8) :: t,e
  real(8) :: Y(*)
  real(8) :: x1,x2,x3,v1,v2,v3,p1,p2,p3,psq
  x1 = Y(1);p1 = Y(4)
  x2 = Y(2);p2 = Y(5)
  x3 = Y(3);p3 = Y(6)
```

```

psq= p1*p1+p2*p2+p3*p3
!Kinetic Energy/m_0
e = sqrt(1.0D0+psq)-1.0D0
!Potential Energy/m_0
e = e - k2*x1 - k3*x2 - k4*x3
energy = e
end function energy

```

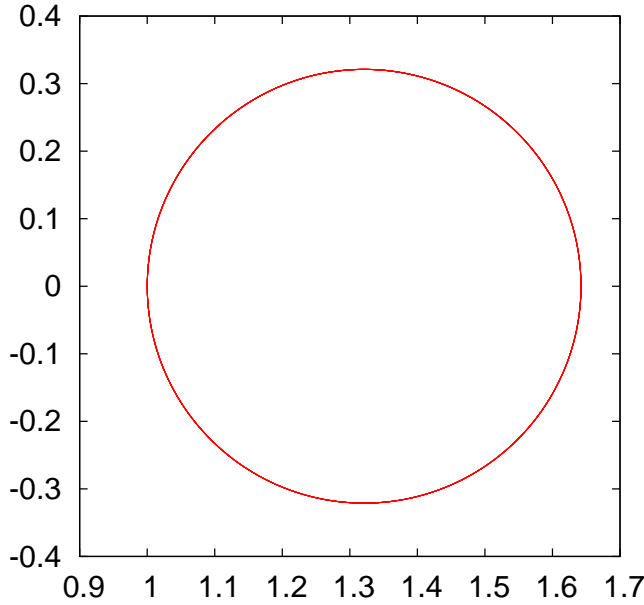
Τα αποτελέσματά μας τα δείχνουμε στα σχήματα 6.5–6.6.



Σχήμα 6.5: Τροχιά σχετικιστικού φορτισμένου σωματιδίου, όταν κινείται μέσα σε μαγνητικό πεδίο  $\vec{B} = B_z \hat{z}$  με  $qB_z/m_0 = 10.0$ ,  $\vec{v}(0) = 0.95\hat{y} + 0.10\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 1000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 20$ . Η τροχιά δεν είναι υπό κλίμακα, προσέξτε τις διαφορετικές κλίμακες στους άξονες.

Αφού βεβαιωθήκαμε για την επιτυχία της προσέγγισης του προβλήματος για το φορτισμένο σωματίο σε σταθερό ΗΜ πεδίο, μπορούμε να προσπαθήσουμε να μελετήσουμε ένα πιο ενδιαφέρον πρόβλημα. Θα φτιάξουμε ένα απλό μοντέλο για την ακτινοβολία Van Allen της γης. Θα υποθέσουμε ότι τα ηλεκτρόνια κινούνται στο μαγνητικό πεδίο της γης το οποίο προσεγγίζεται ως μαγνητικό πεδίο διπόλου της μορφής:

$$\vec{B} = B_0 \left( \frac{R_E}{r} \right)^3 \left[ 3(\hat{d} \cdot \hat{r}) \hat{r} - \hat{d} \right], \quad (6.10)$$



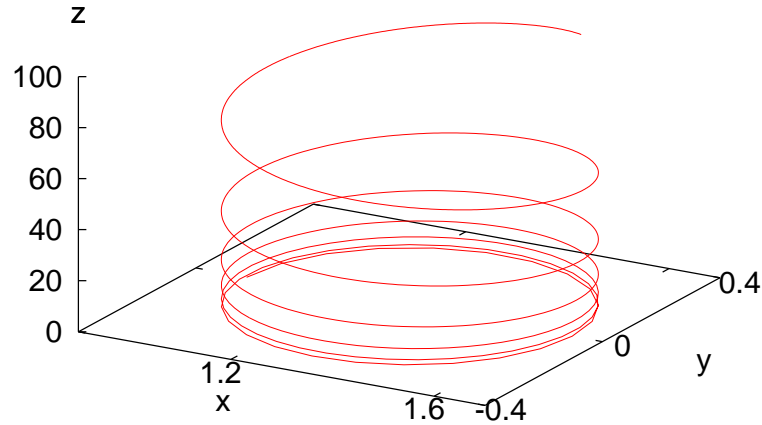
Σχήμα 6.6: Προβολή της τροχιάς σχετικιστικού φορτισμένου σωματιδίου στο επίπεδο  $xy$  όταν κινείται μέσα σε μαγνητικό πεδίο  $\vec{B} = B_z \hat{z}$  με  $qB_z/m_0 = 10.0$ ,  $\vec{v}(0) = 0.95\hat{y} + 0.10\hat{z}$ ,  $\vec{r}(0) = 1.0\hat{x}$ . Η ολοκλήρωση των εξισώσεων κίνησης γίνεται με τη μέθοδο RK45 με 1000 βήματα από  $t_0 = 0$  μέχρι  $t_f = 20$ .

όπου  $\vec{d} = d\hat{d}$  η μαγνητική ροπή διπόλου του μαγνητικού πεδίου της γης και φυσικά  $\vec{r} = r\hat{r}$ . Ενδεικτικά οι τιμές για τις παραμέτρους που υπεισέρχονται στην παραπάνω εξίσωση είναι  $B_0 = 3.5 \times 10^{-5}T$ ,  $r \sim 2R_E$ , και  $R_E$  η ακτίνα της γης. Στις αποστάσεις αυτές, τυπικές ενέργειες για τα κινούμενα ηλεκτρόνια είναι  $\sim 1$  MeV που αντιστοιχούν σε ταχύτητες  $v/c = \sqrt{E^2 - m_0^2}/E \approx \sqrt{1 - 0.512^2}/1 = 0.86$ . Διαλέγοντας το σύστημα αξόνων έτσι ώστε  $\hat{d} = \hat{z}$  και μετρώντας τις αποστάσεις σε μονάδες<sup>7</sup>  $R_E$ , παίρνουμε:

$$\begin{aligned} B_x &= B_0 \frac{3xz}{r^5} \\ B_y &= B_0 \frac{3yz}{r^5} \\ B_z &= B_0 \left( \frac{3zz}{r^5} - \frac{1}{r^3} \right) \end{aligned} \quad (6.11)$$

Το πεδίο δύναμης προγραμματίζεται τώρα εύκολα στο αρχείο `sr_Bd.f90`:

<sup>7</sup>Αφού  $c = 1$ , αυτό σημαίνει ότι η μονάδα του χρόνου είναι ο χρόνος που χρειάζεται το φως στο κενό να διασχίσει απόσταση  $R_E$ .



Σχήμα 6.7: Η επίδραση της προσθήκης ενός ηλεκτρικού πεδίου  $q\vec{E}/m_0 = 1.0\hat{z}$  στην τροχιά του σχήματος 6.5

```

=====
!   Particle in Magnetic dipole field:
!   q B_1/m = k1 (3 x1 x3)/r^5
!   q B_2/m = k1 (3 x2 x3)/r^5
!   q B_3/m = k1[(3 x3 x3)/r^5-1/r^3]
=====
subroutine F(T,Y,YP)
  include 'sr.inc'
  real(8) :: t
  real(8) :: Y(*),YP(*)
  real(8) :: x1,x2,x3,v1,v2,v3,p1,p2,p3
  real(8) :: B1,B2,B3
  real(8) :: r
  x1 = Y(1);p1 = Y(4)
  x2 = Y(2);p2 = Y(5)
  x3 = Y(3);p3 = Y(6)
  call velocity(p1,p2,p3,v1,v2,v3)
!now we can use all x1,x2,x3,p1,p2,p3,v1,v2,v3
  YP(1) = v1
  YP(2) = v2
  YP(3) = v3
!Acceleration:
  r      = sqrt(x1*x1+x2*x2+x3*x3)

```

```

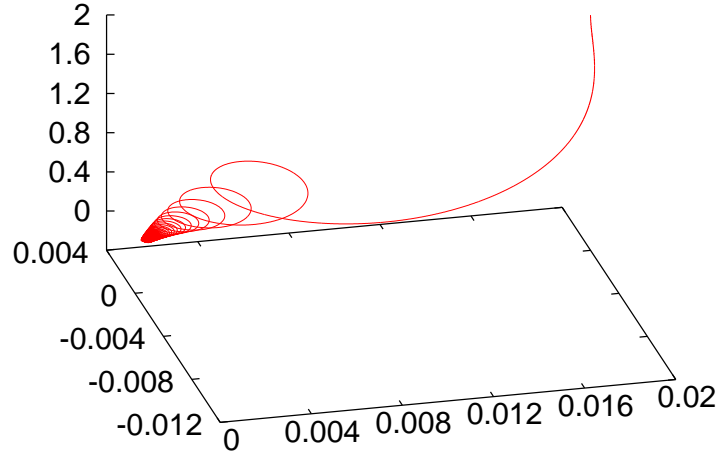
if( r.gt.0.0D0)then
  B1    = k1*( 3.0D0*x1*x3)/r**5
  B2    = k1*( 3.0D0*x2*x3)/r**5
  B3    = k1*((3.0D0*x3*x3)/r**5-1/r**3)
  YP(4) = v2*B3-v3*B2
  YP(5) = v3*B1-v1*B3
  YP(6) = v1*B2-v2*B1
else
  YP(4) = 0.0D0
  YP(5) = 0.0D0
  YP(6) = 0.0D0
endif
end subroutine F
!=====
!Energy per unit rest mass
!=====
real(8) function energy(T,Y)
include 'sr.inc'
real(8) :: t,e
real(8) :: Y(*)
real(8) :: x1,x2,x3,v1,v2,v3,p1,p2,p3,psq
x1 = Y(1);p1 = Y(4)
x2 = Y(2);p2 = Y(5)
x3 = Y(3);p3 = Y(6)
psq= p1*p1+p2*p2+p3*p3
!Kinetic Energy/m_0
e = sqrt(1.0D0+psq)-1.0D0
energy = e
end function energy

```

Τα αποτελέσματα φαίνονται στο σχήμα 6.8 στο οποίο έχουμε υπερβάλλει τις παραμέτρους, ώστε να έχουμε ένα κατατοπιστικό οπτικό αποτέλεσμα. Στην πραγματικότητα, τα ηλεκτρόνια διαγράφουν πολύ λεπτές σπείρες και ο αναγνώστης ενθαρρύνεται να μελετήσει αριθμητικά το πρόβλημα για φαινομενολογικά ρεαλιστικές τιμές των παραμέτρων  $\vec{v}_0$ ,  $B_0$ ,  $\vec{r}_0$  και να κατανοήσει γιατί το φαινόμενο συμβαίνει μόνο κοντά στους πόλους.

## 6.4 Ασκήσεις

- 6.1 Να μελετήσετε την κίνηση σωματιδίου κοντά στην επιφάνεια της γης, όταν το μέτρο της αντίστασης του αέρα είναι ανάλογο του τετραγώνου του μέτρου της ταχύτητάς του.
- 6.2 Δύο φορτία κινούνται με μη σχετικιστικές ταχύτητες μέσα σε σταθερό μαγνητικό πεδίο  $\vec{B} = B\hat{z}$  και αλληλεπιδρούν μεταξύ τους



Σχήμα 6.8: Κίνηση φορτισμένου σωματίου σε μαγνητικό πεδίο διπόλου που δίνεται από τη σχέση (6.11). Για να έχουμε καλύτερα οπτικά αποτελέσματα πήραμε  $B_0 = 1000$ ,  $\vec{r} = 0.02\hat{x} + 2.00\hat{z}$ ,  $\vec{v} = -0.99999\hat{z}$ . Η ολοκλήρωση έγινε με 10000 βήματα από χρόνο  $t_0 = 0$  έως  $t_f = 5$ .

μόνο με ηλεκτροστατικές δυνάμεις Coulomb (οι υπόλοιπες αλληλεπιδράσεις μπορούν να αγνοηθούν). Να γράψετε πρόγραμμα που να υπολογίζει τις τροχιές των σωματιδίων δεδομένων των αρχικών συνθηκών χρησιμοποιώντας τη μέθοδο RK45.

- 6.3 Γράψτε πρόγραμμα για τον ανισοτροπικό αρμονικό ταλαντωτή  $\vec{F} = -k_x x \hat{x} - k_y y \hat{y} - k_z z \hat{z}$  και μελετήστε τις τρισδιάστατες καμπύλες Lissajous που εμφανίζονται για κατάλληλες τιμές των κυκλικών συχνοτήτων  $\omega_x = \sqrt{k_x/m}$ ,  $\omega_y = \sqrt{k_y/m}$ ,  $\omega_z = \sqrt{k_z/m}$ .
- 6.4 Δύο σωματίδια μάζας  $M$  βρίσκονται στερεωμένα στις θέσεις  $\vec{r}_1 = a\hat{z}$  και  $\vec{r}_2 = -a\hat{z}$ . Τρίτο σωματίδιο μάζας  $m$  αλληλεπιδρά μαζί τους με Νευτώνεια βαρυτική δύναμη και κινείται με μη σχετικιστικές ταχύτητες. Μελετήστε τις τροχιές που διαγράφει το σωματίδιο αυτό και εξετάστε αν υπάρχουν κατάλληλες αρχικές συνθήκες τέτοιες, ώστε η κίνηση του σωματιδίου να περιορίζεται στο επίπεδο.
- 6.5 Λύστε το πρόβλημα 5.19 της σελίδας 303 χρησιμοποιώντας τη μέθοδο RK45. Επιλέξτε αρχικές συνθήκες τέτοιες, ώστε το σύστημα



να εκτελεί μόνο μεταφορική κίνηση. Στη συνέχεια επιλέξτε αρχικές συνθήκες έτσι ώστε, για μικρές ταλαντώσεις, το κέντρο μάζας να παραμένει ακίνητο. Βρείτε τους κανονικούς τρόπους ταλάντωσης του συστήματος και βάλτε κατάλληλες αρχικές συνθήκες, ώστε το σύστημα να εκτελέσει κάθε έναν από αυτούς.

- 6.6 Λύστε το προηγούμενο πρόβλημα βάζοντας το σύστημα σε ένα κουτί  $|x| \leq L$  και  $|y| \leq L$ .

Υπόδειξη: Δείτε το αρχείο `springL.f90`.

- 6.7 Λύστε το πρόβλημα 5.20 της σελίδας 303 χρησιμοποιώντας τη μέθοδο RK45.

- 6.8 Λύστε το πρόβλημα 5.21 της σελίδας 304 χρησιμοποιώντας τη μέθοδο RK45.

- 6.9 Το ηλεκτρικό πεδίο που δημιουργεί ηλεκτρικό δίπολο διπολικής ροπής  $\vec{p} = p\hat{z}$  στο χώρο δίνεται από τις σχέσεις:

$$\begin{aligned}\vec{E} &= E_\rho \hat{\rho} + E_z \hat{z} \\ E_\rho &= \frac{1}{4\pi\epsilon_0} \frac{3p \sin \theta \cos \theta}{r^3} \\ E_z &= \frac{1}{4\pi\epsilon_0} \frac{p(3 \cos^2 \theta - 1)}{r^3}\end{aligned}\quad (6.12)$$

όπου  $\rho = \sqrt{x^2 + y^2} = r \sin \theta$ ,  $E_x = E_\rho \cos \phi$ ,  $E_y = E_\rho \sin \phi$  και  $(r, \theta, \phi)$  οι πολικές συντεταγμένες του σημείου στο οποίο υπολογίζεται το ηλεκτρικό πεδίο. Υπολογίστε την τροχιά δοκιμαστικού φορτίου που κινείται μη σχετικιστικά μέσα στο παραπάνω πεδίο. Στη συνέχεια, μελετήστε τη σχετικιστική κίνηση του σωματιδίου. Υπολογίστε τις αποκλίσεις των τροχιών μεταξύ της σχετικιστικής και μη σχετικιστικής κίνησης για τις ίδιες αρχικές συνθήκες ως συνάρτηση του χρόνου, όταν η αρχική ταχύτητα έχει μέτρο  $0.01c, 0.1c, 0.5c, 0.9c$  αντίστοιχα (αγνοήστε φαινόμενα ακτινοβολίας λόγω επιτάχυνσης του φορτίου).

- 6.10 Ηλεκτρικό φορτίο κατανέμεται γραμμικά πάνω στον άξονα των  $z$  με θετική γραμμική πυκνότητα φορτίου  $\lambda$ . Το ηλεκτρικό πεδίο δίνεται από τη σχέση

$$\vec{E} = E_\rho \hat{\rho} = \frac{1}{4\pi\epsilon_0} \frac{2\lambda}{\rho} \hat{\rho}$$

Υπολογίστε τις τροχιές δύο ίδιων δοκιμαστικών αρνητικών φορτίων που κινούνται μη σχετικιστικά μέσα στο παραπάνω πεδίο. Μπορείτε να θεωρήσετε ότι τα δύο φορτία αλληλεπιδρούν μόνο με ηλεκτροστατικές δυνάμεις Coulomb.

- 6.11 Ηλεκτρικό φορτίο κατανέμεται γραμμικά πάνω σε τέσσερις ευθείες που είναι παράλληλες στον άξονα των  $z$  με σταθερή γραμμική πυκνότητα φορτίου  $\lambda$ . Οι ευθείες αυτές τέμνουν το επίπεδο  $xy$  στα σημεία  $(0, 0)$ ,  $(0, a)$ ,  $(a, 0)$ ,  $(a, a)$ . Υπολογίστε την τροχιά δοκιμαστικού φορτίου που κινείται μη σχετικιστικά μέσα στο παραπάνω πεδίο. Στη συνέχεια μελετήστε τη σχετικιστική κίνηση του σωματιδίου (αγνοήστε φαινόμενα ακτινοβολίας λόγω επιτάχυνσης του φορτίου).
- 6.12 Τρία σωματίδια μάζας  $m$  αλληλεπιδρούν μεταξύ τους μόνο με τη δύναμη της Νευτώνειας βαρύτητας. Μελετήστε τη μη σχετικιστική κίνησή τους στο χώρο.

# ΚΕΦΑΛΑΙΟ 7

## Ηλεκτροστατική

Στο κεφάλαιο αυτό θα μελετήσουμε αριθμητικά το ηλεκτροστατικό πεδίο στατικής κατανομής φορτίων. Στην πρώτη παράγραφο μελετάμε τις δυναμικές γραμμές και ισοδυναμικές επιφάνειες κατανομής σημειακών ηλεκτρικών φορτίων στο επίπεδο. Στη δεύτερη παράγραφο εξετάζονται προβλήματα συνεχών κατανομών φορτίου, πάλι στο επίπεδο. Γίνεται αριθμητική λύση προβλημάτων συνοριακών τιμών χρησιμοποιώντας μεθόδους τύπου (over)relaxation.

### 7.1 Σημειακή Κατανομή Φορτίων

Έστω  $N$  σημειακά ηλεκτρικά φορτία  $Q_i$  τα οποία βρίσκονται σε σταθερές θέσεις στο επίπεδο που δίνονται από τα διανύσματα θέσης τους  $\vec{r}_i$ ,  $i = 1, \dots, N$ . Ο νόμος του Coulomb μας δίνει την τιμή του ηλεκτρικού πεδίου

$$\vec{E}(\vec{r}) = \frac{1}{4\pi\epsilon_0} \sum_{i=1}^N \frac{Q_i}{|\vec{r} - \vec{r}_i|^2} \hat{\rho}_i, \quad (7.1)$$

όπου το  $\hat{\rho}_i = (\vec{r} - \vec{r}_i)/|\vec{r} - \vec{r}_i|$  είναι το μοναδιαίο διάνυσμα στη διεύθυνση του  $\vec{r} - \vec{r}_i$ . Οι συνιστώσες του πεδίου είναι

$$\begin{aligned} E_x(x, y) &= \frac{1}{4\pi\epsilon_0} \sum_{i=1}^N \frac{Q_i(x - x_i)}{((x - x_i)^2 + (y - y_i)^2)^{3/2}} \\ E_y(x, y) &= \frac{1}{4\pi\epsilon_0} \sum_{i=1}^N \frac{Q_i(y - y_i)}{((x - x_i)^2 + (y - y_i)^2)^{3/2}}, \end{aligned} \quad (7.2)$$

Το ηλεκτροστατικό δυναμικό στη θέση  $\vec{r}$  είναι

$$V(\vec{r}) = V(x, y) = \frac{1}{4\pi\epsilon_0} \sum_{i=1}^N \frac{Q_i}{((x - x_i)^2 + (y - y_i)^2)^{1/2}}, \quad (7.3)$$

και ισχύει

$$\vec{E}(\vec{r}) = -\vec{\nabla}V(\vec{r}). \quad (7.4)$$

Οι δυναμικές γραμμές είναι οι ολοκληρωτικές καμπύλες του διανυσματικού πεδίου  $\vec{E}$ , δηλ. οι καμπύλες εκείνες πάνω στις οποίες εφάπτεται το διάνυσμα του ηλεκτρικού πεδίου σε κάθε σημείο τους. Μια σύμβαση που ακολουθείται στο σχεδιασμό των δυναμικών γραμμών είναι ότι η πυκνότητα τους ανά μονάδα επιφανείας είναι ανάλογη του μέτρου του  $\vec{E}$ . Δηλ. ο αριθμός των δυναμικών γραμμών που τέμνει μια επιφάνεια  $S$  είναι ανάλογη της ροής  $\Phi_E = \int_S \vec{E} \cdot d\vec{A}$  του ηλεκτρικού πεδίου που τη διαπερνά.

Οι ισοδυναμικές επιφάνειες είναι ο γεωμετρικός τόπος των σημείων εκείνων όπου η συνάρτηση του δυναμικού έχει σταθερή τιμή. Η σχέση (7.4) μας λέει πως πυκνές ισοδυναμικές επιφάνειες (που σημαίνουν γρήγορη χωρική μεταβολή του δυναμικού) συνεπάγονται ισχυρό ηλεκτρικό πεδίο στην περιοχή και αντίστροφα. Επίσης, μας λέει ότι η διεύθυνση του ηλεκτρικού πεδίου είναι κάθετη στις ισοδυναμικές επιφάνειες σε κάθε σημείο<sup>1</sup> (η διεύθυνση ταχύτερης μεταβολής του  $V$ ) και με φορά αυτή της μείωσης του δυναμικού. Όταν περιοριζόμαστε στο επίπεδο που βρίσκεται η κατανομή των φορτίων, οι ισοδυναμικές επιφάνειες δίνουν κλειστές καμπύλες που είναι η τομή τους με το εν λόγω επίπεδο.

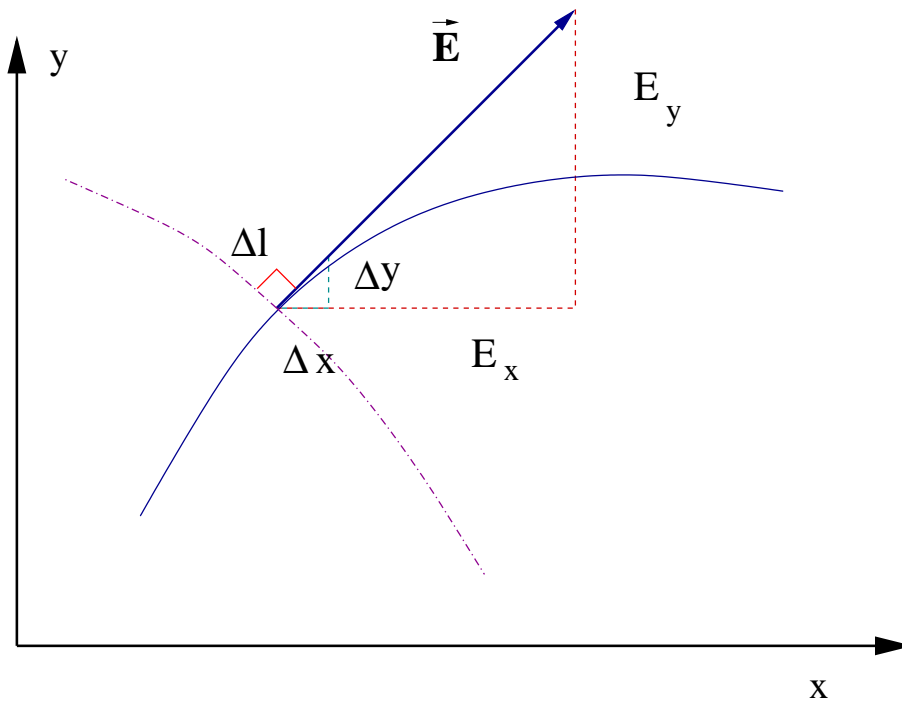
Στον υπολογισμό δεν μπορούμε να λύσουμε το πρόβλημα στο συνεχές. Η συνεχής καμπύλη που θέλουμε να υπολογίσουμε θα προσεγγιστεί με ένα μεγάλο αλλά πεπερασμένο αριθμό από μικρά ευθύγραμμα τμήματα. Η ιδέα περιγράφεται στο σχήμα 7.1: Το μικρό ευθύγραμμο τμήμα μήκους  $\Delta l$  είναι στη διεύθυνση του ηλεκτρικού πεδίου, οπότε από τα όμοια τρίγωνα παίρνουμε

$$\Delta x = \Delta l \frac{E_x}{E} \quad \Delta y = \Delta l \frac{E_y}{E}, \quad (7.5)$$

όπου  $E \equiv |\vec{E}| = \sqrt{E_x^2 + E_y^2}$ .

Για τον υπολογισμό των ισοδυναμικών καμπύλων χρησιμοποιούμε την ιδιότητα που έχουν οι δυναμικές γραμμές να είναι παντού κάθετες

<sup>1</sup>Επειδή σε κάθε μικρή μετατόπιση  $d\vec{r}$  πάνω σε μια ισοδυναμική επιφάνεια το δυναμικό παραμένει σταθερό ( $dV = 0$ ), έχουμε ότι  $0 = dV = \vec{\nabla}V \cdot d\vec{r} = -\vec{E} \cdot d\vec{r}$ , άρα  $\vec{E} \perp d\vec{r}$ .



Σχήμα 7.1: Σε κάθε σημείο του χώρου οι δυναμικές γραμμές εφάπτονται με το διάνυσμα της έντασης του ηλ. πεδίου, ενώ οι ισοδυναμικές επιφάνειες (εδώ καμπύλες) το έχουν κάθετο. Προσεγγίζοντας τη συνεχή καμπύλη με το ευθ. τμήμα  $\Delta l$  έχουμε  $\Delta y / \Delta x = E_y / E_x$ .

στις ισοδυναμικές επιφάνειες (εδώ στις καμπύλες). Άρα αν  $(\Delta x, \Delta y)$  δίνει την εφαπτόμενη στην ισοδυναμική γραμμή, τότε το  $(-\Delta y, \Delta x)$  είναι σε κάθετη κατεύθυνση, αφού  $(\Delta x, \Delta y) \cdot (-\Delta y, \Delta x) = -\Delta x \Delta y + \Delta y \Delta x = 0$ . Οπότε για τις ισοδυναμικές καμπύλες του επιπέδου προκύπτει η εξίσωση

$$\Delta x = -\Delta l \frac{E_y}{E}, \quad \Delta y = \Delta l \frac{E_x}{E}. \quad (7.6)$$

Μπορούμε τώρα να σχεδιάσουμε μια αλγοριθμική διαδικασία που θα μας επιτρέψει τον προσεγγιστικό υπολογισμό των δυναμικών και ισοδυναμικών καμπύλων: Επιλέγουμε αρχικό σημείο από το οποίο περνάει η (μοναδική) δυναμική γραμμή ή ισοδυναμική επιφάνεια που επιθυμούμε. Από την κατανομή των φορτίων υπολογίζουμε το ηλεκτρικό πεδίο χρησιμοποιώντας τις σχέσεις (7.2). Επιλέγοντας αρκετά μικρό βήμα  $\Delta l$ , μετακινούμαστε στη διεύθυνση  $(\Delta x, \Delta y)$

$$x \rightarrow x + \Delta x \quad y \rightarrow y + \Delta y, \quad (7.7)$$

χρησιμοποιώντας τις σχέσεις (7.5) ή (7.6) ανάλογα με την περίπτωση.

Επαναλαμβάνουμε τη διαδικασία μέχρι να τελειώσει ο σχεδιασμός για ένα μεγάλο αριθμό βημάτων. Το κριτήριο για αυτό θα το καθορίσει ο προγραμματιστής ή ο χρήστης ανάλογα με τις ανάγκες του υπολογισμού, λ.χ. η δυναμική γραμμή φεύγει έξω από τα όρια σχεδιασμού ή πλησιάζει κοντύτερα σε ένα φορτίο από μια ελάχιστη απόσταση.

## 7.2 Το Πρόγραμμα – Ορεκτικά και ... επιδόρπιο

Ο βιαστικός, αλλά και ελαφρά καταρτισμένος αναγνώστης μπορεί να συνεχίσει τη μελέτη του πηγαίνοντας κατευθείαν στην παράγραφο 7.4 και να επιστρέψει εδώ αργότερα για διευκρινίσεις. Εκεί θα βρει την τελική και πλήρη μορφή του προγράμματος, καθώς και συνοπτικές οδηγίες χρήσης του.

Για τον προγραμματισμό του αλγόριθμου που περιγράψαμε στην προηγούμενη παράγραφο, θα χωρίσουμε τις διαδικασίες σε τέσσερις ξεχωριστές και ανεξάρτητες μεταξύ τους ομάδες οι οποίες έχουν καλά ορισμένους στόχους και μπορούν να χρησιμοποιηθούν σε διαφορετικά τμήματα ενός προγράμματος.

- Το κυρίως πρόγραμμα: Ορίζεται η δομή των δεδομένων, που αποτελείται κυρίως από τις θέσεις των φορτίων που αποθηκεύονται στα arrays  $X(P)$ ,  $Y(P)$  και την τιμή των φορτίων  $Q(P)$ . Διεπαφή (interface) με το χρήστη: Γίνεται από το χρήστη εισαγωγή δεδομένων, όπως ο αριθμός των φορτίων  $N$ , η θέση τους και το μέγεθός τους, και γίνεται η επεξεργασία των αποτελεσμάτων.
- subroutine `eline(xin,yin,X,Y,Q,N)`: Υπολογίζει τη δυναμική γραμμή που περνάει από το σημείο `xin,yin`. Στην είσοδο ο χρήστης δίνει το σημείο `xin,yin` και τα δεδομένα  $N$ ,  $X(N)$ ,  $Y(N)$ ,  $Q(N)$ . Στην έξοδο η υπορουτίνα τυπώνει στην καθιερωμένη έξοδο (standard output - που είναι συνήθως η οθόνη) τις συντεταγμένες της δυναμικής γραμμής που περνάει από το σημείο `(xin,yin)` και προεκτείνεται μέχρι να φτάσει πολύ κοντά σε κάποιο άλλο φορτίο ή μέχρι να βγει εκτός της περιοχής σχεδίασης (εδώ ορίζεται να εκτείνεται μέχρι μια μέγιστη απόσταση από την αρχή των αξόνων). Καλεί την υπορουτίνα `efield` για τον υπολογισμό του ηλεκτρικού πεδίου και την `mdist` για τον υπολογισμό της ελάχιστης απόστασης των σημείων της δυναμικής γραμμής από τα ηλεκτρικά φορτία.

- subroutine `epotline(xin,yin,X,Y,Q,N)`: Υπολογίζει την ισοδυναμική καμπύλη που περνάει από το σημείο `xin,yin`. Στην είσοδο ο χρήστης δίνει το σημείο `(xin,yin)` και τα δεδομένα `N, X(N), Y(N), Q(N)`. Στην έξοδο η υπορουτίνα τυπώνει στην καθιερωμένη έξοδο τις συντεταγμένες της ισοδυναμικής καμπύλης που περνάει από το σημείο `(xin,yin)` και προεκτείνεται μέχρι είτε να κλείσει αρκετά κοντά στο αρχικό σημείο<sup>2</sup> ή μέχρι να βγει εκτός της περιοχής σχεδίασης. Καλεί την υπορουτίνα `efield` για τον υπολογισμό του ηλεκτρικού πεδίου και την `mdist` για τον υπολογισμό της ελάχιστης απόστασης από τα ηλεκτρικά φορτία.
- subroutine `efield(x0,y0,X,Y,Q,N,Ex,Ey)`: Καλείται από τις παραπάνω ρουτίνες. Υπολογίζει το ηλεκτρικό πεδίο  $(E_x, E_y)$  στη θέση  $(x_0, y_0)$ . Στην είσοδο ο χρήστης παρέχει τον αριθμό `N`, τις θέσεις των φορτίων που αποθηκεύονται στα arrays `X(N), Y(N)`, το μέγεθος των φορτίων που αποθηκεύονται array `Q(N)` και τη θέση `x0, y0` στην οποία θα υπολογιστεί το ηλεκτρικό πεδίο. Στην έξοδο ο χρήστης παίρνει τις συνιστώσες του ηλεκτρικού πεδίου  $E_x, E_y$ .
- subroutine `mdist(x0,y0,X,Y,N,rmin,rmax)`: Καλείται από τις παραπάνω ρουτίνες. Υπολογίζει την ελάχιστη και μέγιστη απόσταση του σημείου `x0, y0` από τα φορτία που βρίσκονται στις θέσεις `X(N), Y(N)`. Στην είσοδο ο χρήστης παρέχει τον αριθμό των φορτίων `N`, τις θέσεις τους `X(N), Y(N)` και το σημείο `x0, y0`. Στην έξοδο η ρουτίνα δίνει την ελάχιστη και μέγιστη απόσταση `rmin,rmax`.

Στο κυρίως πρόγραμμα ο χρήστης πρέπει να ορίσει τις μεταβλητές `N, X(N), Y(N), Q(N)`. Το πρόγραμμα θα πρέπει να ζητήσει από το χρήστη τις σχετικές πληροφορίες (“διεπαφή με χρήστη”) και, χρησιμοποιώντας τις, να καλέσει τις σχετικές ρουτίνες που υλοποιούν τον υπολογισμό. Μια μινιμαλιστική προσέγγιση θα είναι να γράψουμε σε ένα αρχείο `ELines.f90` ένα πρόγραμμα όπως το παρακάτω (“πρώτη έκδοση” - version 1):

```
! *****
program Electric_Fields
! *****
implicit none
integer ,parameter :: P=20      !max number of charges
real ,dimension(P) :: X,Y,Q
```

<sup>2</sup>Θυμάστε φαντάζομαι πως οι ισοδυναμικές επιφάνειες, άρα και οι καμπύλες, είναι κλειστές.

```

integer                :: N
!----- SET CHARGE DISTRIBUTION -----
N      =  2
X(1)   =  1.0
Y(1)   =  0.0
Q(1)   =  1.0
X(2)   = -1.0
Y(2)   =  0.0
Q(2)   = -1.0
!----- DRAWING LINES -----
call eline(0.0, 0.5,X,Y,Q,N)
call eline(0.0, 1.0,X,Y,Q,N)
call eline(0.0, 1.5,X,Y,Q,N)
call eline(0.0, 2.0,X,Y,Q,N)
call eline(0.0,-0.5,X,Y,Q,N)
call eline(0.0,-1.0,X,Y,Q,N)
call eline(0.0,-1.5,X,Y,Q,N)
call eline(0.0,-2.0,X,Y,Q,N)
end program Electric_Fields

```

Οι εντολές

```

!----- SET CHARGE DISTRIBUTION -----
N      =  2
X(1)   =  1.0
Y(1)   =  0.0
Q(1)   =  1.0
X(2)   = -1.0
Y(2)   =  0.0
Q(2)   = -1.0

```

τοποθετούν δύο ίσα και αντίθετα φορτία  $Q(1) = -Q(2) = 1.0$  στις θέσεις  $(1, 0)$  και  $(-1, 0)$  αντίστοιχα. Οι επόμενες γραμμές καλούν την υπορουτίνα `eline` να κάνει τον υπολογισμό για 8 δυναμικές γραμμές που περνούν από τα σημεία  $(0, \pm 1/2)$ ,  $(0, \pm 1)$ ,  $(0, \pm 3/2)$ ,  $(0, \pm 2)$ :

```

!----- DRAWING LINES -----
call eline(0.0, 0.5,X,Y,Q,N)
call eline(0.0, 1.0,X,Y,Q,N)
call eline(0.0, 1.5,X,Y,Q,N)
call eline(0.0, 2.0,X,Y,Q,N)
call eline(0.0,-0.5,X,Y,Q,N)
call eline(0.0,-1.0,X,Y,Q,N)
call eline(0.0,-1.5,X,Y,Q,N)
call eline(0.0,-2.0,X,Y,Q,N)

```



Οι εντολές αυτές τυπώνουν τις συντεταγμένες των σημείων των δυναμικών γραμμών στο standard output, τις οποίες ο χρήστης θα πρέπει να επεξεργαστεί περαιτέρω.

Για τον υπολογισμό των ισοδυναμικών καμπύλων ο κώδικας είναι ακριβώς ο ίδιος αντικαθιστώντας `call eline` → `call epotline`.

Για να συμπληρωθεί το πρόγραμμα θα πρέπει να προγραμματίσουμε τις υπορουτίνες `eline`, `efield`, `mdist`. Αυτό θα το συζητήσουμε παρακάτω. Για την ώρα, αν θέλετε να χρησιμοποιήσετε το πλήρες πρόγραμμα, αναζητήστε το αρχείο `ELines.f90` από το συνοδευτικό λογισμικό. Μεταγλωττίζουμε και τρέχουμε το πρόγραμμα κατά τα γνωστά:

```
> gfortran ELines.f90 -o el
> ./el > el.out
```

όπου με το χαρακτήρα `>` επαναορισμού του standard output μεταφέρουμε τα αποτελέσματα που τυπώνει το πρόγραμμα στο αρχείο `el.out`. Για να δούμε τα αποτελέσματα, χρησιμοποιούμε το `gnuplot`:

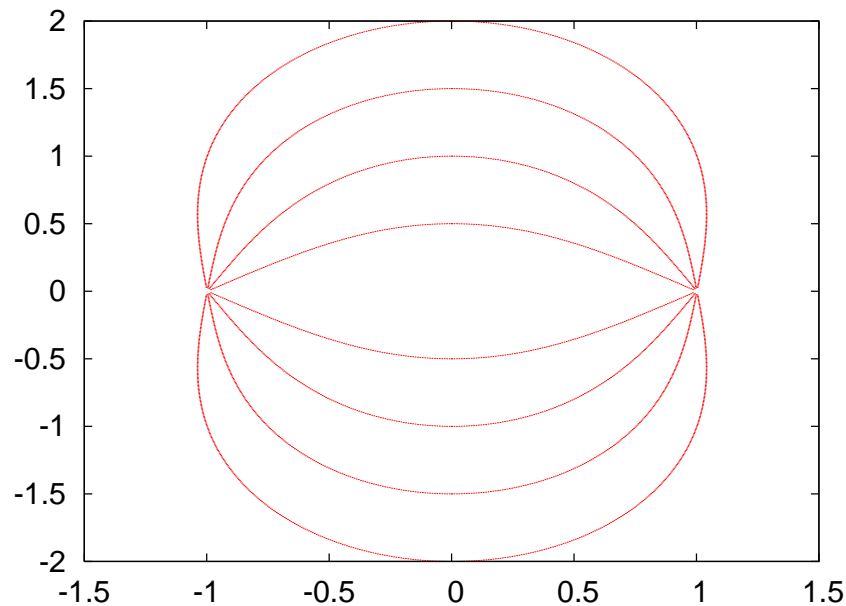
```
gnuplot> plot "el.out" with dots
```

Το αποτέλεσμα φαίνεται στο σχήμα 7.2.

Ας γράψουμε τώρα το πρόγραμμα δίνοντας στο χρήστη την ευκαιρία να μελετήσει ευκολότερα διαφορετικές κατανομές φορτίων και να επιλέξει τις δυναμικές γραμμές που θα υπολογιστούν. Ο χρήστης θα παρέχει διαδραστικά τον αριθμό και τη θέση/μέγεθος των ηλεκτρικών φορτίων καθώς και τον αριθμό και τα αρχικά σημεία των δυναμικών γραμμών που θα υπολογιστούν. Ο πρώτος στόχος επιτυγχάνεται αλλάζοντας τον κώδικα στο σημείο που θέτει την κατανομή φορτίων ως εξής:

```
!----- SET CHARGE DISTRIBUTION -----
print *, '# Enter number of charges:'
read *, N
print *, '# N= ',N
do i=1,N
  print *, '# Charge: ',i
  print *, '# Position and charge: (X,Y,Q):'
  read *, X(i),Y(i),Q(i)
  print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
enddo
```

Η πρώτη εντολή ζητάει από το χρήστη τον αριθμό των φορτίων της κατανομής, τον οποίο διαβάζει από το standard input. Στη συνέχεια,



Σχήμα 7.2: Μερικές δυναμικές γραμμές ηλ. πεδίου δύο αντίθετων φορτίων που σχεδιάζουμε με το πρόγραμμα ELines.f90 στην ... πρώτη του έκδοση.

για κάθε τιμή του  $i$  διαβάζουμε τις τιμές της θέσης/μέγεθος φορτίου και τις αποθηκεύουμε στις αντίστοιχες θέσεις των arrays  $X(i)$ ,  $Y(i)$ ,  $Q(i)$ . Τα αποτελέσματα τυπώνονται για να τα βλέπει ο χρήστης και για να ελέγχει ότι πέρασαν σωστά στη μνήμη του υπολογιστή.

Ο σχεδιασμός των δυναμικών γραμμών γίνεται αλλάζοντας τον κώδικα στο σημείο σχεδίασης των γραμμών ως εξής:

```
!----- DRAWING LINES -----
print *, '# How many lines to draw? '
read  *, draw
do i=1,draw
  print *, '# Initial point (x0,y0): '
  read  *, x0,y0
  call  eline(x0,y0,X,Y,Q,N)
enddo
```

Το πρόγραμμα το τρέχουμε, όπως φαίνεται παρακάτω, για ένα φορτίο  $q = 1.0$  στην αρχή των αξόνων ζητώντας να σχεδιάζει μία δυναμική γραμμή που περνάει από το σημείο  $(0.1, 0.1)$ .

```

> gfortran ELines.f90 -o el
> ./el
# Enter number of charges:
1
# N=          1
# Charge:          1
# Position and charge: (X,Y,Q):
0.0 0.0 1.0
# (X,Y)=      0.000000      0.000000      Q=      1.000000
# How many lines to draw?
1
# Initial point (x0,y0):
0.1 0.1
  9.2928931E-02  9.2928931E-02
  8.5857861E-02  8.5857861E-02
  7.8786790E-02  7.8786790E-02
....

```

Για μεγαλύτερες κατανομές φορτίων χρησιμοποιείτε έναν editor και σε ένα αρχείο με όνομα λ.χ. Input γράψτε τα δεδομένα εισόδου

2	N: Number of Charges
1.0 0.0 1.0	(X,Y,Q): Position and charge
-1.0 0.0 -1.0	(X,Y,Q): Position and charge
8	Number of lines to draw
0.0 0.5	x0,y0: Initial point of line
0.0 1.0	x0,y0: Initial point of line
0.0 1.5	x0,y0: Initial point of line
0.0 2.0	x0,y0: Initial point of line
0.0 -0.5	x0,y0: Initial point of line
0.0 -1.0	x0,y0: Initial point of line
0.0 -1.5	x0,y0: Initial point of line
0.0 -2.0	x0,y0: Initial point of line

Αν δώσετε τώρα την εντολή <sup>3</sup>

```
./el < Input > el.out
```

θα πάρετε τα αποτελέσματα για τις δυναμικές γραμμές που ζητήσατε να υπολογιστούν και τα οποία θα αποθηκευτούν στο αρχείο el.out. Μπορείτε έτσι να φτιάξετε τη δικιά σας “βιβλιοθήκη” από κατανομές φορτίων και δυναμικές γραμμές του ηλεκτρικού πεδίου που δημιουργούν. Ο παραπάνω κώδικας (“δεύτερη έκδοση” - version 2) παρατίθεται παρακάτω:

<sup>3</sup>To “< Input” επαναορίζει το standard input να είναι το αρχείο με όνομα Input.

```

!*****
program Electric_Fields
!*****
implicit none
integer,parameter :: P=20      !max number of charges
real,dimension(P) :: X,Y,Q
integer            :: N
integer            :: i,j,draw
real               :: x0,y0
!----- SET CHARGE DISTRIBUTION -----
print *, '# Enter number of charges: '
read  *, N
print *, '# N= ',N
do i=1,N
  print *, '# Charge: ',i
  print *, '# Position and charge: (X,Y,Q): '
  read  *, X(i),Y(i),Q(i)
  print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
enddo
!----- DRAWING LINES -----
print *, '# How many lines to draw? '
read  *, draw
do i=1,draw
  print *, '# Initial point (x0,y0): '
  read  *, x0,y0
  call eline(x0,y0,X,Y,Q,N)
enddo
end program Electric_Fields

```

Ήδη ο προσεκτικός αναγνώστης θα έχει ασκηθεί αρκετά, ώστε να καταλάβει πως για να φτιάξει μια ωραία εικόνα από αντιπροσωπευτικές δυναμικές γραμμές χρειάζεται αρκετό χρόνο για να το κάνει... Πώς θα μπορούσαμε, τουλάχιστον αρχικά, να πάρουμε γρήγορα μια αντιπροσωπευτική εικόνα του ηλεκτρικού πεδίου; Για τις δυναμικές γραμμές η απάντηση είναι εύκολη: Αρκετά κοντά σε ένα σημειακό ηλεκτρικό φορτίο, το ηλεκτρικό πεδίο είναι κατά πολύ καλή προσέγγιση ισοτροπικά ακτινικό. Ο αριθμός των δυναμικών γραμμών που ξεκινούν/καταλήγουν από/σε ένα σημειακό θετικό/αρνητικό ηλεκτρικό φορτίο είναι ανάλογος του μεγέθους του φορτίου. Έτσι, αρκεί να επιλέξουμε ως αρχικά σημεία έναν αριθμό ανάλογο του φορτίου από ισότροπα κατανομημένα σημεία πάνω σε ένα αρκετά μικρό κυκλάκι γύρω από κάθε φορτίο της κατανομής. Παρακάτω παραθέτουμε τον κώδικα (“τρίτη έκδοση” - version 3) για φορτία ίσα σε μέγεθος και αφήνουμε ως άσκηση στον αναγνώστη την περίπτωση φορτίων που είναι διαφορετικού μεγέθους:

```

!*****
program Electric_Fields
!*****
implicit none
integer,parameter:: P=20      !max number of charges
real,dimension(P):: X,Y,Q
integer           :: N
integer           :: i,j,nd
real              :: x0,y0,theta
real,parameter   :: PI= 3.14159265359
!----- SET CHARGE DISTRIBUTION -----
print *, '# Enter number of charges:'
read  *, N
print *, '# N= ',N
do i=1,N
  print *, '# Charge: ',i
  print *, '# Position and charge: (X,Y,Q):'
  read  *, X(i),Y(i),Q(i)
  print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
enddo
!----- DRAWING LINES -----
!We draw 2*nd field lines around each charge
nd      = 6
do i = 1,N
  do j = 1,(2*nd)
    theta = (PI/nd)*j
    x0     = X(i) + 0.1 * cos(theta)
    y0     = Y(i) + 0.1 * sin(theta)
    call   eline(x0,y0,X,Y,Q,N)
  enddo
enddo
end program Electric_Fields

```

Θα παρατηρήσατε ήδη πως η μόνη αλλαγή που κάναμε είναι στο κομμάτι σχεδιασμού των γραμμών. Θέτουμε τον αριθμό των αρχικών γραμμών γύρω από κάθε φορτίο να είναι 12 με την εντολή `nd = 6` και μετά γύρω από κάθε φορτίο καλούμε την `eline` με αρχικά σημεία  $(x_0, y_0)$  πάνω σε ένα κύκλο κέντρου  $(X(i), Y(i))$  και ακτίνας 0.1. Αυτό γίνεται  $2*nd$  φορές σε σημεία που καθορίζονται από τη γωνία  $\theta = (PI/nd)*j$ .

Για να το τρέξουμε, γράφουμε ένα αρχείο Input με περιεχόμενα την κατανομή των φορτίων. Για παράδειγμα, για μια κατανομή τεσσάρων φορτίων  $q_i = \pm 1$  πάνω στις κορυφές ενός τετραγώνου γράφουμε:

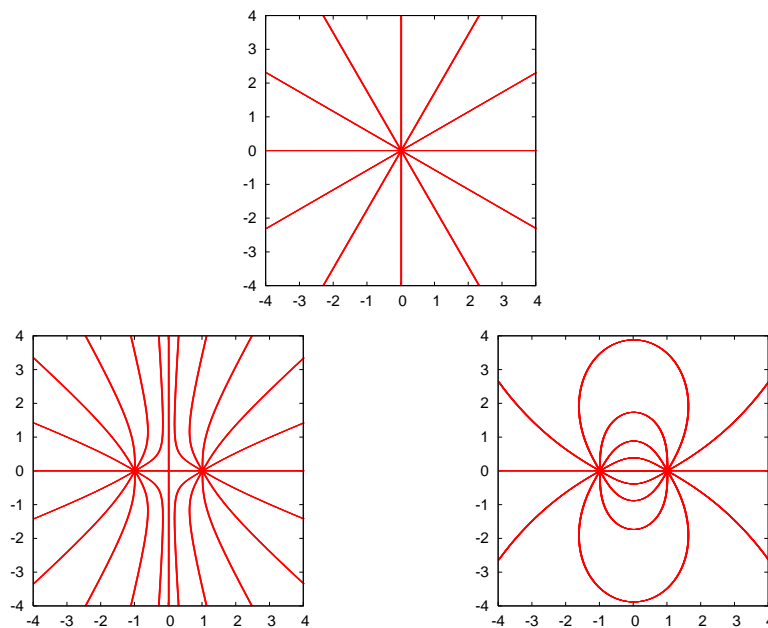
4			N: Number of charges
1	1	-1	(X,Y,Q): Position and charge
-1	1	1	(X,Y,Q): Position and charge

1	-1	1	(X,Y,Q): Position and charge
-1	-1	-1	(X,Y,Q): Position and charge

και μετά δίνουμε τις εντολές:

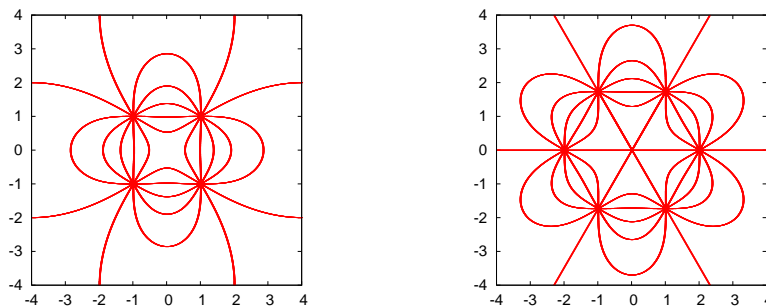
```
> gfortran ELines.f90 -o el
> ./el < Input > el.out
> gnuplot
gnuplot> plot "el.out" with dots
```

Στα σχήματα 7.3-7.4 δείχνουμε τα αποτελέσματα για μερικές κατανομές φορτίου. Αφήνουμε ως άσκηση στον αναγνώστη να βρει από ποιες κατανομές προκύπτουν και να αναπαράγει τα σχήματα αυτά για εξάσκηση.



Σχήμα 7.3: Δυναμικές γραμμές ηλεκτρικού πεδίου κατανομής σημειακών φορτίων που σχεδιάζουμε με το πρόγραμμα ELines.f90.

Με παρόμοιο τρόπο γράφουμε πρόγραμμα και για τις ισοδυναμικές καμπύλες. Η μόνη διαφορά είναι ότι αν θέλουμε τα αρχικά σημεία να επιλέγονται αυτόματα, θα πρέπει να επινοήσουμε ένα νέο αλγόριθμο που να ζωγραφίζει τις δυναμικές γραμμές για δεδομένες πτώσεις δυναμικού. Λόγω της πολυπλοκότητας του αλγόριθμου, αυτό αφήνεται ως άσκηση για τον αναγνώστη (δες σχετική άσκηση με οδηγίες). Εμείς



Σχήμα 7.4: Δυναμικές γραμμές ηλεκτρικού πεδίου κατανομής σημειακών φορτίων που σχεδιάζουμε με το πρόγραμμα ELines.f90.

στον παρακάτω κώδικα επιλέγουμε για αρχικά σημεία ισαπέχοντα σημεία πάνω σε ένα τετραγωνικό πλέγμα. Ο κώδικας αποθηκεύεται στο αρχείο EPotential.f90:

```
! *****
program Electric_Potential
! *****
implicit none
integer,parameter :: P=20      !max number of charges
real,dimension(P) :: X,Y,Q
integer :: N
real,parameter :: PI= 3.14159265359
integer :: i,j,nd
real :: x0,y0,rmin,rmax,L

print *, '# Enter number of charges:'
read *, N
print *, '# N= ',N
do i=1,N
  print *, '# Charge: ',i
  print *, '# Position and charge: (X,Y,Q):'
  read *, X(i),Y(i),Q(i)
  print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
enddo

!----- DRAWING LINES -----
!We draw lines passing through an equally
!spaced lattice of N=(2*nd+1)x(2*nd+1) points
!in the square -L<= x <= L, -L<= y <= L.
nd = 4
L = 1.0
do i = -nd,nd
  do j = -nd,nd
```

```

x0 = i*(L/nd)
y0 = j*(L/nd)
print *, '# @ ', i, j, L/nd, x0, y0
call mdist(x0, y0, X, Y, N, rmin, rmax)
!we avoid getting too close to a charge:
if(rmin .gt. L/(nd*10) )&
    call epotline(x0, y0, X, Y, Q, N)
enddo
enddo
end program Electric_Potential

```

Ο κώδικας είναι πανομοιότυπος με τον προηγούμενο στο πρώτο και δεύτερο μέρος του. Στο τρίτο μέρος, όπου ζητείται ο σχεδιασμός των καμπύλων, καλείται τώρα η υπορουτίνα `epotline` να κάνει το σχεδιασμό. Όλα τα υπόλοιπα αναφέρονται στον προσδιορισμό των αρχικών σημείων που γίνεται ως εξής: Επιλέγουμε τον αριθμό των σημείων του πλέγματος με την εντολή `nd=4`, η οποία δίνει  $(2*nd+1)*(2*nd+1) = 81$  σημεία. Με την εντολή `L=1.0` καθορίζουμε τα όρια του πλέγματος να είναι το τετράγωνο  $(1, 1)$ ,  $(-1, 1)$ ,  $(-1, -1)$ ,  $(1, -1)$ . Στη συνέχεια, για κάθε σημείο του πλέγματος  $(x0, y0)$  υπολογίζουμε την ισοδυναμική καμπύλη που περνάει από αυτό με την προϋπόθεση το σημείο αυτό να μην είναι πολύ κοντά σε ένα από τα φορτία. Αυτό γίνεται καλώντας την υπορουτίνα `mdist` από την οποία παίρνουμε την ελάχιστη απόσταση `rmin` του σημείου και βάζοντας κάτω όριο σε αυτή τον αριθμό `L/(nd*10)`. Για να το τρέξουμε δίνουμε τις εντολές:

```

> gfortran EPotential.f90 -o ep
> ./ep < Input > ep.out
> gnuplot
gnuplot> plot "ep.out" with dots

```

Μερικά από τα αποτελέσματα φαίνονται στο σχήμα 7.5.

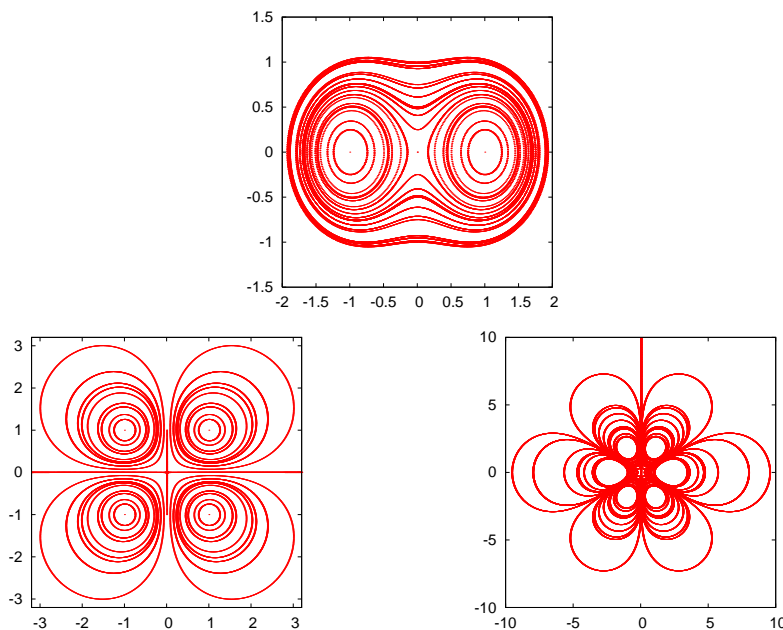
### 7.3 Το Πρόγραμμα - Το Κυρίως Πιάτο

Στην παράγραφο αυτή περιγράφουμε τη λειτουργία και τον προγραμματισμό των κύριων κομματιών του προγράμματος: Τις υπορουτίνες υπολογισμού των καμπύλων `eline`, `epotline`, του ηλεκτρικού πεδίου `efield` και της ελάχιστης απόστασης από τα φορτία `mdist`.

Αρχίζουμε από την ρουτίνα `eline`. Ο χρήστης την καλεί με την εντολή

```
call eline(x0, y0, X, Y, Q, N)
```





Σχήμα 7.5: Ισοδυναμικές καμπύλες ηλεκτρικού πεδίου κατανομής σημειακών φορτίων που σχεδιάζουμε με το πρόγραμμα EPotential.f90. Προσοχή όμως: η πυκνότητα των καμπύλων δεν αντιστοιχεί στο πραγματικό πεδίο, για να πάρετε τις σωστές εικόνες θα πρέπει να κάνετε την Άσκηση 7.5.

δίνοντας στην είσοδο το αρχικό σημείο της γραμμής  $(x_0, y_0)$ , τον αριθμό των φορτίων  $N$ , καθώς και τις θέσεις των φορτίων  $(X(N), Y(N))$  και μεγέθη των φορτίων  $Q(N)$ . Για να σχεδιαστεί η δυναμική γραμμή χρειάζονται το βήμα  $\Delta l$  της σχέσης (7.5), καθώς και τα όρια σχεδίασης των γραμμών. Τα όρια αυτά είναι δύο: Πρώτα όταν πλησιάζουμε πολύ κοντά σε ένα φορτίο. Τότε το ηλεκτρικό πεδίο τείνει στο άπειρο και αυτό θα δημιουργήσει προβλήματα. Το βήμα  $\Delta l$  είναι μια καλή κλίμακα μικρού μήκους και θέτουμε το όριο προσέγγισης σε  $2\Delta l$ . Επίσης, δε θέλουμε να απομακρυνθούμε πολύ από την κατανομή, οπότε θέτουμε μια αυθαίρετη μέγιστη απόσταση από όλα τα φορτία ίση με  $\max\_dist=20.0$ . Άλλες προβληματικές καταστάσεις που πρέπει να προβλέψουμε, είναι να μηδενιστεί το ηλεκτρικό πεδίο, οπότε το αποτέλεσμα του υπολογισμού στη Σχέση (7.5) γίνεται απροσδιόριστο. Τέλος, παίρνοντας  $\Delta l > 0$  μετακινούμαστε πάνω στη δυναμική γραμμή μόνο προς την κατεύθυνση του ηλεκτρικού πεδίου καταλήγοντας πάντα πάνω σε ένα αρνητικό φορτίο ή στη μέγιστη απόσταση (γιατί;). Για να σχεδιάσουμε τη δυναμική γραμμή ολόκληρη, θα επαναλάβουμε τον υπολογισμό από το ίδιο αρχικό σημείο με  $\Delta l < 0$ .

Η κωδικοποίηση της διαδικασίας φαίνεται παρακάτω:

```
! *****
subroutine eline(xin,yin,X,Y,Q,N)
! *****
implicit none
integer      :: N
real ,dimension(N) :: X,Y,Q
real         :: xin,yin,x0,y0
real ,parameter :: step=0.01
real ,parameter :: max_dist=20.0
integer      :: i,direction
real         :: rmin,rmax,r,dx,dy,d1
real         :: Ex,Ey,E
do direction = -1,1,2                !direction= +/- 1
  d1 = direction * step
  x0 = xin
  y0 = yin
  dx = 0.0
  dy = 0.0
  call mdist(x0,y0,X,Y,N,rmin,rmax)
  do while(rmin .gt. (2.0*step) .and. rmax .lt. max_dist)
    print *,x0,y0
    ! We evaluate the E-field at the midpoint: This reduces
    ! systematic errors
    call efield(x0+0.5*dx,y0+0.5*dy,X,Y,Q,N,Ex,Ey)
    E = sqrt(Ex*Ex+Ey*Ey)
    if( E .le. 1.0e-10 ) exit
    dx = d1*Ex/E
    dy = d1*Ey/E
    x0 = x0 + dx
    y0 = y0 + dy
    call mdist(x0,y0,X,Y,N,rmin,rmax)
  enddo                                !do while()
enddo                                  !do direction = -1,1,2
end subroutine eline
```

Στο πρώτο κομμάτι του κώδικα δηλώνονται οι μεταβλητές. Το μόνο καινούργιο που έχουμε να πούμε είναι η δήλωση

```
real ,dimension(N) :: X,Y,Q
```

αντί για dimension(P). Αυτό δεν πειράζει σε τίποτα, αρκεί φυσικά ο προγραμματιστής να έχει ήδη ελέγξει ότι  $N \leq P$ . Στο αρχικό πρόγραμμα που δηλώθηκαν τα arrays X,Y,Q, ζητήθηκε και ο φυσικός χώρος στη μνήμη του υπολογιστή όπου θα αποθηκευτούν τα δεδομένα. Τα arrays

$X, Y, Q$  περνάνε στην υπορουτίνα “by reference”, δηλ. δίνεται η θέση στη μνήμη στην οποία είναι αποθηκευμένα και όχι “by value”. Το μέγεθος του βήματος  $\Delta l$ , καθώς και η μέγιστη απόσταση σχεδιασμού, καθορίζονται με τον προσδιορισμό parameter στις δύο δηλώσεις:

```
real ,parameter      :: step=0.01
real ,parameter      :: max_dist=20.0
```

Οι τιμές αυτές θα πρέπει να καθοριστούν προσεκτικά από τον προγραμματιστή ανάλογα με τη ζητούμενη ακρίβεια επίλυσης του προβλήματος.

Στο κυρίως πρόγραμμα παρατηρούμε τον εξωτερικό βρόχο

```
do direction = -1,1,2
  dl = direction * step
  ...
enddo
```

ο οποίος αλλάζει την κατεύθυνση κίνησης πάνω στη δυναμική γραμμή. Η εντολή `do direction = -1,1,2` εκτελεί το βρόχο για `direction` από  $-1$  έως  $1$  με βήμα  $2$ . Δηλ. θα εκτελεστεί δύο φορές για `direction =  $\pm 1$` . Άρα το βήμα `dl` έχει κάθε φορά διαφορετικό πρόσημο.

Στη συνέχεια, οι εντολές `x0 = xin`, `y0 = yin` ορίζουν το αρχικό σημείο της δυναμικής γραμμής.  $(x_0, y_0)$  είναι το εκάστοτε σημείο της δυναμικής γραμμής το οποίο τυπώνουμε στο standard output με την εντολή `print`.  $(dx, dy)$  είναι το βήμα μετακίνησης πάνω στη δυναμική γραμμή, έτσι ώστε  $(x_0, y_0) \rightarrow (x_0+dx, y_0+dy)$  μετά από κάθε υπολογισμό. Ο σχεδιασμός της δυναμικής γραμμής γίνεται στον εσωτερικό βρόχο

```
call mdist(x0,y0,X,Y,N,rmin,rmax)
do while(rmin .gt. (2.0*step) .and. rmax .lt. max_dist)
  ...
  call mdist(x0,y0,X,Y,N,rmin,rmax)
enddo
```

ο οποίος εκτελείται, όταν η συνθήκη `rmin .gt. (2.0*step) .and. rmax .lt. max_dist` έχει την τιμή `.TRUE.`, είναι δηλ. αληθής. Αυτό ισχύει όσο η ελάχιστη απόσταση από όλα τα φορτία στο εκάστοτε σημείο που βρισκόμαστε δεν έχει γίνει μικρότερη ή ίση από  $2.0 \cdot \text{step}$  και η μέγιστη απόσταση από οποιοδήποτε φορτίο παραμένει μικρότερη από `max_dist`<sup>4</sup>. Οι μέγιστες και ελάχιστες αποστάσεις προσδιορίζονται με

<sup>4</sup>Εδώ μπορείτε να βάλετε και `rmin .lt. max_dist`.

κάλεσμά στην υπορουτίνα `mdist` την οποία θα μελετήσουμε παρακάτω.

Το ηλεκτρικό πεδίο που θα χρησιμοποιηθεί στη Σχέση (7.5) υπολογίζεται καλώντας `efield(x0+0.5*dx,y0+0.5*dy,X,Y,Q,N,Ex,Ey)`. Τα δύο πρώτα ορίσματα της υπορουτίνας είναι το σημείο στο οποίο ζητούμε το ηλεκτρικό πεδίο, και αυτό επιλέγεται να είναι το  $(x_0+dx/2, y_0+dy/2)$  αντί για το  $(x_0, y_0)$ . Αυτό γίνεται για να μειώσουμε το συστηματικό σφάλμα από τη διακριτοποίηση του προβλήματος παίρνοντας συνεισφορά από το μέσο του διαστήματος  $(x_0, x_0+dx)$  και  $(y_0, y_0+dy)$  και όχι από το ένα άκρο μόνο.

Μετά τον υπολογισμό του ηλεκτρικού πεδίου, η εφαρμογή της σχέσης (7.5) γίνεται απλά με τις εντολές

```
E = sqrt(Ex*Ex+Ey*Ey)
dx = dl*Ex/E
dy = dl*Ey/E
x0 = x0 + dx
y0 = y0 + dy
```

Ο έλεγχος των παθολογικών καταστάσεων  $E=0.0$  και  $dx=dy=0.0$  γίνεται με την εντολή

```
if( E .le. 1.0e-10 ) exit
```

όπου, όταν το μέτρο του πεδίου γίνει πολύ μικρό σταματάει ο υπολογισμός βγαίνοντας από το βρόχο με την εντολή `exit`. Ομολογουμένως, χάριν απλότητας, δεν είμαστε πολύ προσεκτικοί και ο αναγνώστης καλείται να θεραπεύσει και άλλες παθολογικές καταστάσεις σε σχετική άσκηση.

Με παρόμοιο τρόπο προγραμματίζουμε και την `epotline`. Ο σχετικός κώδικας παρατίθεται παρακάτω:

```
! *****
subroutine epotline(xin,yin,X,Y,Q,N)
! *****
implicit none
integer      :: N
real,dimension(N) :: X,Y,Q
real         :: xin,yin,x0,y0
real,parameter :: step=0.02
real,parameter :: max_dist=20.0
integer      :: i
real         :: r,dx,dy,dl
real         :: Ex,Ey,E
```

```

dl = step
x0 = xin
y0 = yin
dx = 0.0
dy = 0.0
r = step                                !in order to start loop
do while( r .gt. (0.9*dl) .and. r .lt. max_dist)
  print *,x0,y0
! We evaluate the E-field at the midpoint: This reduces
! systematic errors
  call efield(x0+0.5*dx,y0+0.5*dy,X,Y,Q,N,Ex,Ey)
  E = sqrt(Ex*Ex+Ey*Ey)
  if( E .le. 1.0e-10 ) exit
  dx = dl*Ey/E
  dy = -dl*Ex/E
  x0 = x0 + dx
  y0 = y0 + dy
  r = sqrt((x0-xin)**2+(y0-yin)**2)
enddo                                  !do while()
end subroutine epotline

```

Οι διαφορές με το προηγούμενο πρόγραμμα είναι λίγες: Οι ισοδυναμικές καμπύλες είναι κλειστές, άρα τις διασχίζουμε μόνο κατά μία φορά και το κριτήριο τερματισμού του υπολογισμού είναι το να φτάσουμε αρκετά κοντά στο αρχικό σημείο:

```

do while( r .gt. (0.9*dl) .and. r .lt. max_dist)
  ...
enddo

```

Ο παραπάνω βρόχος εκτελείται μέχρι η απόσταση από το αρχικό σημείο να γίνει μικρότερη από  $0.9*dl$  ή να φύγουμε εκτός της περιοχής σχεδιασμού. Τα  $dx$ ,  $dy$  υπολογίζονται σύμφωνα με την (7.6):

```

dx = dl*Ey/E
dy = -dl*Ex/E

```

Η υπορουτίνα `efield` προγραμματίζεται εφαρμόζοντας τους τύπους (7.2)<sup>5</sup>:

```

! *****
subroutine efield(x0,y0,X,Y,Q,N,Ex,Ey)

```

<sup>5</sup>Βελτιώστε το πρόγραμμα, ώστε να λαμβάνει υπόψη του την περίπτωση  $r_i = 0$ .

```

! *****
implicit none
integer      :: N
real,dimension(N) :: X,Y,Q
real        :: x0,y0,dx,dy,Ex,Ey
integer      :: i
real        :: r3,xi,yi

Ex = 0.0
Ey = 0.0
do i= 1,N
  xi = x0-X(i)
  yi = y0-Y(i)
  r3 = (xi*xi+yi*yi)**(-1.5)
  Ex = Ex + Q(i)*xi*r3
  Ey = Ey + Q(i)*yi*r3
enddo
end subroutine efield

```

Τέλος, η υπορουτίνα `mdist` που χρησιμοποιούμε αρκετά στα παραπάνω υπολογίζει την ελάχιστη και μέγιστη απόσταση `rmin` και `rmax` από ένα δεδομένο σημείο `(x0,y0)`:

```

! *****
subroutine mdist(x0,y0,X,Y,N,rmin,rmax)
! *****
implicit none
integer      :: N
real,dimension(N) :: X,Y
real        :: x0,y0,rmin,rmax
integer      :: i
real        :: r

rmax = 0.0
rmin = 1000.0
do i = 1,N
  r = sqrt((x0-X(i))**2 + (y0-Y(i))**2)
  if(r.GT.rmax) rmax = r
  if(r.LT.rmin) rmin = r
enddo
end subroutine mdist

```

Εδώ η αρχική τιμή του `rmin` πρέπει να οριστεί προσεκτικά ανάλογα με τα όρια σχεδίασης των δυν. γραμμών.

## 7.4 Το Πρόγραμμα - Σύνοψη

Στην παράγραφο αυτή παρατίθενται για διευκόλυνση του αναγνώστη ολόκληρα τα προγράμματα των δύο τελευταίων παραγράφων, καθώς και συνοπτικές οδηγίες για τη μεταγλώττιση και ανάλυση των αποτελεσμάτων. Μπορείτε, αν θέλετε, πρώτα να αντιγράψετε τα προγράμματα στα αντίστοιχα αρχεία και να εκτελέσετε τις εντολές συλλογής και ανάλυσης των δεδομένων που δίνονται εδώ χωρίς εξήγηση και μετά να επιστρέψετε στις προηγούμενες παραγράφους για βαθύτερη κατανόηση των πεπραγμένων.

Πρώτα δίνουμε τα περιεχόμενα του αρχείου ELines.f90:

```
! *****
program Electric_Fields
! *****
  implicit none
  integer ,parameter:: P=20          !max number of charges
  real ,dimension(P):: X,Y,Q
  integer          :: N
  integer          :: i,j,nd
  real             :: x0,y0,theta
  real ,parameter  :: PI= 3.14159265359
!----- SET CHARGE DISTRIBUTION -----
  print *, '# Enter number of charges:'
  read  *, N
  print *, '# N= ',N
  do i=1,N
    print *, '# Charge: ',i
    print *, '# Position and charge: (X,Y,Q):'
    read  *, X(i),Y(i),Q(i)
    print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
  enddo
!----- DRAWING LINES -----
!We draw 2*nd field lines around each charge
  nd = 6
  do i = 1,N
    do j = 1,(2*nd)
      theta = (PI/nd)*j
      x0 = X(i) + 0.1 * cos(theta)
      y0 = Y(i) + 0.1 * sin(theta)
      call eline(x0,y0,X,Y,Q,N)
    enddo
  enddo
end program Electric_Fields
! *****
subroutine eline(xin,yin,X,Y,Q,N)
```

```

! *****
implicit none
integer      :: N
real,dimension(N) :: X,Y,Q
real         :: xin,yin,x0,y0
real,parameter :: step=0.01
real,parameter :: max_dist=20.0
integer      :: i,direction
real         :: rmin,rmax,r,dx,dy,d1
real         :: Ex,Ey,E
do direction = -1,1,2                !direction= +/- 1
  d1 = direction * step
  x0 = xin
  y0 = yin
  dx = 0.0
  dy = 0.0
  call mdist(x0,y0,X,Y,N,rmin,rmax)
  do while(rmin .gt. (2.0*step) .and. rmax .lt. max_dist)
    print *,x0,y0
! We evaluate the E-field at the midpoint: This reduces
! systematic errors
    call efield(x0+0.5*dx,y0+0.5*dy,X,Y,Q,N,Ex,Ey)
    E = sqrt(Ex*Ex+Ey*Ey)
    if( E .le. 1.0e-10 ) exit
    dx = d1*Ex/E
    dy = d1*Ey/E
    x0 = x0 + dx
    y0 = y0 + dy
    call mdist(x0,y0,X,Y,N,rmin,rmax)
  enddo                                !do while()
enddo                                !do direction = -1,1,2
end subroutine eline
! *****
subroutine efield(x0,y0,X,Y,Q,N,Ex,Ey)
! *****
implicit none
integer      :: N
real,dimension(N) :: X,Y,Q
real         :: x0,y0,dx,dy,Ex,Ey
integer      :: i
real         :: r3,xi,yi

Ex = 0.0
Ey = 0.0
do i= 1,N
  xi = x0-X(i)
  yi = y0-Y(i)
! Exercise: Improve code so that xi*xi+yi*yi=0 is taken care of
  r3 = (xi*xi+yi*yi)**(-1.5)

```



```

    Ex = Ex + Q(i)*xi*r3
    Ey = Ey + Q(i)*yi*r3
enddo
end subroutine efield
!*****
subroutine mdist(x0,y0,X,Y,N,rmin,rmax)
!*****
    implicit none
    integer          :: N
    real,dimension(N) :: X,Y
    real             :: x0,y0,rmin,rmax
    integer          :: i
    real             :: r

    rmax = 0.0
    rmin = 1000.0
    do i = 1,N
        r = sqrt((x0-X(i))**2 + (y0-Y(i))**2)
        if(r.GT.rmax) rmax = r
        if(r.LT.rmin) rmin = r
    enddo
end subroutine mdist

```

Στη συνέχεια, δίνουμε τα περιεχόμενα του αρχείου EPotential.f90:

```

!*****
program Electric_Potential
!*****
    implicit none
    integer,parameter :: P=20      !max number of charges
    real,dimension(P) :: X,Y,Q
    integer          :: N
    real,parameter   :: PI= 3.14159265359
    integer          :: i,j,nd
    real             :: x0,y0,rmin,rmax,L

    print *, '# Enter number of charges:'
    read  *, N
    print *, '# N= ',N
    do i=1,N
        print *, '# Charge: ',i
        print *, '# Position and charge: (X,Y,Q):'
        read  *, X(i),Y(i),Q(i)
        print *, '# (X,Y)= ', X(i),Y(i), ' Q= ',Q(i)
    enddo

    !----- DRAWING LINES -----
    !We draw lines passing through an equally
    !spaced lattice of N=(2*nd+1)x(2*nd+1) points

```

```

!in the square  $-L \leq x \leq L$ ,  $-L \leq y \leq L$ .
nd      = 4
L        = 1.0
do i = -nd,nd
  do j = -nd,nd
    x0 = i*(L/nd)
    y0 = j*(L/nd)
    print *, '# @ ', i, j, L/nd, x0, y0
    call mdist(x0,y0,X,Y,N,rmin,rmax)
!we avoid getting too close to a charge:
    if(rmin .gt. L/(nd*10) )&
      call epotline(x0,y0,X,Y,Q,N)
  enddo
enddo
end program Electric_Potential
!*****
subroutine epotline(xin,yin,X,Y,Q,N)
!*****
  implicit none
  integer                :: N
  real , dimension(N)    :: X,Y,Q
  real                   :: xin,yin,x0,y0
  real , parameter       :: step=0.02
  real , parameter       :: max_dist=20.0
  integer                :: i
  real                   :: r,dx,dy,dl
  real                   :: Ex,Ey,E

  dl = step
  x0 = xin
  y0 = yin
  dx = 0.0
  dy = 0.0
  r = step
!in order to start loop
do while( r .gt. (0.9*dl) .and. r .lt. max_dist)
  print *,x0,y0
! We evaluate the E-field at the midpoint: This reduces
! systematic errors
  call efield(x0+0.5*dx,y0+0.5*dy,X,Y,Q,N,Ex,Ey)
  E = sqrt(Ex*Ex+Ey*Ey)
  if( E .le. 1.0e-10 ) exit
  dx = dl*Ey/E
  dy = -dl*Ex/E
  x0 = x0 + dx
  y0 = y0 + dy
  r = sqrt((x0-xin)**2+(y0-yin)**2)
enddo
!do while()
end subroutine epotline
...

```

όπου οι ... είναι οι υπορουτίνες `efield` και `mdist` οι οποίες είναι ίδιες με αυτές του αρχείου `ELines.f90`.

Τέλος, υπενθυμίζουμε στον αναγνώστη πώς να τα τρέξει και να δει τα αποτελέσματα. Μεταγλωττίζουμε το πρόγραμμα με την εντολή

```
> gfortran ELines.f90      -o el
> gfortran EPotential.f90 -o ep
```

Στη συνέχεια, γράφουμε σε ένα αρχείο `Input` τα δεδομένα της κατανομής των φορτίων. Για παράδειγμα:

```
4           N: Number of charges
1   1   -1   (X,Y,Q): Position and charge
-1  1    1   (X,Y,Q): Position and charge
1  -1    1   (X,Y,Q): Position and charge
-1 -1   -1   (X,Y,Q): Position and charge
```

Τα αποτελέσματα τα παίρνουμε με τις εντολές:

```
> ./el < Input > el.dat
> ./ep < Input > ep.dat
> gnuplot
gnuplot> plot "el.dat" with dots
gnuplot> plot "ep.dat" with dots
```

Καλή ... διασκέδαση!

## 7.5 Ηλεκτροστατικό Πεδίο στο Κενό

Θεωρούμε ηλεκτρικό πεδίο το οποίο είναι ανεξάρτητο του χρόνου σε περιοχή του χώρου που είναι άδεια από ηλεκτρικά φορτία. Οι εξισώσεις του Maxwell στην περίπτωση αυτή είναι η εξίσωση Gauss

$$\vec{\nabla} \cdot \vec{E}(x, y, z) = \frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_z}{\partial z} = 0, \quad (7.8)$$

μαζί με την εξίσωση που ορίζει το ηλεκτροστατικό δυναμικό<sup>6</sup>

$$\vec{E}(x, y, z) = -\vec{\nabla}V(x, y, z). \quad (7.9)$$

---

<sup>6</sup>Ισοδύναμη με την εξίσωση  $\vec{\nabla} \times \vec{E} = 0$ .

Οι εξισώσεις (7.8) και (7.9) μας δίνουν μία εξίσωση Laplace για τη  $V(x, y, z)$ :

$$\nabla^2 V(x, y, z) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0. \quad (7.10)$$

Η λύση της παραπάνω εξίσωσης είναι ένα πρόβλημα συνοριακών συνθηκών: Ζητούμε το δυναμικό  $V(x, y, z)$  σε μία περιοχή του χώρου  $\mathcal{S}$  η οποία περιβάλλεται από το όριό της, μία κλειστή επιφάνεια  $\partial\mathcal{S}$ . Όταν το δυναμικό είναι γνωστό πάνω στην  $\partial\mathcal{S}$ , η λύση της (7.10) είναι μοναδική και το δυναμικό, και κατ' επέκταση το ηλεκτρικό πεδίο, προσδιορίζεται παντού στην  $\mathcal{S}$ .

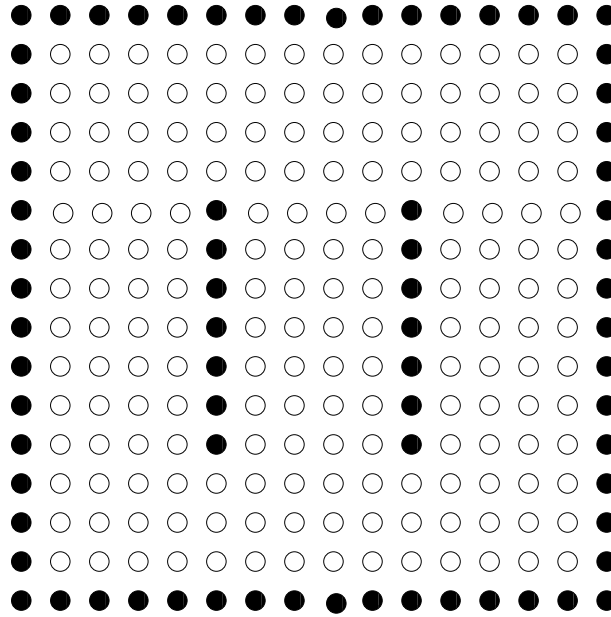
Για απλότητα θεωρούμε το πρόβλημα ανηγμένο στο επίπεδο, οπότε  $V = V(x, y)$ . Στην περίπτωση αυτή, ο τελευταίος όρος στην (7.10) απουσιάζει, η  $\mathcal{S}$  είναι συμπαγές τμήμα του επιπέδου και η  $\partial\mathcal{S}$  είναι μια κλειστή καμπύλη.

Για να μελετήσουμε το πρόβλημα αριθμητικά, προσεγγίζουμε το χώρο με ένα τετραγωνικό και τετράγωνο πλέγμα με το δυναμικό να ορίζεται μόνο πάνω σε  $N$  πλεγματικές θέσεις. Αν το μήκος κάθε πλευράς είναι  $l$  και τα γειτονικά πλεγματικά σημεία απέχουν μεταξύ τους απόσταση<sup>7</sup>  $a$ , τότε  $l = (L - 1)a$ , όπου  $L = \sqrt{N}$  είναι ο αριθμός των πλεγματικών σημείων σε κάθε πλευρά του πλέγματος. Η προσέγγιση της συνεχούς λύσης επιτυγχάνεται, όταν η διακριτοποίηση αυτή εκλεπτύνεται διαρκώς, παίρνοντας τον αριθμό των πλεγματικών σημείων  $N \rightarrow \infty$  και  $a \rightarrow 0$ , έτσι ώστε το μήκος  $l = (L - 1)a$  να παραμένει σταθερό. Η καμπύλη  $\partial\mathcal{S}$  προσεγγίζεται από τις πλεγματικές θέσεις που αποτελούν το όριο του πλέγματος και πιθανώς από άλλα σημεία στο εσωτερικό στα οποία η τιμή του δυναμικού είναι δεδομένη. Το φυσικό σύστημα που αντιστοιχεί στο μοντέλο μας είναι μια διάταξη από αγωγούς στην επιφάνεια των οποίων το δυναμικό είναι σταθερό και ζητείται το ηλεκτρικό πεδίο που δημιουργείται στον περιβάλλοντα χώρο τους. Μία τέτοια διάταξη φαίνεται στο σχήμα 7.6.

Για να βρούμε την εξίσωση πεπερασμένων διαφορών που θα προ-

---

<sup>7</sup>Η απόσταση  $a$  λέγεται πλεγματική σταθερά.



Σχήμα 7.6: Πλέγμα που αντιστοιχεί στη διάταξη δύο παράλληλων, επίπεδων αγωγών μέσα σε γειωμένο μεταλλικό κουτί. Οι μαύρες πλεγματικές θέσεις αντιστοιχούν σε σημεία σταθερού δυναμικού, ενώ οι άσπρες σε σημεία του κενού χώρου.

σεγγίζει την εξίσωση (7.10), θεωρούμε το ανάπτυγμα κατά Taylor:

$$\begin{aligned}
 V(x + \delta x, y) &= V(x, y) + \frac{\partial V}{\partial x} \delta x + \frac{1}{2} \frac{\partial^2 V}{\partial x^2} (\delta x)^2 + \dots \\
 V(x - \delta x, y) &= V(x, y) - \frac{\partial V}{\partial x} \delta x + \frac{1}{2} \frac{\partial^2 V}{\partial x^2} (\delta x)^2 + \dots \\
 V(x, y + \delta y) &= V(x, y) + \frac{\partial V}{\partial y} \delta y + \frac{1}{2} \frac{\partial^2 V}{\partial y^2} (\delta y)^2 + \dots \\
 V(x, y - \delta y) &= V(x, y) - \frac{\partial V}{\partial y} \delta y + \frac{1}{2} \frac{\partial^2 V}{\partial y^2} (\delta y)^2 + \dots
 \end{aligned}$$

Παίρνοντας το άθροισμα και των δύο μελών της παραπάνω εξίσωσης,  $\delta x = \delta y$  και αγνοώντας τους όρους  $\dots$  παίρνουμε

$$\begin{aligned}
 &V(x + \delta x, y) + V(x - \delta x, y) + V(x, y + \delta y) + V(x, y - \delta y) \\
 &= 4V(x, y) + (\delta x)^2 \left( \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} \right) + \dots \\
 &\approx 4V(x, y), \tag{7.11}
 \end{aligned}$$

όπου χρησιμοποιήσαμε την εξίσωση (7.10) για να διώξουμε το δεύτερο όρο της δεύτερης σειράς.

Θεωρώντας τις συντεταγμένες των πλεγματοειδών σημείων να δίνονται από ακέραιους  $(i, j)$  τέτοιους, ώστε  $x_i = (i - 1)a$  και  $y_j = (j - 1)a$  με  $i, j = 1, \dots, L$ , και παίρνοντας  $\delta x = \delta y = a$ , έτσι ώστε  $x_i \pm \delta x = x_{i \pm 1} = (i - 1 \pm 1)a = x_{i \pm 1}$  και  $y_j \pm \delta y = y_{j \pm 1} = (j - 1 \pm 1)a = y_{j \pm 1}$ , η παραπάνω εξίσωση γίνεται:

$$V(i, j) = \frac{1}{4}(V(i - 1, j) + V(i + 1, j) + V(i, j - 1) + V(i, j + 1)), \quad (7.12)$$

η οποία έχει την ακόλουθη απλή ερμηνεία: Το δυναμικό στη θέση  $(i, j)$  είναι απλά ο μέσος όρος του δυναμικού των πλησιέστερων γειτόνων του. Ο αλγόριθμος που θα ακολουθήσουμε εντάσσεται στη γενική κατηγορία μεθόδων “χαλάρωσης” (relaxation methods) και τα βασικά του βήματα είναι:

1. Καθορίζεται το μέγεθος τετραγωνικού πλέγματος με  $L$  πλεγματικές θέσεις σε κάθε πλευρά.
2. Καθορίζονται οι πλεγματικές θέσεις που έχουν καθορισμένο δυναμικό και δίνεται η τιμή του δυναμικού στις θέσεις αυτές.
3. Καθορίζεται αυθαίρετη αρχική τιμή για το δυναμικό στις υπόλοιπες θέσεις. Μια καλή επιλογή θα δώσει γρηγορότερη σύγκλιση προς τη σωστή συνάρτηση  $V(x, y)$ . Μια κακή επιλογή θα δώσει αργή σύγκλιση, σύγκλιση προς λανθασμένη λύση ή καθόλου σύγκλιση.
4. Σε διαδοχικά επαναλαμβανόμενα “περάσματα” (sweeps) του πλέγματος επισκεπτόμαστε κάθε πλεγματική θέση, υπολογίζουμε το δυναμικό  $V(i, j)$  από τη σχέση (7.12) και ορίζουμε αυτή να είναι η νέα τιμή του δυναμικού στη θέση αυτή.
5. Η διαδικασία σταματάει όταν σε δύο διαδοχικά περάσματα η μεταβολή του δυναμικού είναι ικανοποιητικά μικρή. Υποθέτουμε δηλ. ότι έχουμε πρακτικά επιτύχει τη σύγκλιση στη σωστή λύση.

Η προσεκτική χρήση του παραπάνω αλγόριθμου απαιτεί να τον μελετήσουμε με διαφορετικά κριτήρια “ικανοποιητικά μικρής” διαφοράς και διαφορετικές αρχικές συνθήκες.

Παρακάτω, παραθέτουμε πρόγραμμα που χρησιμοποιεί τον παραπάνω αλγόριθμο για τον υπολογισμό του ηλεκτροστατικού δυναμικού δύο παράλληλων επίπεδων μεταλλικών αγωγών που βρίσκονται μέσα σε ένα γειωμένο, τετράγωνο, μεταλλικό αγωγίμο κουτί. Το πλέγμα φαίνεται στο σχήμα 7.6, όπου οι μαύρες κουκκίδες αναπαριστούν τους

αγωγούς. Το κουτί έχει δυναμικό  $V = 0$ , ενώ οι αγωγοί δυναμικό  $V_1$  και  $V_2$  αντίστοιχα. Ο χρήστης στην είσοδο δίνει τις τιμές  $V_1$  και  $V_2$ , το μέγεθος  $L$  του πλέγματος και την ακρίβεια σύγκλισης που επιθυμεί. Η τελευταία καθορίζεται ποσοτικά από ένα μικρό αριθμό  $\epsilon$ . Η μέγιστη μεταβολή στο δυναμικό ανάμεσα σε δύο διαφορετικά περάσματα του πλέγματος θα πρέπει να είναι μικρότερη από  $\epsilon$  για επίτευξη σύγκλισης της μεθόδου.

Η δομή των δεδομένων είναι απλή. Χρειαζόμαστε ένα πραγματικό array  $V(L,L)$  με αποθηκευμένες τις τιμές του δυναμικού και ένα logical array  $isConductor(L,L)$  το οποίο καθορίζει αν μια πλεγματική θέση έχει καθορισμένο δυναμικό (“αγωγός” = `.TRUE.`) ή όχι (“κενός χώρος” = `.FALSE.`).

Το κυρίως πρόγραμμα, αφού διαβάσει τα δεδομένα από το χρήστη, καλεί τρεις υπορουτίνες:

1. `initialize_lattice(V,isConductor,L,V1,V2):`

Τα δεδομένα εισόδου της ρουτίνας είναι το δυναμικό  $V_1$  του αριστερού αγωγού και το δυναμικό  $V_2$  του δεξιού αγωγού, καθώς και το μέγεθος του πλέγματος  $L$ . Οι μεταβλητές εξόδου είναι τα arrays  $V(L,L)$  και  $isConductor(L,L)$  τα οποία στην έξοδο έχουν τις αρχικές τιμές:  $V_1$  στον αριστερό αγωγό,  $V_2$  στο δεξί, 0 στο κουτί, καθώς και την αυθαίρετη τιμή 0 για τις υπόλοιπες πλεγματικές θέσεις. Η γεωμετρία της διάταξης είναι κωδικοποιημένη και ο χρήστης που θα θέλει να τη μεταβάλλει (λ.χ. μεταβολή απόστασης αγωγών, προσθήκη/αφαίρεση αγωγού κλπ) θα πρέπει να επεμβεί στον κώδικα για να κάνει τις απαραίτητες μετατροπές.

2. `laplace(V,isConductor,L,epsilon):`

Η καρδιά του προγράμματος. Στην είσοδο, παρέχουμε τα αρχικοποιημένα arrays  $V(L,L)$ ,  $isConductor(L,L)$  και το μέγεθος κάθε διάστασής τους  $L$ , καθώς και την επιθυμητή ακρίβεια σύγκλισης  $\epsilon$ . Στην έξοδο, παίρνουμε την τελική λύση  $V(L,L)$  για περαιτέρω επεξεργασία. Στη ρουτίνα αυτή υπολογίζουμε τη μέση τιμή του δυναμικού  $V_{av}$  των πλησιέστερων γειτόνων κάθε πλεγματικής θέσης και αμέσως αλλάζουμε την τιμή  $V(i,j)=V_{av}$ <sup>8</sup>. Η μέγιστη τιμή  $error$  της απόκλισης της νέας τιμής του δυναμικού  $V_{av}$  από την παλιά  $V(i,j)$  υπολογίζεται για κάθε σάρωση του πλέγματος. Αν

<sup>8</sup>Παραλλαγή της μεθόδου θα ήταν η τιμή  $V_{av}$  να αποθηκευτεί σε ένα προσωρινό array  $V_{new}(i,j)$  και η αλλαγή των τιμών  $V(i,j)=V_{new}(i,j)$  να γίνεται μετά από τη σάρωση του πλέγματος. Ποια μέθοδος περιμένετε να έχει καλύτερες ιδιότητες σύγκλισης; Δοκιμάστε...

αυτή γίνεται μικρότερη από μια δεδομένη τιμή  $\epsilon$  η διαδικασία ...χαλάρωσης σταματάει.

### 3. print\_results(V,L):

Η υπορουτίνα αυτή απλά τυπώνει το δυναμικό  $V(L,L)$  στο αρχείο data. Τυπώνονται οι συντεταγμένες  $i, j$  και η τιμή  $V(i,j)$  σε κάθε γραμμή. Το μόνο αξιο λόγου σημείο είναι ότι κάθε φορά που αλλάζει ο δείκτης  $i$  το πρόγραμμα τυπώνει μια κενή γραμμή. Αυτό γίνεται για να επεξεργαστούμε τα αποτελέσματα με το πρόγραμμα γραφικών gnuplot. Για να γίνει η τρισδιάστατη γραφική παράσταση συνάρτηση δύο μεταβλητών με την εντολή splot πρέπει το format των δεδομένων να είναι αυτής της μορφής.

Όλο το πρόγραμμα ακολουθεί παρακάτω:

```
! *****
!PROGRAM LAPLACE_EM
!Computes the electrostatic potential around conductors.
!The computation is performed on a square lattice of linear
!dimension L. A relaxation method is used to converge to the
!solution of Laplace equation for the potential.
!DATA STRUCTURE:
!real(8) V(L,L): Value of the potential on the lattice sites
!logical isConductor(L,L): If .TRUE. site has fixed potential
!                          If .FALSE. site is empty space
!real epsilon: Determines the accuracy of the solution
!The maximum difference of the potential on each site between
!two consecutive sweeps should be less than epsilon.
!PROGRAM STRUCTURE
!main program:
! . Data Input
! . call subroutines for initialization , computation and
!   printing of results
!subroutine initialize_lattice:
! . Initialization of V(L,L) and isConductor(L,L)
!subroutine laplace:
! . Solves laplace equation using a relaxation method
!subroutine print_results:
! . Prints results for V(L,L) in a file . Uses format compatible
!with splot of gnuplot.
! *****

program laplace_em
  implicit none
!P defines the size of the arrays and is equal to L
  integer ,parameter      :: P=31
  logical ,dimension(P,P) :: isConductor
```



```

    real(8),dimension(P,P) :: V
!V1 and V2 are the values of the potential on the interior
!conductors. epsilon is the accuracy desired for the
!convergence of the relaxation method in subroutine laplace()
    real(8)                :: V1,V2,epsilon
    integer                 :: L

!The user should provide the necessary data: V1,V2 and epsilon
    L = P
    print *, 'Enter V1,V2: '
    read  *, V1,V2
    print *, 'Enter epsilon: '
    read  *, epsilon
    print *, 'Starting Laplace: '
    print *, 'Grid Size= ',L
    print *, 'Conductors set at V1= ',V1, ' V2= ',V2
    print *, 'Relaxing with accuracy epsilon= ',epsilon
!The arrays V and isConductor are initialized
    call initialize_lattice(V,isConductor,L,V1,V2)
!We enter initialized V,isConductor. On exit the
!routine gives the solution V
    call laplace(V,isConductor,L,epsilon)
!We print V in a file.
    call print_results(V,L)

end program laplace_em
!*****
!subroutine initialize_lattice
!Initializes arrays V(L,L) and isConductor(L,L).
!V(L,L)= 0.0 and isConductor(L,L)= .FALSE. by default
!isConductor(i,j)= .TRUE. on boundary of lattice where V=0
!isConductor(i,j)= .TRUE. on sites with i= L/3+1, 5<= j <= L-5
!isConductor(i,j)= .TRUE. on sites with i=2*L/3+1, 5<= j <= L-5
!V(i,j) = V1 on all sites with i= L/3+1, 5<= j <= L-5
!V(i,j) = V2 on all sites with i=2*L/3+1, 5<= j <= L-5
!V(i,j) = 0 on boundary (i=1,L and j=1,L)
!V(i,j) = 0 on interior sites with isConductor(i,j)= .FALSE.
!INPUT:
!integer L: Linear size of lattice
!real(8) V1,V2: Values of potential on interior conductors
!OUTPUT:
!real(8) V(L,L): Array provided by user. Values of potential
!logical isConductor(L,L): If .TRUE. site has fixed potential
!                          If .FALSE. site is empty space
!*****
subroutine initialize_lattice(V,isConductor,L,V1,V2)
    implicit none
    integer                :: L
    logical,dimension(L,L) :: isConductor

```

```

real(8),dimension(L,L) :: V
real(8)                  :: V1,V2
integer                  :: i,j

!Initialize to 0 and .FALSE (default values for boundary and
!interior sites).
V      = 0.0D0
isConductor = .FALSE.
!We set the boundary to be a conductor: (V=0 by default)
do i=1,L
  isConductor(1,i) = .TRUE.
  isConductor(i,1) = .TRUE.
  isConductor(L,i) = .TRUE.
  isConductor(i,L) = .TRUE.
enddo
!We set two conductors at given potential V1 and V2
do i=5,L-5
  V      ( L/3+1,i) = V1
  isConductor( L/3+1,i) = .TRUE.
  V      (2*L/3+1,i) = V2
  isConductor(2*L/3+1,i) = .TRUE.
enddo

end subroutine initialize_lattice
!*****
!subroutine laplace
!Uses a relaxation method to compute the solution of the
!Laplace equation for the electrostatic potential on a
!2 dimensional square lattice of linear size L.
!At every sweep of the lattice compute the average Vav
!at each site (i,j) and we immediately update V(i,j)
!The computation continues until Max |Vav-V(i,j)| < epsilon
!INPUT:
!integer L: Linear size of lattice
!real(8) V(L,L): Value of the potential at each site
!logical isConductor(L,L): If .TRUE. potential is fixed
!                               If .FALSE. potential is updated
!real(8) epsilon: if Max |Vav-V(i,j)| < epsilon return to
!callingprogram.
!OUTPUT:
!real(8) V(L,L): The computed solution for the potential
!*****
subroutine laplace(V,isConductor,L,epsilon)
  implicit none
  integer :: L
  logical,dimension(L,L) :: isConductor
  real(8),dimension(L,L) :: V
  real(8) :: epsilon
  integer :: i,j,icount

```

```

real(8)                                :: Vav,error,dV

icount = 0                             !counts number of sweeps
do while (.TRUE.)                      !an infinite loop:
  error = 0.0D0                         !Exit when error<epsilon
  do j=2,L-1
    do i=2,L-1
!We change V only for non conductors:
      if( .NOT. isConductor(i,j))then
        Vav = ( V(i-1,j)+V(i+1,j)+V(i,j+1)+V(i,j-1)) * 0.25D0
        dV = DABS(V(i,j)-Vav)
        if(error .LT. dV ) error = dV !maximum error
        V(i,j) = Vav                 ! we immediately update V(i,j)
      endif
    enddo
  enddo
  icount = icount + 1
  print *,icount,' err= ',error
  if( error .LT. epsilon) return !return to main program
enddo

end subroutine laplace
!*****
!subroutine print_results
!Prints the array V(L,L) in file "data"
!The format of the output is appropriate for the splot function
!of gnuplot: Each time i changes an empty line is printed.
!INPUT:
!integer L: size of array V
!real(8) V(L,L): array to be printed
!OUTPUT:
!no output
!*****
subroutine print_results(V,L)
  implicit none
  integer                :: L
  real(8),dimension(L,L) :: V
  integer                :: i,j

  open(unit=11,file="data")
  do i=1,L
    do j =1,L
      write(11,*)i,j,V(i,j)
    enddo
    write (11,*)'' !print empty line for gnuplot,
                  !separate isolines
  enddo

end subroutine print_results

```

## 7.6 Αποτελέσματα

Το παραπάνω πρόγραμμα το αποθηκεύουμε στο αρχείο LaplaceEq.f90. Η μεταγλώττιση του προγράμματος γίνεται με τον μεταγλωττιστή gfortran και για να το τρέξουμε δίνουμε τις εντολές:

```
> gfortran LaplaceEq.f90 -o lf
> ./lf
Enter V1,V2:
100 -100
Enter epsilon:
0.01
Starting Laplace:
Grid Size= 31
Conductors set at V1= 100. V2= -100.
Relaxing with accuracy epsilon= 0.01
1 err= 33.3333333
2 err= 14.8148148
3 err= 9.87654321
.....
110 err= 0.0106860904
111 err= 0.0101182476
112 err= 0.00958048937
```

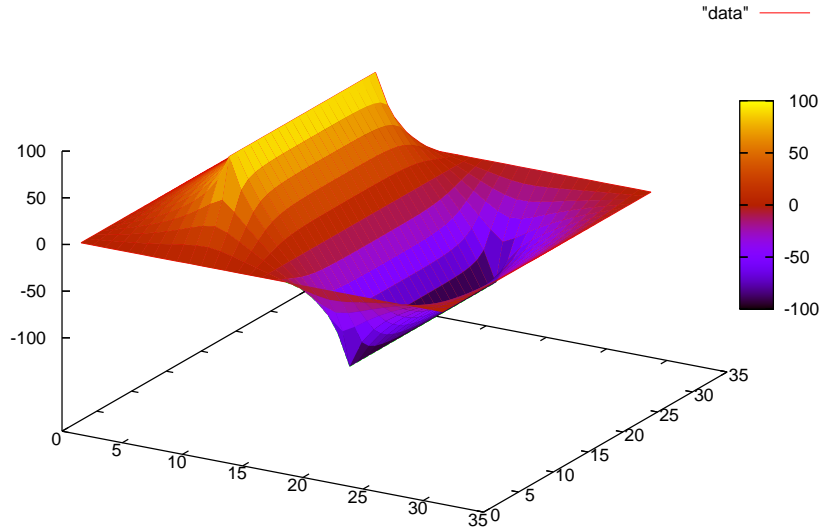
Το πρόγραμμα κάνει 112 περάσματα στο πλέγμα μέχρι το μέγιστο σφάλμα να γίνει  $0.00958048937 < 0.01$ . Τα αποτελέσματα αποθηκεύονται στο αρχείο data. Για να δούμε τα αποτελέσματα χρησιμοποιούμε το πρόγραμμα gnuplot:

```
gnuplot> set pm3d
gnuplot> set hidden3d
gnuplot> set size ratio 1
gnuplot> splot "data" with lines
```

Τα αποτελέσματα φαίνονται στο σχήμα 7.7

## 7.7 Εξίσωση Poisson

Στην παράγραφο αυτή αναφέρουμε περιληπτικά τη λύση του προβλήματος που μελετήσαμε στις προηγούμενες παραγράφους, όταν υπάρχει κατανομή στατικού ηλεκτρικού φορτίου που δίνεται από την πυκνότητα φορτίου  $\rho(\vec{r})$ . Στην περίπτωση αυτή, η εξίσωση Laplace για το δυναμικό



Σχήμα 7.7: Η λύση της εξίσωσης (7.10) από το πρόγραμμα LaplaceEq.f90 για  $L=31$ ,  $V_1=100$ ,  $V_2=-100$ ,  $\epsilon=0.01$ .

γίνεται η εξίσωση Poisson:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -4\pi\rho(x, y, z) \quad (7.13)$$

Η μορφή της εξίσωσης στο επίπεδο πλέγμα που μελετήσαμε γίνεται

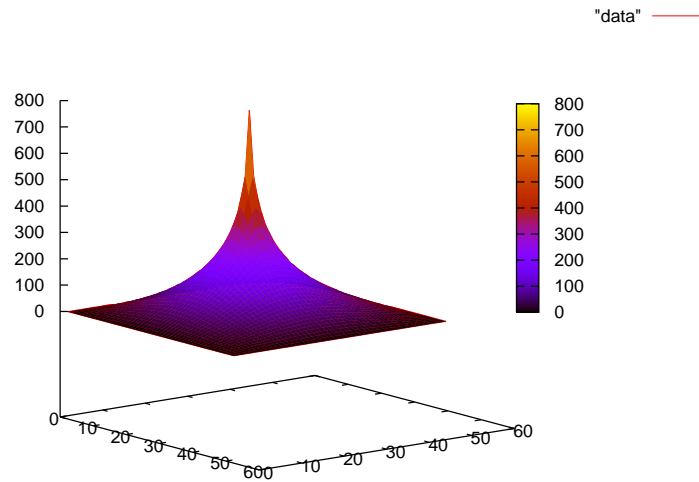
$$V(i, j) = \frac{1}{4}(V(i-1, j) + V(i+1, j) + V(i, j-1) + V(i, j+1) + \tilde{\rho}(i, j)), \quad (7.14)$$

όπου<sup>9</sup>  $\tilde{\rho}(i, j) = 4\pi a^2 \rho(i, j)$ , και είναι πάρα πολύ απλό να μετατρέψουμε τον κώδικα της προηγούμενης παραγράφου, έτσι ώστε να μελετήσουμε κατανομές φορτίου που επιθυμούμε.

Παρακάτω, παραθέτουμε το πρόγραμμα PoissonEq.f90 που λύνει την (7.14) για ομοιόμορφη κατανομή του φορτίου (σχήμα 7.10), στο οποίο έχουμε θέσει  $a=1$ . Ο αναγνώστης καλείται να αναπαράγει το σχήμα αυτό, καθώς και τα 7.8, 7.9.

---

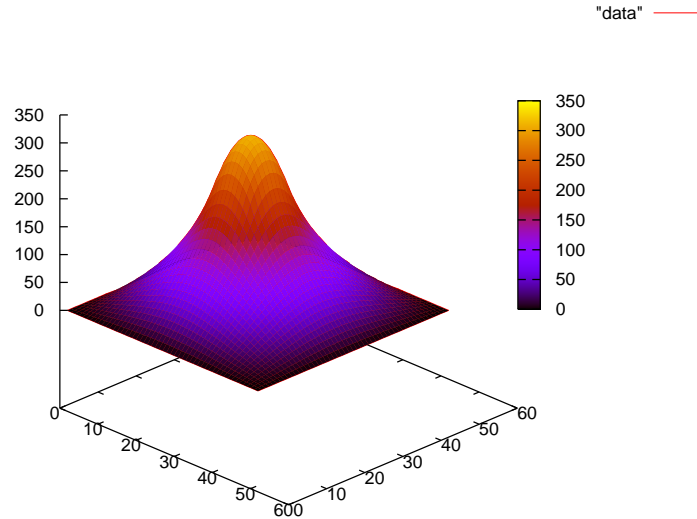
<sup>9</sup>Επειδή  $Q = \int \rho dA \approx \sum_{i,j} \rho a^2 = (1/4\pi) \sum_{i,j} \tilde{\rho}$ . Οπότε  $\sum_{i,j} \tilde{\rho} \approx 4\pi Q$ .



Σχήμα 7.8: Η λύση της εξίσωσης (7.13) από το πρόγραμμα Poisson.f90 για  $L = 51$ ,  $V = 0$  στο σύνορο και το φορτίο  $4\pi Q = 1000$  συγκεντρωμένο σε ένα σημείο.

```
! *****
! set the boundary of a square to given potentials
! *****
program poisson_eq
  implicit none
  integer, parameter      :: P=51
  logical, dimension(P,P) :: isConductor
  real(8), dimension(P,P) :: V, rho
  real(8)                 :: V1, V2, V3, V4, Q, epsilon
  integer                 :: L

  L = P
  print *, 'Enter V1,V2,V3,V4: '
  read  *, V1, V2, V3, V4
  print *, 'Enter 4*PI*Q: '
  read  *, Q
  print *, 'Enter epsilon: '
  read  *, epsilon
  print *, 'Starting Laplace: '
  print *, 'Grid Size= ', L
  print *, 'Boundaries set at V1= ', V1, ' V2= ', V2, ' V3= ', V3, &
    ' V4= ', V4, ' and Q= ', Q
  print *, 'Relaxing with accuracy epsilon= ', epsilon
```



Σχήμα 7.9: Η λύση της εξίσωσης (7.13) από το πρόγραμμα Poisson.f90 για  $L = 51$ ,  $V = 0$  στο σύνορο και το φορτίο  $4\pi Q = 1000$  κατανεμημένο ομοιόμορφα σε ένα μικρό τετράγωνο πλάτους 10 πλεγματικών σημείων.

```

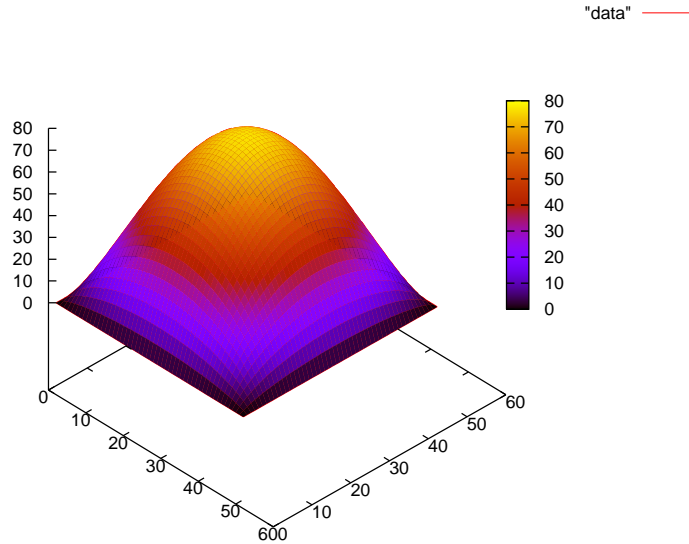
call initialize_lattice(V,isConductor,rho,L,V1,V2,V3,V4,Q)

call laplace(V,isConductor,rho,L,epsilon)

call print_results(V,L)

end program laplace_sq
! *****
subroutine &
  initialize_lattice(V,isConductor,rho,L,V1,V2,V3,V4,Q)
! *****
  implicit none
  integer                :: L
  logical,dimension(L,L) :: isConductor
  real(8),dimension(L,L) :: V,rho
  real(8)                :: V1,V2,V3,V4,Q,Area
  integer                :: i,j,L1,L2
! Initialize to 0 and .FALSE.
  V          = 0.0D0
  isConductor = .FALSE.

```



Σχήμα 7.10: Η λύση της εξίσωσης (7.13) από το πρόγραμμα Poisson.f90 για  $L = 51$ ,  $V = 0$  στο σύνορο και το φορτίο  $4\pi Q = 1000$  κατανεμημένο ομοιόμορφα σε όλες τις εσωτερικές πλεγματικές θέσεις.

```

rho          = 0.0D0
!We set the boundary to be a conductor:
do i=1,L
  isConductor(1,i) = .TRUE.
  isConductor(i,1) = .TRUE.
  isConductor(L,i) = .TRUE.
  isConductor(i,L) = .TRUE.
  V          (1,i) = V1
  V          (i,L) = V2
  V          (L,i) = V3
  V          (i,1) = V4
enddo
!We set the points with non-zero charge
!A uniform distribution at a center square
L1 = (L/2)-5
L2 = (L/2)+5
if(L1.LT.1) stop 'array rho out of bounds. Small L1'
if(L2.GT.L) stop 'array rho out of bounds. Large L2'
Area = (L2-L1+1)*(L2-L1+1)
do j=L1,L2
  do i=L1,L2
    rho(i,j) = Q/Area !rho is \tilde{\rho} in notes
  
```



```

        enddo                                !so Q is 4*PI*Q
    enddo

end subroutine initialize_lattice
!*****
subroutine laplace(V,isConductor,rho,L,epsilon)
!*****
    implicit none
    integer :: L
    logical,dimension(L,L) :: isConductor
    real(8),dimension(L,L) :: V,rho
    real(8) :: epsilon
    integer :: i,j,icount
    real(8) :: Vav,error,dV

    icount = 0
    do while (.TRUE.)
        error = 0.0D0
        do j=2,L-1
            do i=2,L-1
!We change the voltage only for non conductors:
                if( .NOT. isConductor(i,j))then
                    Vav = (V(i-1,j)+V(i+1,j)+V(i,j+1)+V(i,j-1)+rho(i,j))&
                        *0.25D0
                    dV = DABS(V(i,j)-Vav)
                    if(error .LT. dV ) error = dV !maximum error
                    V(i,j) = Vav
                endif
            enddo
        enddo
        icount = icount + 1
        if( error .LT. epsilon) exit
    enddo
    print *,icount,' err= ',error

end subroutine laplace
!*****
subroutine print_results(V,L)
!*****
    implicit none
    integer :: L
    real(8),dimension(L,L) :: V
    integer :: i,j

    open(unit=11,file="data")
    do i=1,L
        do j =1,L
            write(11,*)i,j,V(i,j)
        enddo
    enddo

```

```

write (11,*) ' ' !empty line for gnuplot, separate isolines
enddo

end subroutine print_results

```

Στη βιβλιογραφία ο παραπάνω αλγόριθμος αναφέρεται ως η μέθοδος Gauss–Seidel. Χαρακτηρίζεται από το ότι στο δεξί μέλος της εξίσωσης (7.14) χρησιμοποιούμε τις ήδη ενημερωμένες τιμές για το δυναμικό  $V(i, j)$  και ενημερώνουμε αμέσως τη νέα τιμή  $V(i, j)$ . Μια άλλη μέθοδος είναι η μέθοδος του Jacobi όπου στο δεξί μέλος χρησιμοποιούμε τις παλιές τιμές του δυναμικού από το προηγούμενο πέρασμα του πλέγματος. Η μέθοδος Gauss–Seidel υπερέχει της Jacobi ως προς την ταχύτητα σύγκλισης. Μπορούμε όμως να γενικεύσουμε τη μέθοδο Jacobi ως εξής. Ορίζουμε το υπολειπόμενο (residual) της εξίσωσης (7.14) να είναι

$$R_{i,j} = V(i+1, j) + V(i-1, j) + V(i, j+1) + V(i, j-1) - 4V(i, j) + \tilde{\rho}(i, j), \quad (7.15)$$

το οποίο μηδενίζεται, όταν  $V(i, j)$  είναι η λύση της (7.14). Τότε, χρησιμοποιώντας τα  $R_{i,j}$  η μέθοδος Jacobi μπορεί να γραφτεί ως

$$V^{(n+1)}(i, j) = V^{(n)}(i, j) + \frac{1}{4}R_{i,j}^{(n)}, \quad (7.16)$$

όπου οι ποσότητες με άνω δείκτη  $(n)$  αναφέρονται στις τιμές που υπολογίζονται από τις τιμές που έχει το δυναμικό  $V^{(n)}$  στο  $n$ -οστό πέρασμα του πλέγματος. Η μέθοδος “successive overrelaxation” (SOR) δίνεται από τη γενίκευση της παραπάνω σχέσης:

$$V^{(n+1)}(i, j) = V^{(n)}(i, j) + \frac{\omega}{4}R_{i,j}^{(n)}. \quad (7.17)$$

Όταν η παράμετρος  $\omega < 1$ , έχουμε “underrelaxation” και η μέθοδος συγκλίνει πιο αργά από τη μέθοδο Jacobi. Όταν  $1 < \omega < 2$ , έχουμε “overrelaxation” και, αν η μέθοδος συγκλίνει, η κατάλληλη επιλογή της  $\omega$  μπορεί να οδηγήσει σε αισθητή βελτίωση σε σχέση με τη μέθοδο Jacobi. Όταν  $\omega > 2$ , η SOR αποκλίνει. Η μελέτη της μεθόδου SOR αφήνεται ως άσκηση στον αναγνώστη.

## 7.8 Ασκήσεις

- 7.1 Φτιάξτε τις εικόνες των δυναμικών και ισοδυναμικών γραμμών που βρίσκονται στην παράγραφο 7.2.
- 7.2 Στις προηγούμενες κατανομές φορτίων να κάνετε όλα τα φορτία θετικά. Φτιάξτε τις εικόνες των δυναμικών και ισοδυναμικών γραμμών. Μετά δώστε στα φορτία αυτά διαφορετική τιμή και επαναλάβετε.
- 7.3 Γιατί το πρόγραμμα ELines.f90 κολλάει σε κατανομή ίσων φορτίων πάνω σε κορυφές τετραγώνων; Πώς διορθώνεται η παθολογία αυτή;
- 7.4 Μεταβάλετε το πρόγραμμα ELines.f90, ώστε ο αριθμός των δυναμικών γραμμών που ξεκινάνε από ένα φορτίο  $q$  να είναι ανάλογος του  $q$ .
- 7.5 Βελτιώστε το πρόγραμμα EPotential.f90, έτσι ώστε οι ισοδυναμικές καμπύλες να ζωγραφίζονται με πυκνότητα ανάλογη του ηλεκτρικού πεδίου. Υπόδειξη:
- (α') Γράψτε υπορουτίνα που να υπολογίζει το δυναμικό  $V(x, y)$  στο σημείο  $(x, y)$ .
  - (β') Από κάθε φορτίο παίρνουμε μια ευθεία στην ακτινική διεύθυνση και υπολογίζουμε το δυναμικό πάνω σε αυτή σε σημεία που απέχουν μικρή απόσταση  $\Delta l$ .
  - (γ') Υπολογίζουμε τη μέγιστη/ελάχιστη τιμή του δυναμικού  $V_{max}/V_{min}$  και από εκεί τις τιμές του δυναμικού πάνω στις ισοδυναμικές καμπύλες που θα σχεδιάσουμε. Αν λ.χ. αποφασίσετε να σχεδιάσετε 5 ισοδ. καμπύλες, πάρτε  $\delta V = (V_{max} - V_{min})/4$  και  $V_i = V_{min} + i\delta V$   $i = 0, \dots, 4$ .
  - (δ') Επαναλάβετε το δεύτερο βήμα. Όταν το δυναμικό παίρνει τιμή (σχεδόν) ίση με μια από αυτές που διαλέξατε στο τρίτο βήμα, σχεδιάστε την ισοδυναμική καμπύλη.
- 7.6 Μελετήστε το ηλεκτρικό πεδίο που προκύπτει από το πρόγραμμα LaplaceEq.f90 για:
- (α')  $L=31, V1=100, V2=100$
  - (β')  $L=31, V1=100, V2=0$

και φτιάξτε τα αντίστοιχα σχήματα.

- 7.7 Μελετήστε το ηλεκτρικό πεδίο που προκύπτει από το πρόγραμμα LaplaceEq.f90 για:

(α')  $V_1=100, V_2=100$

(β')  $V_1=100, V_2=100$

(γ')  $V_1=100, V_2=0$

για  $L=31, 61, 121, 241, 501$  και φτιάξτε τα αντίστοιχα σχήματα. Μεταβάλετε την ακρίβεια προσδιορισμού της λύσης  $\epsilonpsilon=0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001$ . Πως εξαρτάται ο αριθμός περασμάτων  $N$  του πλέγματος από το  $\epsilonpsilon$ ; Φτιάξτε τη γραφική παράσταση  $N(\epsilonpsilon)$  στην οποία θα τοποθετήσετε τις καμπύλες  $N(\epsilonpsilon)$  για τα διαφορετικά  $L$  που χρησιμοποιήσατε.

- 7.8 Μελετήστε το ηλεκτρικό πεδίο τετράγωνου αγωγού του οποίου κάθε πλευρά έχει διαφορετικό δυναμικό  $V_1, V_2, V_3, V_4$ . Επαναλάβετε τη μελέτη που κάνατε στην προηγούμενη άσκηση για τις περιπτώσεις:

(α')  $V_1=10, V_2=5, V_3=10, V_4= 5$

(β')  $V_1=10, V_2=0, V_3=0, V_4= -10$

(γ')  $V_1=10, V_2=0, V_3=0, V_4= 0$

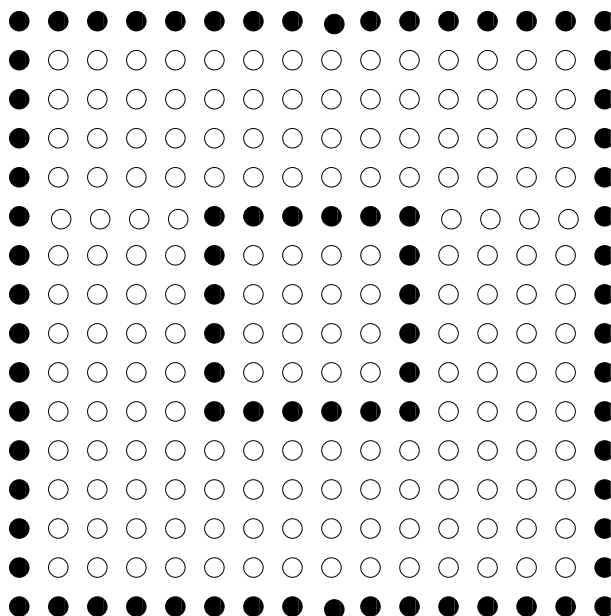
- 7.9 Μελετήστε το ηλεκτρικό πεδίο συστήματος τετράγωνων αγωγών που βρίσκεται ο ένας μέσα στον άλλο όπως στο σχήμα 7.11. Ο κάθε αγωγός έχει πλευρά  $L_1, L_2$  και βρίσκεται σε δυναμικό  $V_1, V_2$  αντίστοιχα. Πάρτε  $L_2= L_1/5$  και επαναλάβετε την μελέτη της προηγούμενης άσκησης για  $V_1=10, V_2=-10$  και  $L_1= 25, 50, 100, 200$ .

- 7.10 Υπολογίστε αριθμητικά τη χωρητικότητα  $C = Q/V$  του συστήματος της προηγούμενης άσκησης, όταν  $V_1 = V, V_2 = -V$ . Για να υπολογίσετε το φορτίο  $Q$  μπορείτε να υπολογίσετε την επιφανειακή πυκνότητα φορτίου  $\sigma$  από τη σχέση

$$\sigma = \frac{E_n}{4\pi},$$

όπου  $E_n$  η κάθετη συνιστώσα του ηλεκτρικού πεδίου στην επιφάνεια την οποία θα προσεγγίσετε από τη σχέση

$$E_n = -\frac{\delta V}{\delta r},$$



Σχήμα 7.11: Η διάταξη αγωγών που ζητείται να μελετηθεί το ηλεκτρικό πεδίο στην Άσκηση 7.9.

όπου  $\delta V$  είναι η διαφορά δυναμικού μεταξύ ενός σημείου του αγωγού και του σημείου που είναι πλησιέστερος γείτονάς του. Με τον τρόπο αυτό υπολογίστε το συνολικό φορτίο σε κάθε αγωγό. Αν αυτά είναι αντίθετα και κατ' απόλυτη τιμή ίσα με  $Q$  τότε προσδιορίζεται η χωρητικότητα  $C$ . Κάνετε τον παραπάνω υπολογισμό για  $V = 10$  και  $L=25, 75$ .

- 7.11 Στο σύστημα της προηγούμενης άσκησης μελετήστε τη συνάρτηση  $Q(V)$ . Βεβαιωθείτε ότι η χωρητικότητα είναι ανεξάρτητη της διαφοράς δυναμικού. Πάρτε  $L=25, 50$ ,  $V_1 = -V_2 = 1, 2, 5, 10, 15, 20, 25$ .
- 7.12 Αναπαράγετε τα σχήματα 7.8, 7.9 και 7.10. Στην πρώτη περίπτωση, συγκρίνετε τη λύση που θα πάρετε με αυτή που δίνεται από το δυναμικό σημειακού ηλεκτρικού φορτίου.
- 7.13 Εισάγετε το μήκος της πλεγματικής σταθεράς στις αντίστοιχες εξισώσεις στο πρόγραμμα PoissonEq.f90. Καθορίστε το μήκος της πλευράς να είναι  $l = 1$  και τυπώστε τα αποτελέσματά σας στο αρχείο data ως  $(x_i, y_i, V(x_i, y_i))$  αντί για  $(i, j, V(i, j))$ . Επαναλάβετε την προηγούμενη άσκηση για  $L=51, 101, 151, 201, 251$  και φτιάξτε

τις γραφικές παραστάσεις  $V(x, y)$  στο τετράγωνο  $0 < x < 1$ ,  $0 < y < 1$ . Στη συνέχεια μελετήστε τη σύγκλιση των λύσεων που παίρνετε κάνοντας τις γραφικές παραστάσεις  $V(x, 1/2)$  για τα  $L$  που επιλέξατε.

- 7.14 Γράψτε πρόγραμμα που να υλοποιεί τον αλγόριθμό SOR που περιγράφεται από τη σχέση (7.16) για το πρόβλημα που λύσαμε στο πρόγραμμα LaplaceEq.f90. Συγκρίνετε την ταχύτητα σύγκλισης της μεθόδου για  $L = 51$ ,  $\omega = 1.0, 0.9, 0.8, 0.6, 0.4, 0.2$  με αυτή της Gauss-Seidel. Τι γίνεται όταν  $\omega > 1$ ;
- 7.15 Γράψτε πρόγραμμα που να υλοποιεί τον αλγόριθμό SOR που περιγράφεται από τη σχέση (7.16) για το πρόβλημα που λύσαμε στο πρόγραμμα PoissonEq.f90. Συγκρίνετε την ταχύτητα σύγκλισης της μεθόδου για  $L = 51$ ,  $\omega = 1.0, 0.9, 0.8, 0.6, 0.4, 0.2$  με αυτή της Gauss-Seidel. Τι γίνεται όταν  $\omega > 1$ ;

## ΚΕΦΑΛΑΙΟ 8

# Εξίσωση Διάχυσης

### 8.1 Εισαγωγή

Η εξίσωση διάχυσης είναι στενά συνδεδεμένη με τη διαδρομή ενός τυχαίου περιπατητή (random walker). Ας υποθέσουμε ότι μελετάμε την κίνηση ενός τέτοιου σωματίου πάνω στην ευθεία (“μία διάσταση”). Η διαδικασία της κίνησης είναι στοχαστική και η συνάρτηση (“πυρήνας”)

$$K(x, x_0; t) \quad (8.1)$$

ερμηνεύεται ως η πυκνότητα πιθανότητας τη χρονική στιγμή  $t$  να παρατηρηθεί το σωματίο στη θέση  $x$  αν τη χρονική στιγμή  $t = 0$  το σωματίο βρίσκεται στη θέση  $x_0$ . Η εξίσωση που καθορίζει το  $K(x, x_0; t)$  είναι η

$$\frac{\partial K(x, x_0; t)}{\partial t} = D \frac{\partial^2 K(x, x_0; t)}{\partial x^2}, \quad (8.2)$$

που είναι η εξίσωση διάχυσης. Ο συντελεστής διάχυσης  $D$  μπορεί να καθοριστεί από τις λεπτομέρειες του συστήματος που μελετάμε. Για την κίνηση Brown ενός σωματιδίου σκόνης μέσα σε ένα υγρό, το οποίο κινείται με την επίδραση των τυχαίων θερμικών κρούσεων με τα μόρια του υγρού παίρνουμε  $D = kT/\gamma$ , όπου  $T$  είναι η απόλυτη θερμοκρασία του υγρού,  $\gamma$  ο συντελεστής τριβής<sup>1</sup> του σωματιδίου μέσα στο υγρό και  $k$  είναι η σταθερά του Boltzmann.

Συνήθως επιλέγουμε για αρχικές συνθήκες ( $t = 0$ ) το σωματίο να είναι εντοπισμένο σε ένα σημείο  $x_0$ , δηλ.<sup>2</sup>

$$K(x, x_0; 0) = \delta(x - x_0) \quad (8.3)$$

---

<sup>1</sup>Για ένα σφαιρικό σωματίο ακτίνας  $R$  μέσα σε ένα Νευτώνειο υγρό με ιξώδες  $\eta$  έχουμε ότι  $\gamma = 6\pi\eta R$ .

<sup>2</sup>Θυμίζουμε ότι  $\delta(x - x_0)$  είναι το περίφημο δέλτα του Dirac. Ορίζεται από την

Η ερμηνεία της  $K(x, x_0; t)$  ως συνάρτηση πυκνότητας πιθανότητας συνεπάγεται ότι για κάθε  $t$  θα πρέπει να έχουμε<sup>3</sup>

$$\int_{-\infty}^{+\infty} K(x, x_0; t) dx = 1. \quad (8.4)$$

Αυτή η σχέση δεν είναι προφανές ότι μπορεί να ισχύει για κάθε χρονική στιγμή. Ακόμα και αν την επιβάλλουμε για  $t = 0$ , η χρονική εξέλιξη που καθορίζεται από την (8.2) μπορεί να την αλλάξει σε μεγαλύτερους χρόνους.

Αυτό είναι εύκολο να αναλυθεί. Αν επιβάλλουμε την (8.4) όταν  $t = 0$ , η συνθήκη θα ισχύει για κάθε χρονική στιγμή αν

$$\frac{d}{dt} \int_{-\infty}^{+\infty} K(x, x_0; t) dx = 0. \quad (8.5)$$

Λαμβάνοντας υπόψη ότι  $\frac{d}{dt} \int_{-\infty}^{+\infty} K(x, x_0; t) dx = \int_{-\infty}^{+\infty} \frac{\partial K(x, x_0; t)}{\partial t} dx$  και ότι  $\frac{\partial K(x, x_0; t)}{\partial t} = D \frac{\partial^2 K(x, x_0; t)}{\partial x^2}$  παίρνουμε

$$\begin{aligned} \frac{d}{dt} \int_{-\infty}^{+\infty} K(x, x_0; t) dx &= D \int_{-\infty}^{+\infty} \frac{\partial}{\partial x} \left( \frac{\partial K(x, x_0; t)}{\partial x} \right) dx \\ &= D \left. \frac{\partial K(x, x_0; t)}{\partial x} \right|_{x \rightarrow +\infty} - D \left. \frac{\partial K(x, x_0; t)}{\partial x} \right|_{x \rightarrow -\infty}. \end{aligned} \quad (8.6)$$

Η παραπάνω σχέση μας λέει πως για συναρτήσεις που το δεξί μέλος μηδενίζεται, η συνθήκη κανονικοποίησης μπορεί να επιβληθεί για όλες τις χρονικές στιγμές  $t > 0$ .

Η προσεκτική ανάλυση της εξίσωσης (8.2) δίνει ότι, για μικρούς χρόνους, η ασυμπτωτική συμπεριφορά του  $K(x, x_0; t)$  είναι

$$K(x, x_0; t) \sim \frac{e^{-\frac{|x-x_0|^2}{4Dt}}}{t^{d/2}} \sum_{i=0}^{\infty} a_i(x, x_0) t^i. \quad (8.7)$$

Η σχέση αυτή δείχνει πως η διάχυση είναι ισότροπη (ίδια προς όλες τις κατευθύνσεις) και η πιθανότητα ανίχνευσης ελαττώνεται δραστηκά

---

απαίτηση για κάθε συνάρτηση  $f(x)$  να έχουμε  $\int_{-\infty}^{+\infty} f(x) \delta(x - x_0) dx = f(x_0)$ . Προφανώς τότε έχουμε ότι  $\int_{-\infty}^{+\infty} \delta(x - x_0) dx = 1$ . Μπορεί κανείς να το φανταστεί σαν μια συνάρτηση που είναι πρακτικά μηδέν παντού, εκτός από μια απειροστή περιοχή γύρω από το  $x_0$ .

<sup>3</sup>Εναλλακτικά, αν η  $K(x, x_0; t)$  δίνει λ.χ. την πυκνότητα μάζας μιας σταγόνας μελανιού μάζας  $m_{ink}$  που διαχέεται μέσα σε ένα διαφανές υγρό, θα έχουμε  $\int_{-\infty}^{+\infty} K(x, x_0; t) dx = m_{ink}$  και  $K(x, x_0; 0) = m_{ink} \delta(x - x_0)$ .



με την απόσταση από την αρχική θέση του σωματιδίου. Αυτή η σχέση δεν μπορεί να ισχύει για πάντα, αφού για αρκετά μεγάλους χρόνους το σωματίο κατανέμεται ομοιόμορφα μέσα στο χώρο<sup>4</sup>.

Η πιθανότητα επιστροφής του σωματιδίου στην αρχική του θέση ορίζεται να είναι

$$P_R(t) = K(x_0, x_0; t) \sim \frac{1}{t^{d/2}} \sum_{i=0}^{\infty} a_i(x_0, x_0) t^i \quad (8.8)$$

που ορίζει τη φασματική διάσταση  $d$  του χώρου. Στην περίπτωση που μελετάμε  $d = 1$ .

Η μέση τιμή του τετραγώνου της απόστασης από την αρχή των αξόνων που βρίσκεται το σωματίδιο σε χρόνο  $t$  είναι εύκολο να υπολογιστεί<sup>5</sup>

$$\langle r^2 \rangle = \langle (x - x_0)^2 \rangle(t) = \int_{-\infty}^{+\infty} (x - x_0)^2 K(x, x_0; t) dx \sim 2Dt. \quad (8.9)$$

Η τελευταία σχέση είναι πολύ σημαντική. Μας λέει πως η κίνηση του τυχαίου περιπατητή (κίνηση Brown) δεν μπορεί να έχει κλασική περιγραφή αλλά μόνο στοχαστική: Για ένα κλασικό σωματίο που κινείται πάνω σε μια ομαλή τροχιά  $x - x_0 \sim vt$  άρα  $r^2 \sim t^2$ .

Στα επόμενα κεφάλαια, για απλότητα παίρνουμε<sup>6</sup>  $D = 1$  και ορίζουμε

$$u(x, t) \equiv K(x - x_0, x_0; t). \quad (8.10)$$

## 8.2 Απαγωγή Θερμότητας

Έστω μια λεπτή ευθύγραμμη ράβδος μήκους  $L$  και  $T(x, t)$  η κατανομή της θερμοκρασίας της τη χρονική στιγμή  $t$ , και έστω ότι τα άκρα της τα κρατάμε σε σταθερή θερμοκρασία  $T(0, t) = T(L, t) = T_0$ . Αν η αρχική κατανομή της θερμοκρασίας είναι  $T(x, 0)$ , η θερμοκρασία σε κάθε άλλη χρονική στιγμή προσδιορίζεται από την εξίσωση διάχυσης

$$\frac{\partial T(x, t)}{\partial t} = \alpha \frac{\partial^2 T(x, t)}{\partial x^2} \quad (8.11)$$

όπου  $\alpha = k/(c_p \rho)$  ο θερμικός συντελεστής διάχυσης (thermal diffusivity),  $k$  η θερμική αγωγιμότητα,  $\rho$  η πυκνότητα και  $c_p$  η ειδική θερμότητα της ράβδου.

<sup>4</sup>Θυμηθείτε την αναλογία με τη σταγόνα μελανιού που διαχέεται μέσα σε ένα ποτήρι νερό και μετά από αρκετό χρόνο έχει διαχυθεί ομοιόμορφα μέσα στο νερό.

<sup>5</sup> $\int_0^\infty dr r^n e^{-r^2/4Dt} = 2^n \Gamma(\frac{n+1}{2}) (Dt)^{\frac{n+1}{2}}$ .

<sup>6</sup>Αυτό σύμφωνα με την (8.2) αντιστοιχεί στο να πάρουμε  $t \rightarrow Dt$ .

Ορίζουμε

$$u(x, t) = \frac{T(xL, \frac{L^2}{\alpha}t) - T_0}{T_0}, \quad (8.12)$$

όπου  $x \in [0, 1]$ . Με τον ορισμό αυτό, η συνάρτηση  $u(x, t)$  είναι καθαρός αριθμός (αδιάστατη) και εκφράζει το κλάσμα της διαφοράς θερμοκρασίας σε σχέση με αυτής των άκρων της ράβδου και

$$u(0, t) = u(1, t) = 0. \quad (8.13)$$

Αυτές λέγονται συνοριακές συνθήκες τύπου Dirichlet<sup>7</sup>

Η (8.11) γίνεται

$$\frac{\partial u(x, t)}{\partial t} = \frac{\partial^2 u(x, t)}{\partial x^2} \quad (8.14)$$

Η σχέση (8.6) γίνεται

$$\frac{d}{dt} \int_0^1 u(x, t) dx = \left. \frac{\partial u}{\partial x} \right|_{x=1} - \left. \frac{\partial u}{\partial x} \right|_{x=0} \quad (8.15)$$

Η παραπάνω σχέση δεν μπορεί να δίνει πάντα 0 λόγω των συνοριακών συνθηκών (8.13). Αυτό μπορούμε να το δούμε με ένα παράδειγμα. Έστω

$$u(x, 0) = \sin(\pi x), \quad (8.16)$$

τότε μπορείτε εύκολα να επιβεβαιώσετε ότι ικανοποιούνται οι απαιτούμενες συνοριακές συνθήκες και ότι η συνάρτηση

$$u(x, t) = \sin(\pi x)e^{-\pi^2 t}, \quad (8.17)$$

είναι η ζητούμενη λύση της εξίσωσης διάχυσης που ικανοποιεί επίσης τις συνοριακές συνθήκες. Είναι εύκολο να διαπιστώσετε ότι το

$$\int_0^1 u(x, t) dx = \frac{2}{\pi} e^{-\pi^2 t}$$

τείνει εκθετικά γρήγορα στο μηδέν με το χρόνο και ότι

$$\frac{d}{dt} \int_0^1 u(x, t) dx = -2\pi e^{-\pi^2 t},$$

σε συμφωνία με τις σχέσεις (8.15).

Η εκθετική πτώση του μέτρου της  $u(x, t)$  είναι σε συμφωνία με την φυσική απαίτηση ότι η ράβδος σε αρκετά μεγάλο χρόνο θα έχει ομοιόμορφη θερμοκρασία, ίση με αυτή που επιβάλαμε στα άκρα της ( $\lim_{t \rightarrow +\infty} u(x, t) = 0$ ).

<sup>7</sup>Αν προσδιορίζαμε τις παραγώγους  $\partial u / \partial x$  στα άκρα (λ.χ. ταλάντωση ελεύθερης ράβδου), θα είχαμε συνοριακές συνθήκες τύπου Neumann.

### 8.3 Διακριτοποίηση

Η αριθμητική λύση της εξίσωσης (8.14) θα αναζητηθεί στο διάστημα  $x \in [0, 1]$  και  $t \in [0, t_f]$ . Το πρόβλημα πρέπει να οριστεί πάνω σε ένα διακριτό πλέγμα και η διαφορική εξίσωση να προσεγγιστεί από αλγεβρικές εξισώσεις πεπερασμένων διαφορών.

Το πλέγμα ορίζεται από  $N_x$  χωρικά σημεία  $x_i \in [0, 1]$

$$x_i = 0 + (i - 1)\Delta x \quad i = 1, \dots, N_x, \quad (8.18)$$

όπου τα  $N_x - 1$  διαστήματα έχουν σταθερό πλάτος

$$\Delta x = \frac{1 - 0}{N_x - 1}, \quad (8.19)$$

και από  $N_t$  χρονικά πλεγματικά σημεία  $t_j \in [0, t_f]$

$$t_j = 0 + (j - 1)\Delta t \quad j = 1, \dots, N_t, \quad (8.20)$$

όπου τα  $N_t - 1$  διαστήματα έχουν σταθερό πλάτος

$$\Delta t = \frac{t_f - 0}{N_t - 1}. \quad (8.21)$$

Σημειώνουμε ότι τα άκρα των διαστημάτων αντιστοιχούν στα

$$x_1 = 0, \quad x_{N_x} = 1, \quad t_1 = 0, \quad t_{N_t} = t_f. \quad (8.22)$$

Η συνάρτηση  $u(x, t)$  προσεγγίζεται από τις τιμές της πάνω στο διακριτό  $N_x \times N_t$  πλέγμα

$$u_{i,j} \equiv u(x_i, t_j). \quad (8.23)$$

Οι παράγωγοι διακριτοποιούνται σύμφωνα με τις σχέσεις

$$\frac{\partial u(x, t)}{\partial t} \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} \equiv \frac{1}{\Delta t} (u_{i,j+1} - u_{i,j}), \quad (8.24)$$

$$\begin{aligned} \frac{\partial^2 u(x, t)}{\partial x^2} &\approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{(\Delta x)^2} \\ &\equiv \frac{1}{(\Delta x)^2} (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}). \end{aligned} \quad (8.25)$$

Εξισώνοντας τα δύο μέλη των παραπάνω σχέσεων σύμφωνα με την (8.14), παίρνουμε τη δυναμική εξέλιξη της  $u_{i,j}$  στο χρόνο

$$u_{i,j+1} = u_{i,j} + \frac{\Delta t}{(\Delta x)^2} (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}). \quad (8.26)$$

Αυτή είναι μια επαγωγική σχέση ενός βήματος ως προς το χρόνο. Αυτό είναι πολύ σημαντικό γιατί δε χρειάζεται στο πρόγραμμα να αποθηκεύσουμε στη μνήμη τις τιμές  $u_{i,j}$  για κάθε  $j$ .

Ο δεύτερος όρος της “δεύτερης παραγώγου” στην (8.26) περιέχει μόνο τους πλησιέστερους γείτονες  $u_{i\pm 1,j}$  κάθε πλεγματοειδούς σημείου  $u_{i,j}$  μιας χρονικής φέτας  $t_j$  του πλέγματος, άρα μπορεί να χρησιμοποιηθεί για κάθε  $i = 2, \dots, N_x - 1$ . Για τα σημεία  $i = 1$  και  $i = N_x$  δε χρειάζεται να χρησιμοποιηθούν οι σχέσεις (8.26), αφού κρατάμε τις τιμές  $u_{1,j} = u_{N_x,j} = 0$  αμετάβλητες.

Τέλος, η παράμετρος

$$\frac{\Delta t}{(\Delta x)^2} \quad (8.27)$$

είναι αυτή που καθορίζει τη χρονική εξέλιξη στον αλγόριθμο. Ονομάζεται παράμετρος του Courant και για να έχουμε χρονική εξέλιξη χωρίς να παρουσιάζονται γρήγορα αστάθειες, θα πρέπει

$$\frac{\Delta t}{(\Delta x)^2} < \frac{1}{2}. \quad (8.28)$$

Αυτό είναι κάτι που εμείς θα το ελέγξουμε εμπειρικά με την αριθμητική ανάλυση που θα κάνουμε.

## 8.4 Το Πρόγραμμα

Τα μόνα σημεία που τονίζουμε σχετικά με το σχεδιασμό του προγράμματος είναι ότι η σχέση (8.26) είναι μια επαγωγική σχέση ενός βήματος ως προς το χρόνο. Άρα, σε κάθε χρονικό βήμα αρκεί να αποθηκεύσουμε σε ένα array τις τιμές του δεύτερου όρου (τη “δεύτερη παράγωγο”) και να το χρησιμοποιήσουμε για να ενημερώσουμε τις νέες τιμές της συνάρτησης  $u_{i,j}$ . Άρα, στην επαναλαμβανόμενη διαδικασία (8.26) υπολογισμού της  $u_{i,j+1}$  από την  $u_{i,j}$  αρκεί να χρησιμοποιήσουμε μονάχα ένα array  $u_i$ ,  $i = 1, \dots, N_x$  και ένα  $(\partial^2 u / \partial x^2)_i$ ,  $i = 1, \dots, N_x$  που δίνουν τις αντίστοιχες τιμές της  $u_{i,j}$  και  $\Delta t / (\Delta x)^2 (u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$  τη χρονική στιγμή  $t_j$  αντίστοιχα. Στο παρακάτω πρόγραμμα αυτά κωδικοποιούνται στα arrays  $u(P)$  και  $d2udx2(P)$ .

Τα χρήσιμα δεδομένα βρίσκονται στις θέσεις  $u(1) \dots u(N_x)$   $d2udx2(1) \dots d2udx2(N_x)$  και η παράμετρος  $P$  επιλέγεται αρκετά μεγάλη, ώστε οι τιμές του  $N_x$  που θα μελετηθούν να είναι πάντα μικρότερες.

Ο χρήστης δίνει στην είσοδο τις τιμές  $N_x = Nx$ ,  $N_t = Nt$ ,  $t_f = tf$ . Οι τιμές  $\Delta x$ ,  $\Delta t$  και  $\Delta t / \Delta x^2 = \text{courant}$  υπολογίζονται στα αρχικά στάδια του προγράμματος.

Στην έξοδο παίρνουμε το αρχείο d.dat που περιέχει σε στήλες τις τιμές  $(t_j, x_i, u_{i,j})$ . Όταν τελειώνει μια χρονική φέτα  $t_j$ , το πρόγραμμα τυπώνει μια κενή γραμμή, έτσι ώστε το gnuplot να κάνει αμέσως την τρισδιάστατη γραφική παράσταση.

Το πρόγραμμα μπορεί να βρεθεί στο αρχείο diffusion.f90 στο συνοδευτικό λογισμικό και ο κώδικας που περιέχει δίνεται παρακάτω:

```
!=====
! 1-dimensional Diffusion Equation with simple
! Dirichlet boundary conditions u(0,t)=u(1,t)=0
! 0<= x <= 1 and 0<= t <= tf
!
! We set initial condition u(x,t=0) that satisfies
! the given boundary conditions.
! Nx is the number of points in spatial lattice:
! x = 0 + (j-1)*dx, j=1,...,Nx and dx = (1-0)/(Nx-1)
! Nt is the number of points in temporal lattice:
! t = 0 + (j-1)*dt, j=1,...,Nt and dt = (tf-0)/(Nt-1)
!
! u(x,0) = sin(pi*x) tested against analytical solution
! u(x,t) = sin(pi*x)*exp(-pi*pi*t)
!
!=====
program diffusion_1d
  implicit none
  integer,parameter :: P =100000 ! Max no of points
  real(8),parameter :: PI=3.1415926535897932D0
  real(8),dimension(P) :: u, d2udx2
  real(8) :: t,x,dx,dt,tf,courant
  integer Nx,Nt,i,j
! --- Input:
  print *, '# Enter: Nx, Nt, tf: (P= ',P,' Nx must be < P)'
  read *, Nx,Nt,tf
  if(Nx .ge. P) stop 'Nx >= P'
  if(Nx .le. 3) stop 'Nx <= 3'
  if(Nt .le. 2) stop 'Nt <= 2'
! --- Initialize:
  dx = 1.0D0/(Nx-1)
  dt = tf/(Nt-1)
  courant = dt/dx**2
  print *, '# 1d Diffusion Equation: 0<=x<=1, 0<=t<=tf'
  print *, '# dx= ',dx,' dt= ',dt,' tf= ',tf
  print *, '# Nx= ',Nx,' Nt= ',Nt
  print *, '# Courant Number= ',courant
  if(courant .gt. 0.5D0) print *, '# WARNING: courant > 0.5'
  open(unit=11,file='d.dat') ! data file
! --- Initial condition at t=0
```

```

!u(x,0) = sin( pi x)
do i= 1, Nx
  x    = (i-1)*dx
  u(i) = sin(PI*x)
enddo
u(1)  = 0.0d0
u(Nx) = 0.0d0
do i= 1,Nx
  x    = (i-1)*dx
  write(11,*) 0.0D0, x, u(i)
enddo
write(11,*) ' '
!
! ----- Calculate time evolution:
do j=2,Nt
  t = (j-1)*dt
! ----- second derivative:
  do i=2,Nx-1
    d2udx2(i) = courant*(u(i+1)-2.0D0*u(i)+u(i-1))
  enddo
! ----- update:
  do i=2,Nx-1
    u(i) = u(i) + d2udx2(i)
  enddo
  do i=1,Nx
    x = (i-1)*dx
    write(11,*) t, x, u(i)
  enddo
  write(11,*) ' '
enddo ! do j=2,Nt

close(11)
end program diffusion_1d

```

## 8.5 Αποτελέσματα

Αρχικά γίνεται η μεταγλώττιση και το τρέξιμο του προγράμματος

```

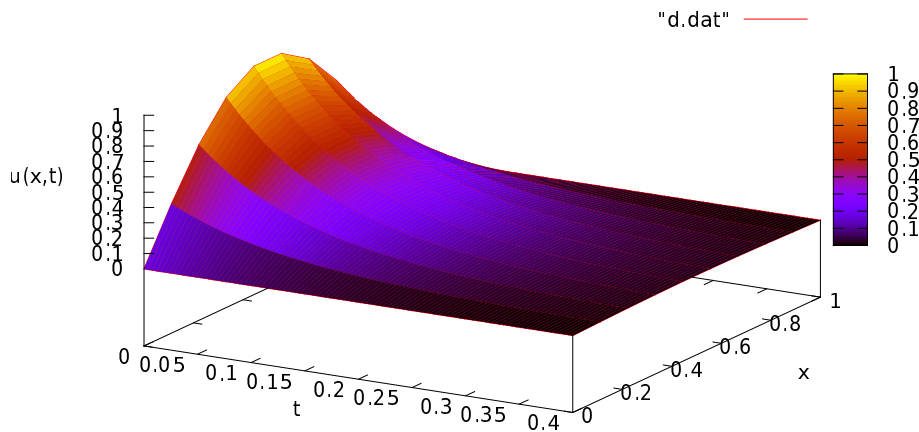
> gfortran diffusion.f90 -o d
> echo "10 100 0.4" | ./d
# Enter: Nx, Nt, tf: (P= 100000 Nx must be < P)
# 1d Diffusion Equation: 0<=x<=1, 0<=t<=tf
# dx= 0.11111111111111110 dt= 4.04040404040404040E-3 tf= 0.4
# Nx= 10 Nt= 100
# Courant Number= 0.32727272727272733

```

Στη δεύτερη σειρά, εισάγουμε στο stdin του προγράμματος τις τιμές  $N_x=10$ ,  $N_t=100$ ,  $tf=0.4$  από το stdout της εντολής `echo`. Οι επόμενες γραμμές είναι το output του προγράμματος.

Στη συνέχεια, μπορούμε να κάνουμε μια τρισδιάστατη γραφική αναπαράσταση της  $u(x,t)$  με τη βοήθεια του `gnuplot`:

```
gnuplot> set pm3d
gnuplot> set hidden3d
gnuplot> splot "d.dat" with lines
gnuplot> unset pm3d
```



Σχήμα 8.1: Η συνάρτηση  $u(x,t)$  για  $N_x=10$ ,  $N_t=100$ ,  $tf=0.4$ .

Στη συνέχεια, θέλουμε να δούμε τη συνάρτηση  $u(x,t)$  ως συνάρτηση του  $x$  για δεδομένες τιμές του χρόνου. Παρατηρούμε ότι ο χρόνος αλλάζει κάθε φορά που συναντάμε μια κενή γραμμή στο αρχείο `d.dat`. Το παρακάτω πρόγραμμα `awk` μετράει τις κενές γραμμές και τυπώνει μόνο εκείνη που εμείς επιθυμούμε. Ο μετρητής  $n=0, 1, \dots, N_t-1$  μπορεί να καθορίσει την τιμή του  $t_j = t_{n-1}$ . Τα αποτελέσματα τα σώζουμε σε ένα αρχείο `tj` το οποίο μπορούμε να το δούμε με το `gnuplot`. Επαναλαμβάνουμε όσες φορές χρειάζεται:

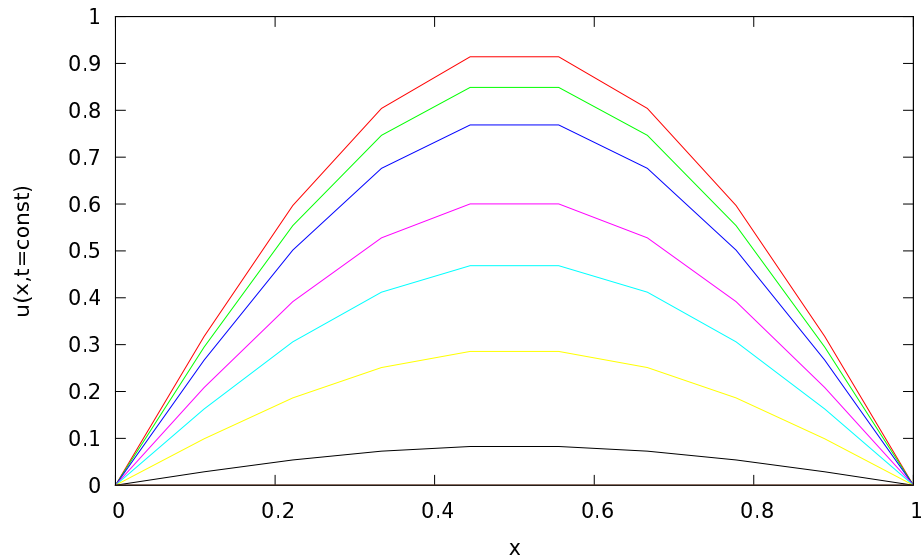
```
> awk 'NF<3{n++}n==3 {print}' d.dat > tj
gnuplot> plot "tj" using 2:3 with lines
```

Την παραπάνω εργασία μπορούμε να την κάνουμε χωρίς τη δημιουργία ενδιάμεσων αρχείων `tj` χρησιμοποιώντας το φίλτρο της `awk` μέσα από

το gnuplot. Έτσι, για παράδειγμα, οι εντολές

```
gnuplot> ! echo "10 800 2" | ./d
gnuplot> plot "<awk 'NF<3{n++}n==3 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==6 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==10 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==20 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==30 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==50 {print}' d.dat" u 2:3 w l
gnuplot> replot "<awk 'NF<3{n++}n==100{print}' d.dat" u 2:3 w l
```

τρέχουν το πρόγραμμα για  $N_x=10$ ,  $N_t=800$ ,  $t_f=2$  και παράγουν το σχήμα 8.2



Σχήμα 8.2: Η συνάρτηση  $u(x, t)$  για  $N_x=10$ ,  $N_t=800$ ,  $t_f=2$  για διαφορετικές σταθερές τιμές του χρόνου  $t_j$ . Εδώ  $j = 4, 7, 11, 21, 31, 51, 101$  όπου η  $u(x, t)$  φθίνει όταν αυξάνει το  $j$ .

Στη συνέχεια είναι ενδιαφέρον να συγκρίνει κανείς τα αποτελέσματα του με την ακριβή λύση  $u(x, t) = \sin(\pi x)e^{-\pi^2 t}$ . Ένας τρόπος να γίνει είναι να ορίσουμε το σχετικό σφάλμα

$$\frac{u_{i,j} - u(x_i, t_j)}{u_{i,j}},$$

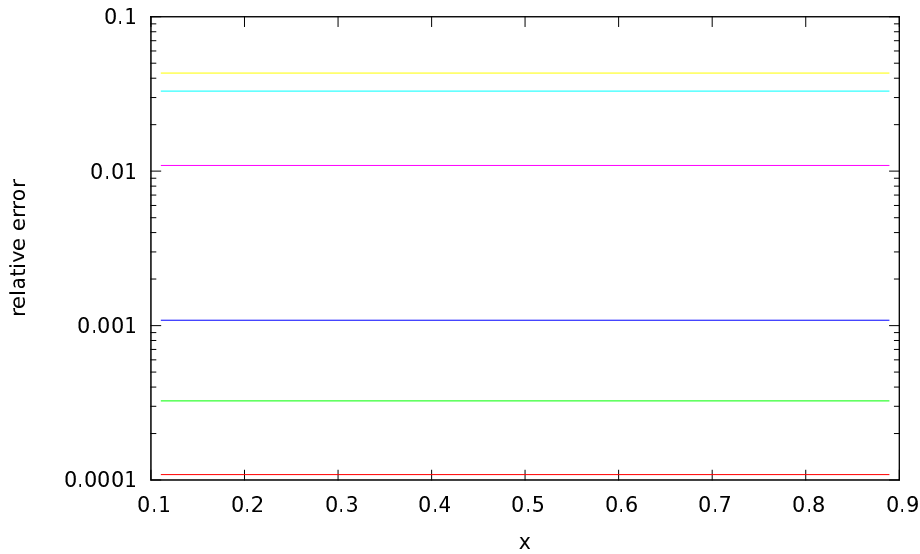
και να το υπολογίσουμε ορίζοντας τη σχετική συνάρτηση μέσα στο gnuplot:



```

gnuplot> du(x,y,z) = (z - sin(pi*x)*exp(-pi*pi*y))/z
gnuplot> plot "<awk 'NF<3{n++}n==2 ' d.dat" u 2:(du($2,$1,$3))
gnuplot> plot "<awk 'NF<3{n++}n==6 ' d.dat" u 2:(du($2,$1,$3))
gnuplot> plot "<awk 'NF<3{n++}n==20 ' d.dat" u 2:(du($2,$1,$3))
gnuplot> plot "<awk 'NF<3{n++}n==200 ' d.dat" u 2:(du($2,$1,$3))
gnuplot> plot "<awk 'NF<3{n++}n==600 ' d.dat" u 2:(du($2,$1,$3))
gnuplot> plot "<awk 'NF<3{n++}n==780 ' d.dat" u 2:(du($2,$1,$3))

```



Σχήμα 8.3: Η απόλυτη τιμή του σχετικού σφάλματος του αριθμητικού υπολογισμού για  $N_x=10$ ,  $N_t=800$ ,  $t_f=2$  για διαφορετικές σταθερές τιμές του χρόνου  $t_j$ . Εδώ  $j = 3, 7, 21, 201, 601, 781$  και το σχετικό σφάλμα αυξάνει με το  $j$ .

Τα αποτελέσματα μπορούμε να τα δούμε στο σχήμα 8.3.

## 8.6 Διάχυση Πάνω στον Κύκλο.

Για να μελετήσουμε τον πυρήνα  $K(x, x_0; t)$  στο πρόβλημα της διάχυσης ή των τυχαίων διαδρομών, πρέπει να επιβάλλουμε τη συνθήκη κανονικοποίησης (8.4) για κάθε χρονική στιγμή. Στην περίπτωση της  $u(x, t)$  ορισμένης για  $x \in [0, 1]$  η σχέση γίνεται

$$\int_0^1 u(x, t) dx = 1, \quad (8.29)$$

η οποία για να ισχύει για κάθε χρονική στιγμή είναι αναγκαίο το δεξί μέλος της (8.15) να είναι 0. Ένας τρόπος να επιβάλλουμε αυτή τη συνθήκη

είναι να θεωρήσουμε το πρόβλημα της διάχυσης πάνω στον κύκλο. Αν παραμετροποιήσουμε τα σημεία του κύκλου με τη μεταβλητή  $x \in [0, 1]$ , τότε τα σημεία  $x = 0$  και  $x = 1$  ταυτίζονται και έχουμε

$$u(0, t) = u(1, t), \quad \frac{\partial u(0, t)}{\partial x} = \frac{\partial u(1, t)}{\partial x}. \quad (8.30)$$

Η δεύτερη από τις παραπάνω σχέσεις μηδενίζει το δεξί μέλος της (8.15) με αποτέλεσμα, αν θέσουμε  $\int_0^1 u(x, 0) dx = 1$ , να έχουμε  $\int_0^1 u(x, t) dx = 1$ ,  $\forall t > 0$ .

Με τις παραπάνω παραδοχές, η διακριτοποίηση της διαφορικής εξίσωσης γίνεται ακριβώς όπως και στο πρόβλημα της απαγωγής της θερμότητας. Αντί τώρα να κρατάμε τις τιμές  $u(0, t) = u(1, t) = 0$  σταθερές, θα εφαρμόσουμε την εξίσωση δυναμικής εξέλιξης (8.26) και για τα σημεία  $x_1, x_{N_x}$  αφού πάνω στον κύκλο αυτά τα σημεία δεν ξεχωρίζουν από τα υπόλοιπα. Για να λάβουμε υπόψη την κυκλική τοπολογία αρκεί να πάρουμε

$$u_{1,j+1} = u_{1,j} + \frac{\Delta t}{(\Delta x)^2} (u_{2,j} - 2u_{1,j} + u_{N_x,j}), \quad (8.31)$$

και

$$u_{N_x,j+1} = u_{N_x,j} + \frac{\Delta t}{(\Delta x)^2} (u_{1,j} - 2u_{N_x,j} + u_{N_x-1,j}), \quad (8.32)$$

αφού ο γείτονας εκ “δεξιών” του σημείου  $x_{N_x}$  είναι το σημείο  $x_1$  και ο γείτονας εξ “αριστερών” του σημείου  $x_1$  είναι το σημείο  $x_{N_x}$ . Για τα υπόλοιπα σημεία  $i = 2, \dots, N_x - 1$  η σχέση (8.26) εφαρμόζεται κανονικά.

Το πρόγραμμα που κωδικοποιεί το παραπάνω πρόβλημα δίνεται παρακάτω και βρίσκεται στο αρχείο diffusionS1.f90. Η επιβολή των συνοριακών συνθηκών (8.30) γίνεται στις γραμμές

```
nnr = i+1
if (nnr .gt. Nx) nnr = 1
nnl = i-1
if (nnl .lt. 1) nnl = Nx
d2udx2(i) = courant*(u(nnr)-2.0D0*u(i)+u(nnl))
```

Οι αρχικές συνθήκες τη χρονική στιγμή  $t = 0$  επιλέγονται έτσι, ώστε να είναι το σωματίο στη θέση  $x_{N_x/2}$ . Σε κάθε χρονική στιγμή γίνονται μετρήσεις με σκοπό να επαληθευτούν οι εξισώσεις (8.4), (8.9) και το γεγονός ότι  $\lim_{t \rightarrow +\infty} u(x, t) = \text{σταθ}$ .

Η μεταβλητή prob =  $\sum_{i=1}^{N_x} u_{i,j}$  και ελέγχεται αν διατηρεί την αρχική της τιμή που είναι ίση με 1.

Η μεταβλητή  $r2 = \sum_{i=1}^{N_x} (x_i - x_{N_x/2})^2 u_{i,j}$  είναι η διακριτή εκτίμηση της μέσης τιμής του τετραγώνου της απόστασης από την αρχική θέση η οποία για αρκετά μικρούς χρόνους θα πρέπει να ακολουθεί το νόμο που δίνει η εξίσωση (8.9).

Οι παραπάνω μεταβλητές αποθηκεύονται στο αρχείο e.dat μαζί με τις τιμές  $u_{N_x/2,j}$ ,  $u_{N_x/4,j}$  και  $u_{1,j}$ . Οι τελευταίες ελέγχονται αν μετά από αρκετά μεγάλο χρόνο αποκτούν την ίδια σταθερή τιμή, σύμφωνα με το αναμενόμενο αποτέλεσμα  $\lim_{t \rightarrow +\infty} u(x,t) = \text{σταθ}$ .

Όλος ο πηγαίος κώδικας είναι:

```
!=====
! 1-dimensional Diffusion Equation with
! periodic boundary conditions u(0,t)=u(1,t)
! 0<= x <= 1 and 0<= t <= tf
!
! We set initial condition u(x,t=0) that satisfies
! the given boundary conditions.
! Nx is the number of points in spatial lattice:
! x = 0 + (j-1)*dx, j=1,...,Nx and dx = (1-0)/(Nx-1)
! Nt is the number of points in temporal lattice:
! t = 0 + (j-1)*dt, j=1,...,Nt and dt = (tf-0)/(Nt-1)
!
! u(x,0) = \delta_{x,0.5}
!
!=====
program diffusion_1d
  implicit none
  integer,parameter :: P=100000 ! Max no of points
  real(8),parameter :: PI=3.1415926535897932D0
  real(8),dimension(P) :: u, d2udx2
  real(8) :: t,x,dx,dt,tf,courant,prob,r2,x0
  integer Nx,Nt,i,j,nnl,nnr
! --- Input:
  print *, '# Enter: Nx, Nt, tf: (P= ',P,' Nx must be < P)'
  read *, Nx,Nt,tf
  if(Nx .ge. P) stop 'Nx >= P'
  if(Nx .le. 3) stop 'Nx <= 3'
  if(Nt .le. 2) stop 'Nt <= 2'
! --- Initialize:
  dx = 1.0D0/(Nx-1)
  dt = tf/(Nt-1)
  courant = dt/dx**2
  print *, '# 1d Diffusion Equation on S1: 0<=x<=1, 0<=t<=tf'
  print *, '# dx= ',dx,' dt= ',dt,' tf= ',tf
  print *, '# Nx= ',Nx,' Nt= ',Nt
  print *, '# Courant Number= ',courant
```

```

if(courant .gt. 0.5D0) print *, '# WARNING: courant > 0.5'
open(unit=11,file='d.dat') ! data file
open(unit=12,file='e.dat') ! data file
! ----- Initial condition at t=0 -----
do i= 1, Nx
  x      = (i-1)*dx
  u(i)   = 0.0D0
enddo
u(Nx/2) = 1.0D0
do i= 1,Nx
  x      = (i-1)*dx
  write(11,*) 0.0D0, x, u(i)
enddo
write(11,*) ' '
! -----
! ----- Calculate time evolution:
do j=2,Nt
  t = (j-1)*dt
  ! ----- second derivative:
  do i=1,Nx
    nnr = i+1
    if(nnr .gt. Nx) nnr = 1
    nnl = i-1
    if(nnl .lt. 1 ) nnl = Nx
    d2udx2(i) = courant*(u(nnr)-2.0D0*u(i)+u(nnl))
  enddo
  ! ----- update:
  prob = 0.0D0
  r2    = 0.0D0
  x0    = ((Nx/2)-1)*dx !original position
  do i=1,Nx
    x      = (i-1)*dx
    u(i)   = u(i) + d2udx2(i)
    prob   = prob + u(i)
    r2     = r2 + u(i)*(x-x0)*(x-x0)
  enddo
  do i=1,Nx
    x = (i-1)*dx
    write(11,*) t, x, u(i)
  enddo
  write(11,*) ' '
  write(12,*) 'pu ',t, prob,r2,u(Nx/2),u(Nx/4),u(1)
enddo ! do j=2,Nt

close(11)
end program diffusion_1d

```

## 8.7 Ανάλυση

Το πρόγραμμα αποθηκεύει στο αρχείο `e.dat` για κάθε χρονική στιγμή τις ποσότητες

$$U_j = \sum_{i=1}^{N_x} u_{i,j} \quad (8.33)$$

που είναι ο διακριτός εκτιμητής της (8.29) και περιμένουμε να παίρνουμε  $U_j = 1$  για κάθε τιμή του  $j$ ,

$$\langle r^2 \rangle_j = \sum_{i=1}^{N_x} u_{i,j} (x_i - x_{N_x/2})^2 \quad (8.34)$$

που είναι ο διακριτός εκτιμητής της (8.9) και περιμένουμε για μικρούς χρόνους να ισχύει

$$\langle r^2 \rangle_j \sim 2t_j, \quad (8.35)$$

καθώς και τις τιμές  $u_{N_x/2,j}$ ,  $u_{N_x/4,j}$ ,  $u_{1,j}$ .

Οι τιμές  $t_j$ ,  $U_j$ ,  $\langle r^2 \rangle_j$ ,  $u_{N_x/2,j}$ ,  $u_{N_x/4,j}$ ,  $u_{1,j}$  βρίσκονται αντίστοιχα στις στήλες 2, 3, 4, 5, 6 και 7 του αρχείου `e.dat`. Ξεκινάμε το `gnuplot` και μέσα από αυτό δίνουμε τις εντολές

```
gnuplot> ! gfortran diffusionS1.f90 -o d
gnuplot> ! echo "10 100 0.4" | ./d
```

που ορίζουν τις τιμές  $N_x = 10$ ,  $N_t = 100$ ,  $t_f = 0.4$ ,  $\Delta x \approx 0.111$ ,  $\Delta t \approx 4.0404$ ,  $\Delta t / \Delta x^2 \approx 0.327$ . Με τις εντολές

```
gnuplot> plot "e.dat" u 2:5 w l
gnuplot> replot "e.dat" u 2:6 w l
gnuplot> replot "e.dat" u 2:7 w l
```

φτιάχνουμε το σχήμα 8.4 από όπου βλέπουμε την ομοιόμορφη κατανομή της διάχυσης για αρκετά μεγάλους χρόνους.

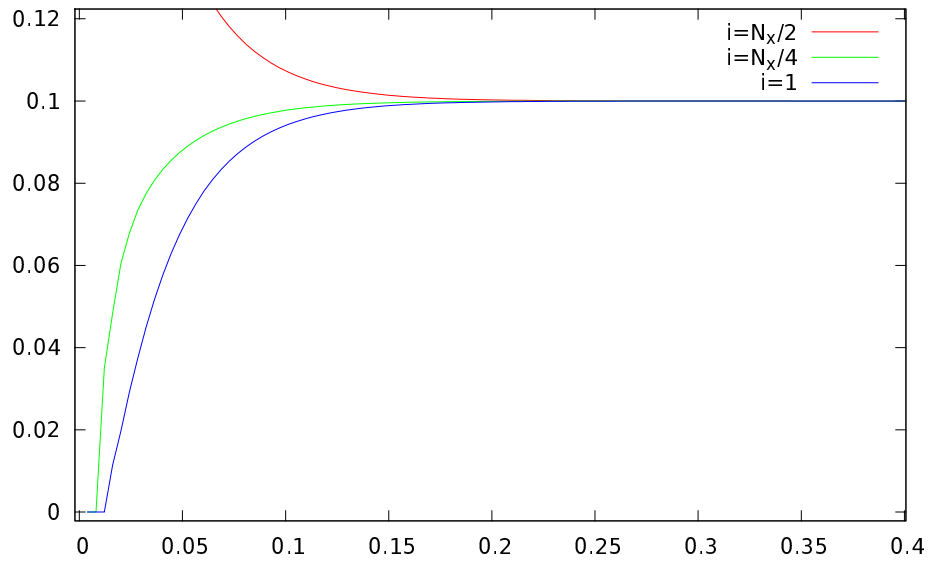
Η σχέση  $U_j = 1$  επιβεβαιώνεται με απλό κοίταγμα στο αρχείο `e.dat`.

Η ασυμπτωτική σχέση  $\langle r^2 \rangle_j \sim 2t_j$  επιβεβαιώνεται με τις εντολές

```
gnuplot> plot [:][:0.11] "e.dat" u 2:4,2*x
```

που μας δίνει το σχήμα 8.5.

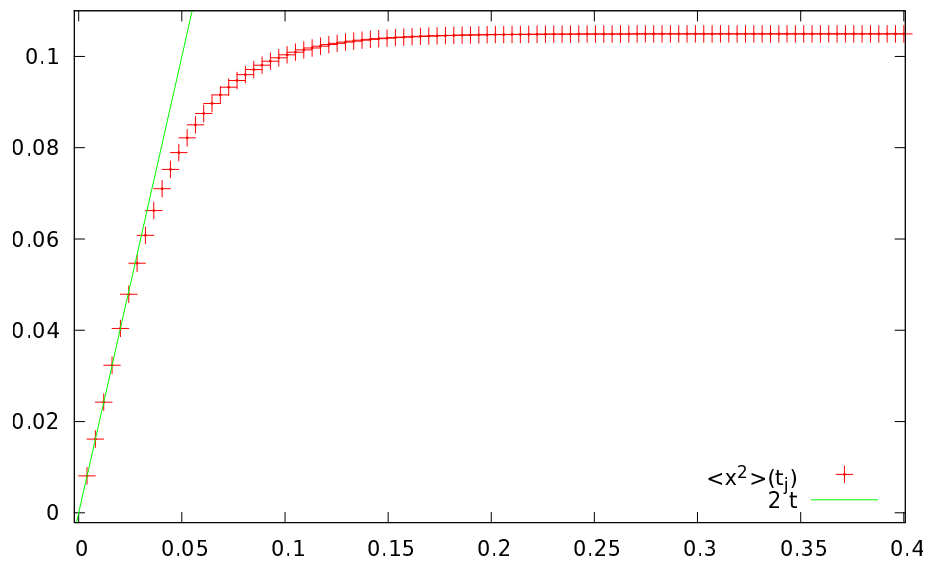
Τέλος, κάνουμε μια επισκόπηση της συνάρτησης  $u(x, t)$  με τις εντολές



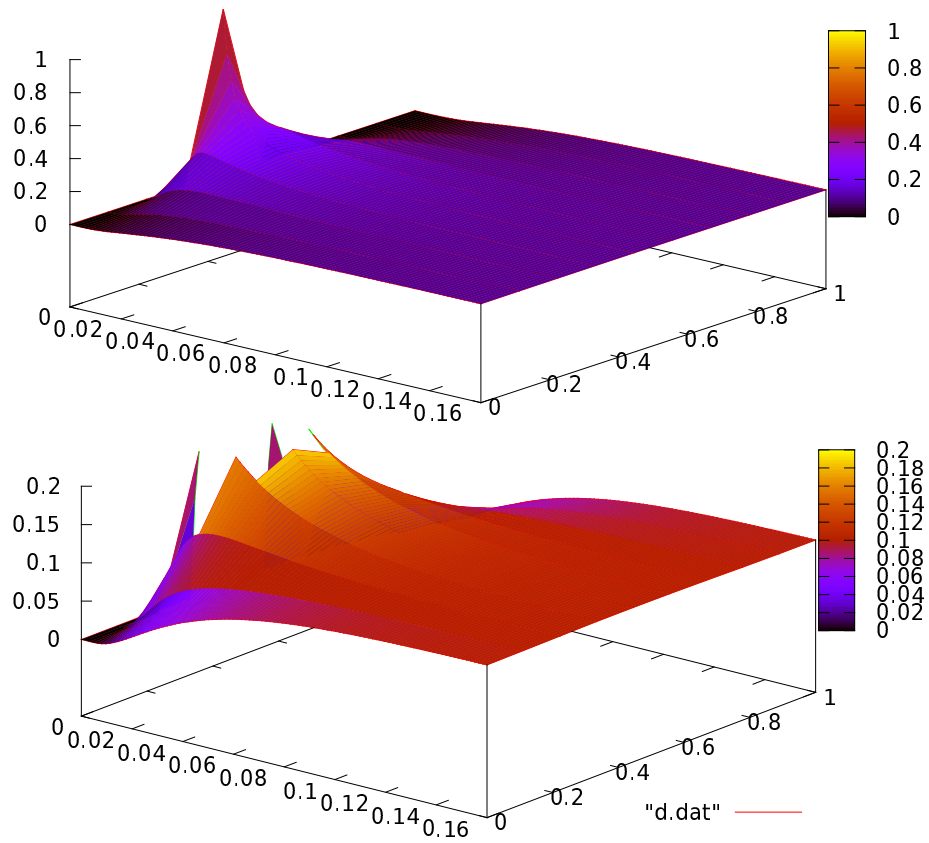
Σχήμα 8.4: Οι συναρτήσεις  $u_{N_x/2,j}$ ,  $u_{N_x/4,j}$ ,  $u_{1,j}$  ως συνάρτηση του  $t_j$  για  $N_x = 10$ ,  $N_t = 100$ ,  $t_f = 0.4$ . Για μεγάλο χρόνο τείνουν προς μια σταθερή τιμή που αντιστοιχεί στην ομοιόμορφη διάχυση.

```
gnuplot> ! echo "10 100 0.16" | ./d
gnuplot> set pm3d
gnuplot> splot [0:0.16][0:1][0: 1] "d.dat" w l
gnuplot> splot [0:0.16][0:1][0:.2] "d.dat" w l
```

και το αποτέλεσμα φαίνεται στο σχήμα 8.6.



Σχήμα 8.5: Η μέση τιμή  $\langle r^2 \rangle_j$  ως συνάρτηση του  $t_j$  για  $N_x = 10$ ,  $N_t = 100$ ,  $t_f = 0.4$ . Για μικρές τιμές του  $t_j$  ισχύει  $\langle r^2 \rangle_j \approx 2t_j$  το οποίο συγκρίνεται με την ευθεία  $2t$ .



Σχήμα 8.6: Η συνάρτηση  $u(x, t)$  για  $N_x = 10$ ,  $N_t = 100$ ,  $t_f = 0.16$ . Στο δεύτερο σχήμα αλλάζουμε μόνο την κλίμακα του άξονα  $z$  ώστε να φανούν οι λεπτομέρειες της διάχυσης μακριά από το σημείο  $x_0 \equiv x_{N_x/2} = x_5$ .



## 8.8 Ασκήσεις

8.1 Να αναπαράγετε τα αποτελέσματα του σχήματος 8.3.

8.2 Η κατανομή της θερμοκρασίας  $u(x, t)$  σε μία λεπτή ράβδο ικανοποιεί την εξίσωση (8.14) μαζί με τις συνοριακές συνθήκες (8.13) στα άκρα της ράβδου  $x = 0, 1$ . Η κατανομή της θερμοκρασίας, όταν  $t = 0$ , δίνεται από τη συνάρτηση

$$u(x, 0) = \begin{cases} 0.5 & x \in [x_1, x_2] \\ 0.3 & x \notin [x_1, x_2] \end{cases},$$

όπου  $x_1 = 0.25$  και  $x_2 = 0.75$ .

(α') Υπολογίστε την κατανομή της θερμοκρασίας  $u(x, t_f)$  όταν  $t_f = 0.0001, 0.001, 0.01, 0.05$ . Να πάρετε  $N_x = 100$  και  $N_t = 1000$ . Να κάνετε το ίδιο για  $t_f = 0.1$  παίρνοντας κατάλληλο  $N_x$  και κρατώντας  $N_t = 1000$ . Να κάνετε τις γραφικές παραστάσεις των  $u(x, t_f)$  στο ίδιο διάγραμμα.

(β') Υπολογίστε γραφικά τη μέγιστη τιμή της θερμοκρασίας για  $t_f = 0.0001, 0.001, 0.01, 0.05, 0.1, 0.15, 0.25$ . Να πάρετε  $N_x = 100$  και να επιλέξετε κατάλληλη τιμή για τα  $N_t$ .

(γ') Υπολογίστε τη χρονική στιγμή κατά την οποία η θερμοκρασία στη ράβδο είναι παντού μικρότερη από 0.1.

Υπόδειξη: Να κάνετε το πρόγραμμά σας να τυπώνει μόνο την τελική κατανομή της θερμοκρασίας  $u(x, t_f)$ .

8.3 Η κατανομή της θερμοκρασίας  $u(x, t)$  σε μία λεπτή ράβδο ικανοποιεί την εξίσωση

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}.$$

Η θερμοκρασία στα άκρα της ράβδου  $u(0, t) = u(1, t) = 0$ , ενώ όταν  $t = 0$

$$u(x, 0) = \begin{cases} 0.5 \left[ 1 - \cos \left( \frac{2\pi x}{b} \right) \right] & 0 \leq x < b \\ 0 & b \leq x \leq 1 \end{cases}.$$

(α') Να υπολογίσετε την κατανομή θερμοκρασίας  $u(x, t_f)$  όταν  $\alpha = 0.5$ ,  $b = 0.09$  και για  $t_f = 0.0001, 0.001, 0.01$ , παίρνοντας  $N_x = 300$ ,  $N_t = 1000$ . Να κάνετε το ίδιο για  $t_f = 0.05$  επιλέγοντας κατάλληλο  $N_x$ . Να κάνετε τις γραφικές παραστάσεις της  $u(x, t_f)$  στο ίδιο διάγραμμα.

(β') Για τις ίδιες παραμέτρους, να υπολογίσετε τη χρονική εξέλιξη της θερμοκρασίας στα σημεία  $x_1 = 0.05$ ,  $x_2 = 0.50$  και  $x_3 = 0.95$  για  $0 \leq t \leq 0.05$ . Να κάνετε τις γραφικές παραστάσεις της  $u(x_{1,2,3}, t)$  στο ίδιο διάγραμμα.

(γ') Να υπολογίσετε την κατανομή θερμοκρασίας  $u(x, t_f)$  για  $b = 0.09$  για τρεις τιμές του  $\alpha = 5, 2, 1$  για  $t_f = 0.001$ . Να κάνετε τις γραφικές παραστάσεις της  $u(x, t_f)$  στο ίδιο διάγραμμα. Να σχολιάσετε την επίδραση της παραμέτρου  $\alpha$  στα αποτελέσματά σας.

8.4 Η κατανομή της θερμοκρασίας  $u(x, t)$  σε μία λεπτή ράβδο μήκους  $L$  ικανοποιεί την εξίσωση

$$\frac{\partial u}{\partial t} = D(x) \frac{\partial^2 u}{\partial x^2} - \frac{4}{L} D(x) \frac{\partial u}{\partial x},$$

όπου  $D(x) = ae^{-4x/L}$  είναι η μεταβλητή παράμετρος θερμοδιαχύσης (thermal diffusivity) της οποίας η τιμή εξαρτάται από τη θέση  $x$ . Η θερμοκρασία της ράβδου στα άκρα της είναι τέτοια, ώστε  $u(0, t) = u(L, t) = 0$ , ενώ τη χρονική στιγμή  $t = 0$ , η κατανομή της θερμοκρασίας κατά μήκος της ράβδου είναι

$$u(x, 0) = Ce^{-(x-L/2)^2/\sigma^2}.$$

(α') Να γράψετε πρόγραμμα που στην είσοδο θα ζητάει από το χρήστη τις παραμέτρους  $L$ ,  $a$ ,  $C$ ,  $\sigma$ ,  $N_x$ ,  $N_t$  και  $t_f$ . Στην έξοδο θα υπολογίζει την  $u(x, t_f)$  και θα τυπώνει σε ένα αρχείο d.dat τα σημεία  $(x_i, u(x_i, t_f))$  σε δύο στήλες.

(β') Να εκτελέσετε το πρόγραμμα για  $L = 4$ ,  $a = 0.2$ ,  $C = 1$ ,  $\sigma = 1/2$ ,  $N_x = 400$ ,  $N_t = 20000$  και να υπολογίσετε την  $u(x, t_f)$  για  $t_f = 0.05, 1.0, 5.0$ . Να κάνετε τις γραφικές παραστάσεις της  $u(x, t_f)$  στο ίδιο διάγραμμα.

(γ') Για τις ίδιες παραμέτρους, να υπολογίσετε τη χρονική εξέλιξη της θερμοκρασίας στα σημεία  $x_1 = 1$  και  $x_2 = 2$  για  $0 \leq t \leq 5$ . Να κάνετε τις γραφικές παραστάσεις της  $u(x_{1,2}, t)$  στο ίδιο διάγραμμα.

8.5 Να αναπαράγετε τα αποτελέσματα που δείχνονται στα σχήματα 8.4 και 8.5.

## ΚΕΦΑΛΑΙΟ 9

# Ο Αναρμονικός Ταλαντωτής

Στο κεφάλαιο αυτό θα εφαρμόσουμε μεθόδους πινάκων για τη λύση του κβαντομηχανικού προβλήματος του προσδιορισμού των ενεργειακών επιπέδων του αναρμονικού ταλαντωτή. Το πρόβλημα αυτό δεν μπορεί να λυθεί αναλυτικά, οπότε πρέπει να προσφύγουμε σε διαταρακτικές ή άλλες προσεγγιστικές μεθόδους. Εμείς θα αντιμετωπίσουμε το πρόβλημα αριθμητικά. Για το σκοπό αυτό θα επιλέξουμε κατάλληλη βάση για να αναπαραστήσουμε τη Χαμιλτονιανή  $H$  υπό μορφή πίνακα τον οποίο θα διαγωνιοποιήσουμε με αριθμητικές μεθόδους με σκοπό να πάρουμε τις ενεργειακές ιδιοτιμές του τελεστή. Φυσικά, ο πίνακας αυτός είναι απείρου μεγέθους και, αρχικά, η βάση αναπαράστασης που θα επιλέξουμε θα καλύπτει μόνο έναν υπόχωρο  $\mathcal{H}_N$  του χώρου Hilbert  $\mathcal{H}$  των καταστάσεων πεπερασμένης διάστασης  $N$ , έτσι ώστε να πάρουμε πίνακα πεπερασμένου μεγέθους  $N \times N$ . Οι ιδιοτιμές του  $N \times N$  πίνακα θα υπολογιστούν αριθμητικά και οι ενεργειακές ιδιοτιμές θα καθοριστούν από το όριο που θα τείνουν αυτές, όταν  $N \rightarrow \infty$ .

Για τον υπολογισμό των ιδιοτιμών θα χρησιμοποιήσουμε τις υπορουτίνες που βρίσκουμε στην βιβλιοθήκη LAPACK. Το κεφάλαιο αυτό, εκτός των άλλων, θα είναι και μια άσκηση για το πώς να συνδέουμε το πρόγραμμά μας με προγράμματα που βρίσκονται σε βιβλιοθήκες λογισμικού. Για να λύσετε το ίδιο πρόβλημα με Mathematica ή Matlab δείτε τις αναφορές [39] και [40] αντίστοιχα.

## 9.1 Εισαγωγή

Η Χαμιλτονιανή του αρμονικού ταλαντωτή δίνεται από τη σχέση:

$$H_0 = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2 \quad (9.1)$$

και ορίζοντας  $x_0 = \sqrt{\hbar/(m\omega)}$ ,  $p_0 = \sqrt{\hbar m\omega}$  παίρνουμε την εξίσωση αδιάστατων μεγεθών:

$$\frac{H_0}{\hbar\omega} = \frac{1}{2}\left(\frac{p}{p_0}\right)^2 + \frac{1}{2}\left(\frac{x}{x_0}\right)^2. \quad (9.2)$$

Μετρώντας την ενέργεια σε μονάδες  $\hbar\omega$ , τις αποστάσεις σε μονάδες  $x_0$  και τις ορμές σε μονάδες  $p_0$  παίρνουμε

$$H_0 = \frac{1}{2}p^2 + \frac{1}{2}x^2 \quad (9.3)$$

Ο τελεστής  $H_0$  μπορεί να διαγωνιοποιηθεί εύκολα με τη βοήθεια των τελεστών δημιουργίας/καταστροφής:

$$x = \frac{1}{\sqrt{2}}(a^\dagger + a) \quad p = \frac{i}{\sqrt{2}}(a^\dagger - a) \quad (9.4)$$

ή

$$a = \frac{1}{\sqrt{2}}(x + ip) \quad a^\dagger = \frac{1}{\sqrt{2}}(x - ip) \quad (9.5)$$

που ικανοποιούν τη σχέση μετάθεσης

$$[a, a^\dagger] = 1 \quad (9.6)$$

και τότε

$$H_0 = a^\dagger a + \frac{1}{2}. \quad (9.7)$$

Οι ιδιοκαταστάσεις  $|n\rangle$ ,  $n = 0, 1, 2, \dots$  της  $H_0$  καλύπτουν τον χώρο Hilbert  $\mathcal{H}$  και ικανοποιούν τις σχέσεις

$$a^\dagger |n\rangle = \sqrt{n+1} |n+1\rangle \quad a |n\rangle = \sqrt{n} |n-1\rangle \quad a |0\rangle = 0 \quad (9.8)$$

οπότε

$$a^\dagger a |n\rangle = n |n\rangle \quad (9.9)$$

και

$$H_0 |n\rangle = E_n |n\rangle, \quad E_n = n + \frac{1}{2}. \quad (9.10)$$

Η αναπαράσταση θέσης των ιδιοκαταστάσεων  $|n\rangle$  είναι:

$$\psi_n(x) = \langle x|n\rangle = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-x^2/2} H_n(x) \quad (9.11)$$

όπου  $H_n(x)$  τα πολυώνυμα Hermite.

Από τις σχέσεις (9.4), (9.8) προκύπτει ότι

$$x_{nm} = \langle n | x | m \rangle = \frac{1}{\sqrt{2}} \sqrt{m+1} \delta_{n,m+1} + \frac{1}{\sqrt{2}} \sqrt{m} \delta_{n,m-1} \quad (9.12)$$

$$= \frac{1}{2} \sqrt{n+m+1} \delta_{|n-m|,1} \quad (9.13)$$

$$p_{nm} = \langle n | p | m \rangle = \frac{i}{\sqrt{2}} \sqrt{m+1} \delta_{n,m+1} - \frac{i}{\sqrt{2}} \sqrt{m} \delta_{n,m-1} \quad (9.14)$$

Από την παραπάνω σχέση μπορούμε εύκολα να υπολογίσουμε την Χαμιλτονιανή του αναρμονικού ταλαντωτή

$$H(\lambda) = H_0 + \lambda x^4 \quad (9.15)$$

και τα στοιχεία του πίνακα αναπαράστασης στον  $\mathcal{H}$ :

$$H_{nm}(\lambda) \equiv \langle n | H(\lambda) | m \rangle = \langle n | H_0 | m \rangle + \lambda \langle n | x^4 | m \rangle \quad (9.16)$$

$$= (n + \frac{1}{2}) \delta_{n,m} + \lambda (x^4)_{nm} \quad (9.17)$$

όπου το  $(x^4)_{nm}$  μπορούμε να υπολογίσουμε από τη σχέση (9.12):

$$(x^4)_{nm} = \sum_{i_1, i_2, i_3=0}^{\infty} x_{ni_1} x_{i_1 i_2} x_{i_2 i_3} x_{i_3 m} . \quad (9.18)$$

Το πρόβλημα εύρεσης του ενεργειακού φάσματος ανάγεται στον υπολογισμό των ιδιοτιμών του πίνακα  $H_{nm}$ .

## 9.2 Υπολογισμός Ιδιοτιμών του $H_{nm}(\lambda)$

Αρχικά επιλέγουμε τη διάσταση  $N$  του υπόχωρου  $\mathcal{H}_N$  του χώρου Hilbert  $\mathcal{H}$  των καταστάσεων στον οποίο θα περιοριστούμε. Χρησιμοποιούμε τις σχέσεις που αναφέραμε στην προηγούμενη παράγραφο για να υπολογίσουμε τους πίνακες αναπαράστασης των τελεστών  $x$ ,  $H_0$ ,  $H(\lambda)$  μέσα σε αυτόν τον υπόχωρο. Για παράδειγμα, όταν  $N = 4$  έχουμε:

$$x = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 1 & 0 \\ 0 & 1 & 0 & \sqrt{\frac{3}{2}} \\ 0 & 0 & \sqrt{\frac{3}{2}} & 0 \end{pmatrix} \quad (9.19)$$

$$H_0 = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{3}{2} & 0 & 0 \\ 0 & 0 & \frac{5}{2} & 0 \\ 0 & 0 & 0 & \frac{7}{2} \end{pmatrix} \quad (9.20)$$

$$H(\lambda) = \begin{pmatrix} \frac{1}{2} + \frac{3\lambda}{4} & 0 & \frac{3\lambda}{\sqrt{2}} & 0 \\ 0 & \frac{3}{2} + \frac{15\lambda}{4} & 0 & 3\sqrt{\frac{3}{2}}\lambda \\ \frac{3\lambda}{\sqrt{2}} & 0 & \frac{5}{2} + \frac{27\lambda}{4} & 0 \\ 0 & 3\sqrt{\frac{3}{2}}\lambda & 0 & \frac{7}{2} + \frac{15\lambda}{4} \end{pmatrix} \quad (9.21)$$

Σκοπός μας είναι να γράψουμε ένα πρόγραμμα που θα υπολογίζει τις ιδιοτιμές  $E_n(N, \lambda)$  του  $N \times N$  πίνακα  $H_{nm}(\lambda)$ . Αντί, όμως, να γράψουμε το δικό μας κώδικα για τον υπολογισμό των ιδιοτιμών και ιδιοδιανυσμάτων των πινάκων που μας ενδιαφέρουν θα χρησιμοποιήσουμε τις έτοιμες ρουτίνες που υπάρχουν στη βιβλιοθήκη LAPACK. Η βιβλιοθήκη αυτή υπάρχει στον δικτυακό τόπο <http://www.netlib.org/lapack/> και λεπτομέρειες για τη χρήση της μπορούν να βρεθούν στο <http://www.netlib.org/lapack/lug/>. Επισκεφτείτε το δικτυακό τόπο και αναζητήστε ρουτίνες που σας ενδιαφέρουν<sup>1</sup>.

Ως απλοί, άπειροι χρήστες, θα αναζητήσουμε “ρουτίνες οδηγούς” (driver routines) που κάνουν μια εργασία διαγωνιοποίησης. Έχουμε να διαγωνιοποιήσουμε έναν πίνακα συμμετρικό και διαλέγουμε τη ρουτίνα DSYEV (D = double precision, SY = symmetric, EV = eigenvalues with optional eigenvectors). Οι συναρτήσεις της LAPACK έχουν βοήθεια online από τα man pages του συστήματος (Unix/Linux). Η εντολή που δίνουμε είναι η

```
> man dsyev
```

Από εκεί μαθαίνουμε ότι η χρήση της είναι:

```
SUBROUTINE DSYEV( JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, INFO )
  CHARACTER      JOBZ, UPLO
  INTEGER         INFO, LDA, LWORK, N
  DOUBLE          PRECISION A( LDA, * ), W( * ), WORK( * )

ARGUMENTS
```

<sup>1</sup>Η βιβλιοθήκη μπορεί να εγκατασταθεί εύκολα σε διανομές Linux. Α.χ. στη διανομή Ubuntu εκτελέστε την εντολή `apt-get install liblapack3 liblapack-doc liblapack-dev`.

**JOBZ** (input) CHARACTER\*1  
 = 'N': Compute eigenvalues only;  
 = 'V': Compute eigenvalues and eigenvectors.

**UPLO** (input) CHARACTER\*1  
 = 'U': Upper triangle of A is stored;  
 = 'L': Lower triangle of A is stored.

**N** (input) INTEGER  
 The order of the matrix A.  $N \geq 0$ .

**A** (input/output) DOUBLE PRECISION array, dimension (LDA, N)  
 On entry, the symmetric matrix A. If UPLO = 'U', the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A. On exit, if JOBZ = 'V', then if INFO = 0, A contains the orthonormal eigenvectors of the matrix A. If JOBZ = 'N', then on exit the lower triangle (if UPLO='L') or the upper triangle (if UPLO='U') of A, including the diagonal, is destroyed.

**LDA** (input) INTEGER  
 The leading dimension of the array A.  $LDA \geq \max(1, N)$ .

**W** (output) DOUBLE PRECISION array, dimension (N)  
 If INFO = 0, the eigenvalues in ascending order.

**WORK** (workspace/output) DOUBLE PRECISION array, dimension (LWORK).  
 On exit, if INFO = 0, WORK(1) returns the optimal LWORK.

**LWORK** (input) INTEGER  
 The length of the array WORK.  $LWORK \geq \max(1, 3*N - 1)$ .  
 For optimal efficiency,  $LWORK \geq (NB+2)*N$ , where NB is the blocksize for DSYTRD returned by ILAENV.  
 If LWORK = -1, then a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued by XERBLA.

**INFO** (output) INTEGER  
 = 0: successful exit  
 < 0: if INFO = -i, the i-th argument had an illegal value

```

value
> 0: if INFO = i, the algorithm failed to converge; ←
      i
      off-diagonal elements of an intermediate tridiagonal
      form did not converge to zero.

```

Σελίδες όπως τις παραπάνω, τις διαβάζουμε αναζητώντας τα δεδομένα που πρέπει να δώσουμε στην είσοδο στην υπορουτίνα που μας ενδιαφέρει και τα δεδομένα που θα πάρουμε στην έξοδο με την πληροφορία που θέλουμε να χρησιμοποιήσουμε. Μαθαίνουμε πως ο προς διαγωνιοποίηση πίνακας  $A$  έχει αριθμό γραμμών και στηλών που είναι  $\leq N$ . Ο αριθμός των γραμμών LDA (LDA= “leading dimension of A”) μπορεί να είναι μεγαλύτερος από  $N$  και η DSYEV θα διαγωνιοποιήσει το πάνω αριστερά  $N \times N$  κομμάτι του πίνακα<sup>2</sup>. Αυτό θα το χρησιμοποιήσουμε προς όφελός μας, έτσι ώστε να ορίσουμε ένα πίνακα μεγάλου μεγέθους  $A(LDA, LDA)$  διαγωνιοποιώντας κάθε φορά μεταβλητού μεγέθους τετράγωνο κομμάτι του  $A(N, N)$ . Η υπορουτίνα μπορεί να χρησιμοποιηθεί με δύο τρόπους:

- Αν  $JOBZ='N'$ , υπολογίζει μόνο τις ιδιοτιμές του πίνακα  $A(N, N)$  και τις αποθηκεύει στο array  $W(N)$  με σειρά αυξανόμενου μεγέθους. Μόλις η ρουτίνα επιστρέψει, το άνω ή κάτω τριγωνικό κομμάτι του  $A$  έχει καταστραφεί ανάλογα με το αν  $UPLO='U'$  ή  $'L'$  αντίστοιχα. Αυτό είναι και το κομμάτι του  $A$  που είμαστε υποχρεωμένοι να παρέχουμε στην DSYEV. Αν θέλουμε να χρησιμοποιήσουμε ξανά τον πίνακα  $A(N, N)$  μετά την κλήση της DSYEV θα πρέπει να ανακατασκευάσουμε τον πίνακα χρησιμοποιώντας αποθηκευμένο αντίγραφο του.
- Αν  $JOBZ='V'$ , υπολογίζει ιδιοτιμές και ιδιοδιανύσματα του  $A(N, N)$ . Οι ιδιοτιμές αποθηκεύονται όπως και πριν στο array  $W(N)$ , ενώ τα ιδιοδιανύσματα στις στήλες του πίνακα  $A(N, N)$ . Δηλαδή αν θέλουμε να πάρουμε τα ιδιοδιανύσματα, ένας τρόπος είναι με τη σχέση  $v = A(1:N, j)$  όπου το array  $v(N)$  θα έχει τις συνιστώσες του  $j$ -οστού ιδιοδιανύσματος του πίνακα που αντιστοιχεί στην ιδιοτιμή  $\lambda_j$ . Το διάνυσμα αυτό είναι κανονικοποιημένο στη μονάδα, δηλ.  $\sum_{i=1}^N v(i) * v(i) = 1$ . Ο πίνακας  $A(N, N)$  καταστρέφεται και αν τον χρειάζομαστε πάλι θα πρέπει να τον έχουμε αποθηκεύσει.

Κάτι άλλο που μπορεί να προκαλέσει απορίες είναι η χρήση του array WORK. Αυτός είναι βοηθητικός χώρος στη μνήμη που δίνεται στην DSYEV

<sup>2</sup>Ο αριθμός LDA είναι αναγκαίος, γιατί το στοιχείο  $A(i, j)$  απέχει  $i + (LDA - 1) * j$  θέσεις στη μνήμη από το  $A(1, 1)$ .



για να μπορέσει να κάνει τους ενδιάμεσους υπολογισμούς. Αυτό είναι ένα array μεγέθους LWORK και η επιλογή του LWORK δεν είναι μοναδική. Εμείς θα επιλέξουμε την απλή περίπτωση  $LWORK=3*LDA-1$ . Η μεταβλητή INFO είναι η “σημαία” για τον αν εξελίχθηκε ο υπολογισμός ομαλά και στο πρόγραμμα ελέγχουμε αν η τιμή της είναι 0 και αν όχι, σταματάμε το πρόγραμμα.

Για την κατανόηση της χρήσης ενός προγράμματος μιας βιβλιοθήκης είναι πάντα απαραίτητο να γράφουμε ένα δοκιμαστικό πρόγραμμα που να το χρησιμοποιεί και του οποίου τα αποτελέσματα μπορούμε εύκολα να ελέγξουμε. Οδηγούμαστε λοιπόν στο να γράψουμε τον εξής κώδικα για να δοκιμάσουμε τη χρήση της DSYEV σε ένα πίνακα  $A(N,N)$ :

```

program test_evs
  implicit none
  integer , parameter :: P      = 100 ! P= LDA
  integer , parameter :: LWORK = 3*P-1
  real(8) :: A(P,P),W(P),WORK(LWORK)
  integer :: N ! DSYEV diagonalizes A(N,N)
  integer :: i,j
  integer :: LDA,INFO
  character(1) :: JOBZ,UPLD
  !Define the **symmetric** matrix to be diagonalized
  !The subroutine uses the upper triangular part (UPLD='U')
  !therefore the lower triangular part needs not to be defined
  N=4
  A(1,1)=-7.7;
  A(1,2)= 2.1;A(2,2)= 8.3;
  A(1,3)=-3.7;A(2,3)=-16.;A(3,3)=-12.
  A(1,4)= 4.4;A(2,4)= 4.6;A(3,4)=-1.04;A(4,4)=-3.7
  !We print the matrix A before calling DSYEV since it is
  !destroyed after the call.
  do i=1,N
    do j=i,N
      print *, 'A( ',i,' , ',j,' )=',A(i,j)
    enddo
  enddo
  !We ask for eigenvalues AND eigenvectors (JOBZ='V')
  JOBZ='V'; UPLD='U'
  print *, 'COMPUTING WITH DSYEV: '
  LDA=P !notice that LDA-> P>N !!
  call DSYEV(JOBZ,UPLD,N,A,LDA,W,WORK,LWORK,INFO)
  print *, 'DSYEV: DONE. CHECKING NOW: '
  !If INFO is nonzero, then there is an error:
  if(INFO .ne. 0)then
    print *, 'DSYEV FAILED. INFO= ',INFO
    stop
  end if
end program

```

```

endif
!Print results: W(I) has the eigenvalues:
print *, 'DSYEV: DONE.: '
print *, 'EIGENVALUES OF MATRIX: '
do i=1,N
  print *, 'LAMBDA( ',i, ')= ',W(i)
enddo
!Eigenvectors are in stored in the columns of A:
print *, 'EIGENVECTORS OF MATRIX'
do J=1,N
  print *, 'EIGENVECTOR ',j, ' FOR EIGENVALUE ',W(j)
  do i=1,N
    print *, 'V_',j, '( ',i, ')= ',A(i,j)
  enddo
enddo
end program test_evs

```

Το επόμενο βήμα είναι να μεταγλωττίσουμε τον κώδικα. Το σημείο που πρέπει να προσέξουμε είναι ότι στο στάδιο της σύνδεσης (linking) πρέπει να δώσουμε οδηγίες στον *linker ld* που βρίσκονται οι βιβλιοθήκες LAPACK και η BLAS (οι βασικές υπολογιστικές ρουτίνες γραμμικής άλγεβρας είναι στην BLAS). Όλες οι συναρτήσεις είναι μεταγλωττισμένες και τα object files τους είναι αρχειοθετημένα στα αρχεία **liblapack.a** και **libblas.a** που μπορούμε να αναζητήσουμε με τις εντολές:

```

> locate libblas
> locate liblapack

```

Για να δούμε τα περιεχόμενά τους δίνουμε τις εντολές<sup>3</sup>:

```

> ar -t /usr/lib/libblas.a
> ar -t /usr/lib/liblapack.a

```

(ή αντικαθιστούμε το /usr/lib με τη διαδρομή που αντιστοιχεί στο σύστημά μας). Αν ο κώδικάς μας είναι στο αρχείο test.f90 για τη μεταγλώττιση δίνουμε την εντολή:

```

> gfortran test.f90 -o test -L/usr/lib -llapack -lblas

```

Η επιλογή -L/usr/lib λέει στον linker να αναζητήσει τις βιβλιοθήκες στο /usr/lib<sup>4</sup>, ενώ οι -llapack -lblas του λένε να αναζητήσει όποια

<sup>3</sup> Αν δεν υπάρχουν αρχεία με κατάληξη .a, δοκιμάστε ar -t /usr/lib/libblas.so κλπ.

<sup>4</sup> Άχρηστο στην περίπτωσή μας, γιατί το ψάχνει έτσι και αλλιώς, χρήσιμο, αν έχουμε

σύμβολα<sup>5</sup> δεν έχουν ξεκαθαριστεί κατά τη μεταγλώττιση του αρχείου `test.f90` πρώτα στη βιβλιοθήκη `liblapack.a` και μετά στην `libblas.a`

Η παραπάνω εντολή έχει ως αποτέλεσμα το εκτελέσιμο αρχείο `test` που, όταν το τρέξουμε, παίρνουμε το αποτέλεσμα:

```
EIGENVALUES OF MATRIX:
LAMBDA( 1)= -21.4119907
LAMBDA( 2)= -9.93394359
LAMBDA( 3)= -2.55765591
LAMBDA( 4)= 18.8035905
EIGENVECTORS OF MATRIX
EIGENVECTOR 1 FOR EIGENVALUE -21.4119907
V_ 1( 1)= -0.197845668
V_ 1( 2)= -0.464798676
V_ 1( 3)= -0.854691009
V_ 1( 4)= 0.119676904
EIGENVECTOR 2 FOR EIGENVALUE -9.93394359
V_ 2( 1)= 0.824412399
V_ 2( 2)= -0.132429396
V_ 2( 3)= -0.191076519
V_ 2( 4)= -0.516039161
EIGENVECTOR 3 FOR EIGENVALUE -2.55765591
V_ 3( 1)= 0.502684215
V_ 3( 2)= -0.247784372
V_ 3( 3)= 0.132853329
V_ 3( 4)= 0.817472616
EIGENVECTOR 4 FOR EIGENVALUE 18.8035905
V_ 4( 1)= 0.168848655
V_ 4( 2)= 0.839659187
V_ 4( 3)= -0.464050682
V_ 4( 4)= 0.226096318
```

Τώρα είμαστε έτοιμοι να λύσουμε το πρόβλημα του αναρμονικού ταλαντωτή. Το πρόγραμμα βρίσκεται στο συνοδευτικό λογισμικό στο αρχείο `anharmonic.f90`. Στην κύρια ρουτίνα του προγράμματος ο χρήστης εισάγει τις βασικές παραμέτρους, τη διάσταση  $\text{DIM} \equiv N$  του  $\mathcal{H}_N$  και την τιμή του  $\lambda$  για την οποία επιθυμεί να υπολογιστούν οι ιδιοτιμές  $E_n(N, \lambda)$  του πίνακα αναπαράστασης  $H_{nm}(\lambda)$  του τελεστή  $H(\lambda)$  στην  $\{|n\rangle\}_{n=0,1,\dots,N-1}$  αναπαράσταση στον  $\mathcal{H}_N$ . Το πρόγραμμα καλεί την υπορουτίνα `calculate_X4` για να υπολογίσει τον  $N \times N$  πίνακα αναπαράστασης  $(x^4)_{nm}$  του τελεστή  $x^4$ . Ο υπολογισμός στην υπορουτίνα αυτή μπορεί να γίνει γρηγορότερα υπολογίζοντας τα  $(x^4)_{nm}$  αναλυτικά. Αυτό αφήνεται ως άσκηση στον αναγνώστη. Στη συνέχεια, υπολογίζονται οι

βιβλιοθήκες σε μη συμβατικά μέρη.

<sup>5</sup>Λ.χ. ονόματα συναρτήσεων και υπορουτινών.

ιδιοτιμές του  $H_{nm}(\lambda)$  καλώντας την υπορουτίνα `calculate_evs` και τα αποτελέσματα τυπώνονται στο `stdout` (standard output).

Η υπορουτίνα `calculate_evs` καλεί την `calculate_H` να υπολογίσει τον πίνακα  $H_{nm}(\lambda)$  η οποία κάνει χρήση των σχέσεων (9.16). Στη συνέχεια, καλείται η `DSYEV` της LAPACK να κάνει τη διαγωνιοποίηση. Προσέχουμε στο όρισμα LDA της `DSYEV` να βάλουμε τη σωστή διάσταση του πίνακα  $H$  που είναι  $P$  και όχι  $DIM$ . Στη συνέχεια, παρατίθεται ο κώδικας:

```
!=====
program anharmonic_elevels
!=====
  implicit none
  integer ,parameter      :: P      = 1000
  integer ,parameter      :: LWORK = 3*P-1
  integer                 :: DIM
  real(8) ,dimension(P,P) :: H,X,X4 !Hamiltonian+Position Ops
  real(8) ,dimension(P)   :: E      !energy eigenvalues
  real(8) ,dimension(LWORK):: WORK
  real(8)                 :: lambda
  integer                 :: i

  print *, '# Enter Hilbert Space dimension: '
  read  *, DIM
  print *, '# Enter lambda: '
  read  *, lambda
  print *, '# lambda= ', lambda
!Print Message:
  print *, '# #####'
  print *, '# Energy spectrum of anharmonic oscillator '
  print *, '# using matrix methods.'
  print *, '# Hilbert Space Dimension DIM = ', DIM
  print *, '# lambda coupling = ', lambda
  print *, '# #####'
  print *, '# Output: DIM lambda E_0 E_1 .... E_{N-1}'
  print *, '# _____'

!Calculate X^4 operator:
  call calculate_X4(X,X4,DIM)
!Calculate eigenvalues:
  call calculate_evs(H,X4,E,WORK,lambda,DIM)
  write(6,100) 'EV ', DIM, lambda, (E(i), i=1,DIM)
100 FORMAT(A3,I8,20000G25.15)
end program anharmonic_elevels
!=====
subroutine calculate_evs(H,X4,E,WORK,lambda,DIM)
!=====
  implicit none
```

```

integer ,parameter      :: P      = 1000
integer ,parameter      :: LWORK = 3*P-1
real(8) ,dimension(P,P) :: H,X4
real(8) ,dimension(P)   :: E
real(8) ,dimension(LWORK) :: WORK
integer                 :: DIM
real(8)                 :: lambda
character(1)            :: JOBZ,UPL0
integer                 :: LDA,INFO,i,j

call calculate_H(H,X4,lambda,DIM)
JOBZ='V';UPL0='U'
call DSYEV(JOBZ,UPL0,DIM,H,P,E,WORK,LWORK,INFO)
print *,'# ***** EVEC ***** '
do j=1,DIM
  write(6,101)'# EVEC ',lambda,(H(i,j), i=1,DIM)
enddo
print *,'# ***** EVEC ***** '
101 FORMAT(A7,F15.3,20000G14.6)
!If INFO is nonzero then we have an error
if(INFO .ne. 0)then
  print *,'dsyev failed. INFO= ',INFO
  stop
endif

end subroutine calculate_evs
!=====
subroutine calculate_H(H,X4,lambda,DIM)
!=====
implicit none
integer ,parameter      :: P = 1000
real(8) ,dimension(P,P) :: H,X4
integer                 :: DIM
real(8)                 :: lambda
integer                 :: i,j

do j=1,DIM
  do i=1,DIM
    H(i,j)=lambda*X4(i,j)
  enddo
  H(j,j) = H(j,j) + DBLE(j) - 0.5D0 !E_n=n+1/2,n=j-1=>E_n=j-1/2
enddo

print *,'# ***** H ***** '
do j=1,DIM
  write(6,102)'# HH ',(H(i,j), i=1,DIM)
enddo
print *,'# ***** H ***** '

```

```

102 FORMAT(A5,20000G20.6)
end subroutine calculate_H
!=====
subroutine calculate_X4(X,X4,DIM)
!=====
  implicit none
  integer ,parameter      :: P=1000
  real(8) ,dimension(P,P) :: X,X4,X2
  integer                  :: DIM
  integer                  :: i,j,m,n
  real(8) ,parameter      :: isqrt2=1.0D0/sqrt(2.0D0)
!Compute the position operator:
  X = 0.0D0
!Compute the nonzero elements
  do i=1,DIM
    n=i-1 !indices 0,...,DIM-1
    ! The delta_{n,m+1} term, i.e. m=n-1
    m=n-1 !the energy level n -> i=n+1, m-> j=m+1
    j=m+1
    if(j.ge.1) X(i,j)=isqrt2*sqrt(DBLE(m+1))
    ! The delta_{n,m-1} term, i.e. m=n+1
    m=n+1
    j=m+1
    if(j.le.DIM) X(i,j)=isqrt2*sqrt(DBLE(m))
  enddo
!Compute the Hamiltonian operator:
!Start with the X^4 operator:
  X2 = MATMUL(X,X) !first X2, then X4:
  X4 = MATMUL(X2,X2)
end subroutine calculate_X4

```

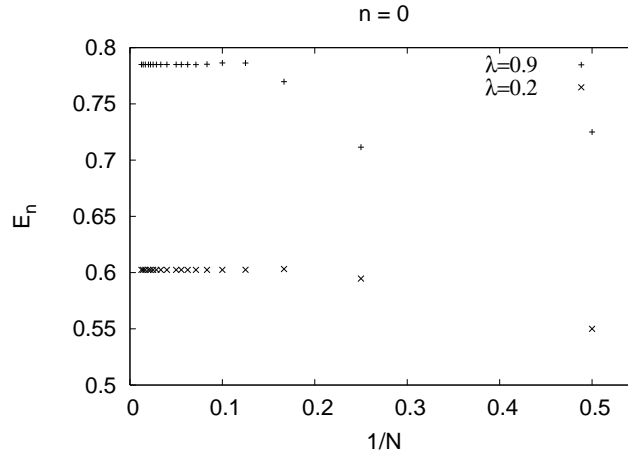
### 9.3 Αποτελέσματα

Για να τρέξουμε το πρόγραμμα, μεταγλωττίζουμε και δίνουμε τα δεδομένα στο πρόγραμμα

```

> gfortran -O2 anharmonic.f90 -o an -llapack -lblas
> ./an
# Enter Hilbert Space dimension:
4
# Enter lambda:
0.0
.....
# ***** H *****
# HH          0.50   0.00   0.00   0.00

```



Σχήμα 9.1: Η ενεργειακή στάθμη  $E_0(\lambda)$  για  $\lambda = 0.2, 0.9$  υπολογίζεται από το όριο των ιδιοτιμών  $E_0(N, \lambda)$ , καθώς  $N \rightarrow \infty$ . Σύγκλιση επιτυγχάνεται για σχετικά μικρές τιμές του  $N$ , ενώ φαίνεται ότι για  $\lambda = 0.2$  γίνεται ελαφρώς γρηγορότερα από ότι για  $\lambda = 0.9$ .

```
# HH      0.00  1.50  0.00  0.00
# HH      0.00  0.00  2.50  0.00
# HH      0.00  0.00  0.00  3.50
# ***** H *****
# ***** EVEC *****
# EVEC 0.000  1.00  0.00  0.00  0.00
# EVEC 0.000  0.00  1.00  0.00  0.00
# EVEC 0.000  0.00  0.00  1.00  0.00
# EVEC 0.000  0.00  0.00  0.00  1.00
# ***** EVEC *****
EV  4  0.000  0.50  1.50  2.50  3.50
```

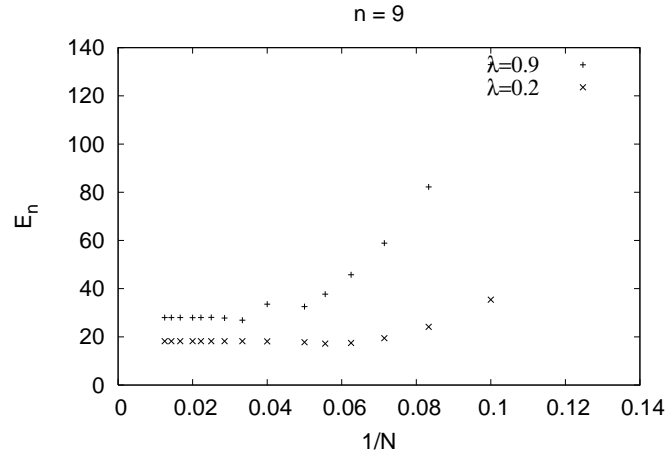
όπου παραπάνω τρέξαμε το πρόγραμμα για  $N = 4$  και  $\lambda = 0$ . Αυτό αντιστοιχεί στον απλό αρμονικό ταλαντωτή και βλέπουμε πως παίρνουμε τις αναμενόμενες λύσεις:  $H_{nm} = (n+1/2)\delta_{n,m}$ ,  $E_n = (n+1/2)$  και οι ιδιοκαταστάσεις (ιδιοδιανύσματα του  $H_{nm}$ )  $|n\rangle_{\lambda=0} = |n\rangle = \sum_{m=0}^3 \delta_{n,m} |m\rangle$ . Ανάλογα αποτελέσματα θα πάρουμε και για μεγαλύτερα  $N$ .

Για μη μηδενικές τιμές του  $\lambda$ , ο υπολογισμός για πεπερασμένο  $N$  ενέχει συστηματικά σφάλματα, επειδή αγνοούνται όλα τα στοιχεία του πίνακα  $H_{nm}(\lambda)$  για  $n \geq N$  ή  $m \geq N$ . Το πρόγραμμά μας υπολογίζει τις ιδιοτιμές  $E_n(N, \lambda)$  του πεπερασμένου πίνακα  $H_{nm}(\lambda)$ ,  $m, n = 0, \dots, N-1$  και αναμένεται ότι

$$E_n(\lambda) = \lim_{N \rightarrow \infty} E_n(N, \lambda), \quad (9.22)$$

όπου

$$H(\lambda) |n\rangle_\lambda = E_n(\lambda) |n\rangle_\lambda, \quad (9.23)$$



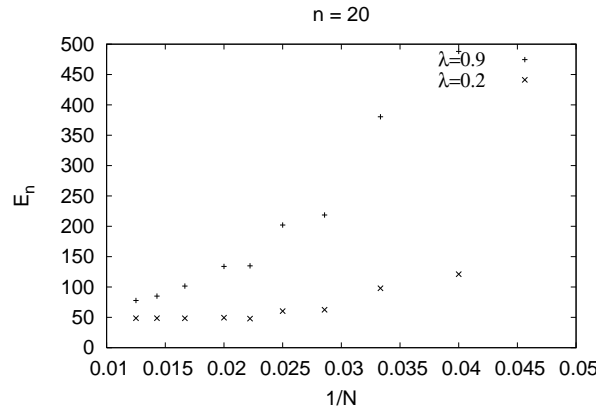
Σχήμα 9.2: Η ενεργειακή στάθμη  $E_9(\lambda)$  για  $\lambda = 0.2, 0.9$  υπολογίζεται από το όριο των ιδιοτιμών  $E_9(N, \lambda)$ , καθώς  $N \rightarrow \infty$ .

είναι η πραγματική ιδιοτιμή της Χαμιλτονιανής  $H(\lambda)$ . Στην πράξη, το όριο 9.22 για δεδομένα  $\lambda$  και  $n$  υπολογίζεται από τις αριθμητικές τιμές  $E_n(N, \lambda)$  για ολοένα και μεγαλύτερες τιμές του  $N$ . Αν επιτευχθεί σύγκλιση σε ένα επιθυμητό επίπεδο για τις τιμές του  $N$  που είναι εφικτές, τότε το όριο είναι μια προσέγγιση στην  $E_n(\lambda)$ . Η διαδικασία αυτή δείχνεται γραφικά στα σχήματα 9.1-9.3 για  $\lambda = 0.2, 0.9$ . Η σύγκλιση είναι ικανοποιητική για αρκετά μικρά  $N$  για  $n = 0, 9$ , αλλά για  $n = 20$  είναι αναγκαίοι υπολογισμοί σε μεγαλύτερες τιμές του  $N$ . Η αύξηση του  $n$  για δεδομένο  $\lambda$  κάνει αναγκαία την πρόσβαση με μεγαλύτερες τιμές του  $N$ . Για δεδομένο ενεργειακό επίπεδο  $n$ , η αύξηση του  $\lambda$  κάνει επίσης αναγκαίο τον υπολογισμό σε μεγαλύτερα  $N$ . Μια πλήρης συνεδρία υπολογισμού της ενέργειας της θεμελιώδους κατάστασης  $E_0(\lambda = 0.9)$  παρατίθεται παρακάτω<sup>6</sup>:

```
> tcsh
> gfortran -O2 anharmonic.f90 -llapack -lblas -o an
> foreach N (4 8 12 16 24 32)
foreach? (echo $N; echo 0.9) |./an >> data
foreach? end
> grep ^EV data | awk '{ print $2,$4 }'
4 0.711467845686790
8 0.786328966767866
```

<sup>6</sup>Ο foreach βρόχος είναι ειδικός για το φλοιό tcsh. Για το λόγο αυτό δίνεται ρητά η εντολή tcsh. Αν χρησιμοποιείτε διαφορετικό φλοιό χρησιμοποιείστε το αντίστοιχο συντακτικό.





Σχήμα 9.3: Η ενεργειακή στάθμη  $E_{20}(\lambda)$  για  $\lambda = 0.2, 0.9$  υπολογίζεται από το όριο των ιδιοτιμών  $E_{20}(N, \lambda)$ , καθώς  $N \rightarrow \infty$ . Η σύγκλιση δεν έχει επιτευχθεί για τις προβεβλημένες τιμές του  $N \leq 80$ .

```
12 0.785237674919165
16 0.784964461939594
24 0.785032515135677
32 0.785031492177730
> gnuplot
gnuplot> plot "<grep ^EV data | awk '{print 1/$2,$4}'"
```

Περαιτέρω αυτοματοποίηση της διαδικασίας μπορεί να βρεθεί στο σε-νάριο φλοιού `anharmmonic.csh` στο συνοδευτικό λογισμικό. Παρατηρήστε τη σύγκλιση για μεγάλο  $N$  της ιδιοτιμής  $E_0(N, 0.9)$  και ότι τελικά παίρνουμε  $E_0(0.9) \approx 0.78503$ . Αν είναι επιθυμητή η επίτευξη μεγαλύτερης ακρίβειας, τότε είναι αναγκαίο να γίνουν υπολογισμοί σε μεγαλύτερα  $N$ .

Είναι, επίσης, δυνατόν να υπολογίσουμε τις αναμενόμενες τιμές  $\langle A \rangle_n(\lambda)$  ενός τελεστή  $A = A(p, q)$ , όταν ο αναρμονικός ταλαντωτής βρίσκεται στην κατάσταση  $|n\rangle_\lambda$ :

$$\langle A \rangle_n(\lambda) = {}_\lambda \langle n | A | n \rangle_\lambda. \quad (9.24)$$

Στην πράξη, η αναμενόμενη τιμή θα υπολογιστεί από το όριο

$$\langle A \rangle_n(\lambda) = \lim_{N \rightarrow \infty} \langle A \rangle_n(N, \lambda) \equiv \lim_{N \rightarrow \infty} {}_{N, \lambda} \langle n | A | n \rangle_{N, \lambda}, \quad (9.25)$$

όπου  $|n\rangle_{N, \lambda}$  είναι τα ιδιοδιανύσματα του πεπερασμένου  $N \times N$  πίνακα  $H_{nm}(\lambda)$  τα οποία υπολογίζονται αριθμητικά από την DSYEV. Αυτά καθορίζονται από τις συνιστώσες τους  $c_m(N, \lambda)$ , όπου

$$|n\rangle_{N, \lambda} = \sum_{m=0}^{N-1} c_m(N, \lambda) |m\rangle, \quad (9.26)$$

οι οποίες αποθηκεύονται στο array H, αφού επιστρέψει η DSYEV:

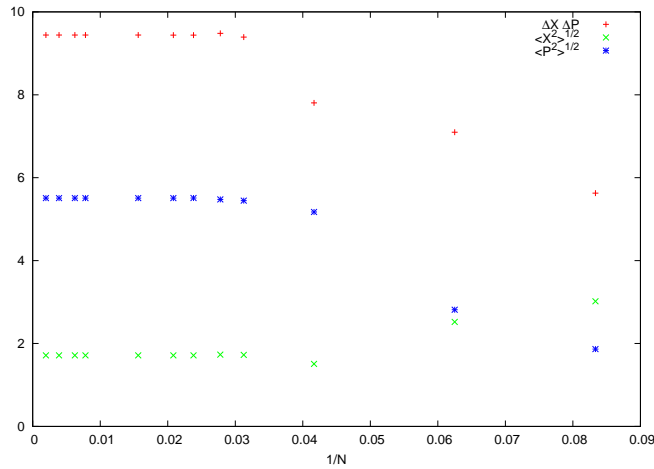
$$c_m(N, \lambda) = H(m+1, n+1). \quad (9.27)$$

Αντικαθιστώντας την εξίσωση (9.26) στην (9.24), παίρνουμε

$$\langle A \rangle_n(\lambda) = \sum_{m,m'=0}^{N-1} c_m^*(N, \lambda) c_{m'}(N, \lambda) A_{mm'}, \quad (9.28)$$

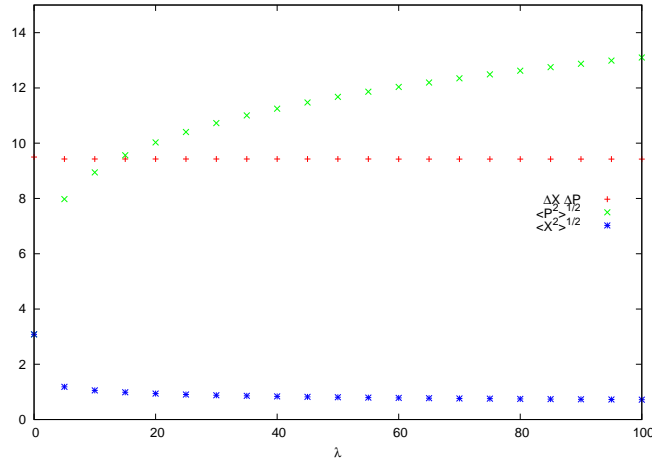
και μπορούμε να χρησιμοποιήσουμε την (9.27) στον υπολογισμό του αθροίσματος.

Τα παραπάνω θα τα εφαρμόσουμε στον υπολογισμό των αναμενόμενων τιμών των τελεστών  $x^2$ ,  $x^4$  και  $p^2$ . Λαμβάνοντας υπόψη ότι  $\langle x \rangle_n = \langle p \rangle_n = 0$ , παίρνουμε και για τις αβεβαιότητες  $\Delta x_n \equiv \sqrt{\langle x^2 \rangle_n - \langle x \rangle_n^2} = \sqrt{\langle x^2 \rangle_n}$  και  $\Delta p_n = \sqrt{\langle p^2 \rangle_n}$ . Από αυτές μπορούμε να υπολογίσουμε και το γινόμενο των αβεβαιοτήτων που θα πρέπει να ικανοποιεί τη σχέση αβεβαιότητας του Heisenberg  $\Delta x_n \cdot \Delta p_n \gtrsim 1/2$ . Τα αποτελέσματα παρουσιάζονται στον πίνακα 9.1 και στα σχήματα 9.4-9.5 και ο υπολογισμός αφήνεται ως άσκηση για τον αναγνώστη.



Σχήμα 9.4: Οι αναμενόμενες τιμές  $\langle x^2 \rangle_n^{1/2}(\lambda)$ ,  $\langle p^2 \rangle_n^{1/2}(\lambda)$  και το γινόμενο αβεβαιοτήτων  $\Delta x_n \cdot \Delta p_n$  για  $n = 9$  και  $\lambda = 0.5$  υπολογίζονται από το όριο των  $\langle x^2 \rangle_n^{1/2}(N, \lambda)$ ,  $\langle p^2 \rangle_n^{1/2}(N, \lambda)$  καθώς  $N \rightarrow \infty$ .

Τέλος, θα προσπαθήσουμε να καταλάβουμε καλύτερα τη φυσική του αναρμονικού ταλαντωτή μελετώντας τις ιδιότητες του φάσματός του, καθώς  $\lambda \rightarrow \infty$ . Όπως παρατηρούμε στο σχήμα 9.5, καθώς μεγαλώνει το  $\lambda$ , ο όρος  $\lambda x^4$  επικρατεί και η αναμενόμενη τιμή  $\langle x^2 \rangle_n(\lambda)$  ελαττώνεται, άρα ευνοούνται καταστάσεις που περιορίζουν τον ταλαντωτή σε



Σχήμα 9.5: Οι αναμενόμενες τιμές  $\langle x^2 \rangle_n^{1/2}(\lambda)$ ,  $\langle p^2 \rangle_n^{1/2}(\lambda)$  και το γινόμενο αβεβαιοτήτων  $\Delta x_n \cdot \Delta p_n$  για  $n = 9$ .

μικρότερη περιοχή του χώρου. Από την αρχή της αβεβαιότητας συμπεραίνουμε ότι η τυπική ορμή του ταλαντωτή θα αυξάνει κατά μέτρο. Αυτό επιβεβαιώνεται από το σχήμα 9.5 όπου βλέπουμε την αναμενόμενη τιμή  $\langle p^2 \rangle_n(\lambda)$  να αυξάνει με το  $\lambda$ . Για να δούμε ποσοτικά το αποτέλεσμα των ανταγωνιστικών αυτών τάσεων θα χρησιμοποιήσουμε ένα απλό επιχείρημα βάθμισης του Symanzik. Στη Χαμιλτονιανή  $H(\lambda) = p^2/2 + x^2/2 + \lambda x^4$  επαναορίζουμε  $x \rightarrow \lambda^{-1/6}x$  (άρα και  $p \rightarrow \lambda^{1/6}p$ ) και για  $\lambda$  αρκετά μεγάλο παίρνουμε<sup>7</sup> την ασυμπτωτική συμπεριφορά

$$H(\lambda) \sim \lambda^{1/3} h(1), \quad \lambda \rightarrow \infty, \quad (9.29)$$

όπου  $h(\lambda) = p^2/2 + \lambda x^4$  είναι η Χαμιλτονιανή του αναρμονικού “ταλαντωτή” με  $\omega = 0$ . Εφόσον ο τελεστής  $h(1)$  είναι ανεξάρτητος του  $\lambda$ , το ενεργειακό φάσμα θα έχει την ασυμπτωτική συμπεριφορά

$$E_n(\lambda) \sim C_n \lambda^{1/3}, \quad \lambda \rightarrow \infty. \quad (9.30)$$

Στην εργασία [41] δείχνεται ότι για  $\lambda > 100$  ισχύει ότι

$$E_0(\lambda) = \lambda^{1/3} (0.667\,986\,259\,18 + 0.143\,67\lambda^{-2/3} - 0.0088\lambda^{-4/3} + \dots), \quad (9.31)$$

με ακρίβεια καλύτερη από ένα μέρος στα  $10^6$ . Για μεγάλες τιμές του  $n$ , στην ίδια εργασία υπολογίζεται η ασυμπτωτική συμπεριφορά

$$E_n(\lambda) \sim C \lambda^{1/3} \left( n + \frac{1}{2} \right)^{4/3}, \quad \lambda \rightarrow \infty, n \rightarrow \infty, \quad (9.32)$$

<sup>7</sup>Για  $x \rightarrow \lambda^{-1/6}x$ ,  $H \rightarrow \lambda^{1/3}(p^2/2 + \lambda^{-2/3}x^2/2 + x^4)$  οπότε στο όριο  $\lambda \rightarrow \infty$  ο δεύτερος όρος εξαφανίζεται και παίρνουμε την (9.29).

$n$	$\lambda = 0.5$			$\lambda = 2.0$		
	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x \cdot \Delta p$	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x \cdot \Delta p$
0	0.305814	0.826297	0.502686	0.21223	1.19801	0.504236
1	0.801251	2.83212	1.5064	0.540792	4.21023	1.50893
2	1.15544	5.38489	2.49438	0.761156	8.15146	2.49089
3	1.46752	8.28203	3.48627	0.958233	12.6504	3.48166
4	1.75094	11.4547	4.47845	1.13698	17.596	4.47285
5	2.01407	14.8603	5.47079	1.30291	22.9179	5.46443
6	2.2617	18.4697	6.4632	1.45905	28.5683	6.45619
7	2.49696	22.2616	7.45562	1.60735	34.5124	7.44805
8	2.72198	26.2196	8.44804	1.74919	40.7234	8.43998
9	2.93836	30.3306	9.44045	1.88558	47.1801	9.43194

Πίνακας 9.1: Οι αναμενόμενες τιμές των  $\langle x^2 \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x \cdot \Delta p$  για τον αναρμονικό ταλαντωτή για τις καταστάσεις  $|n\rangle$ ,  $n = 0, \dots, 9$ . Παρατηρούμε τη μείωση της  $\Delta x = \sqrt{\langle x^2 \rangle}$  και αύξηση της  $\Delta p = \sqrt{\langle p^2 \rangle}$ , καθώς αυξάνεται το  $\lambda$ . Το γινόμενο  $\Delta x \cdot \Delta p$  φαίνεται να είναι πολύ κοντά στις τιμές  $(n + 1/2)$  που παίρνουμε από τον αρμονικό ταλαντωτή και για τις δύο τιμές του  $\lambda$ .

όπου  $C = 3^{4/3}\pi^2/\Gamma(1/4)^{8/3} \approx 1.376\,507\,40$ . Η σχέση αυτή εξετάζεται στο σχήμα 9.6 όπου βρίσκουμε ικανοποιητική συμφωνία με τους υπολογισμούς μας.

## 9.4 Το Διπλό Πηγάδι Δυναμικού

Θα χρησιμοποιήσουμε τις μεθόδους πινάκων που αναφέραμε για να υπολογίσουμε τα ενεργειακά επίπεδα σωματιδίου μέσα στο διπλό πηγάδι δυναμικού. Αυτό δίνεται από τη Χαμιλτονιανή:

$$H = \frac{p^2}{2} - \frac{x^2}{2} + \lambda \frac{x^4}{4} \quad (9.33)$$

και τα σημεία ισορροπίας στην κλασική κίνηση βρίσκονται στα ελάχιστα:

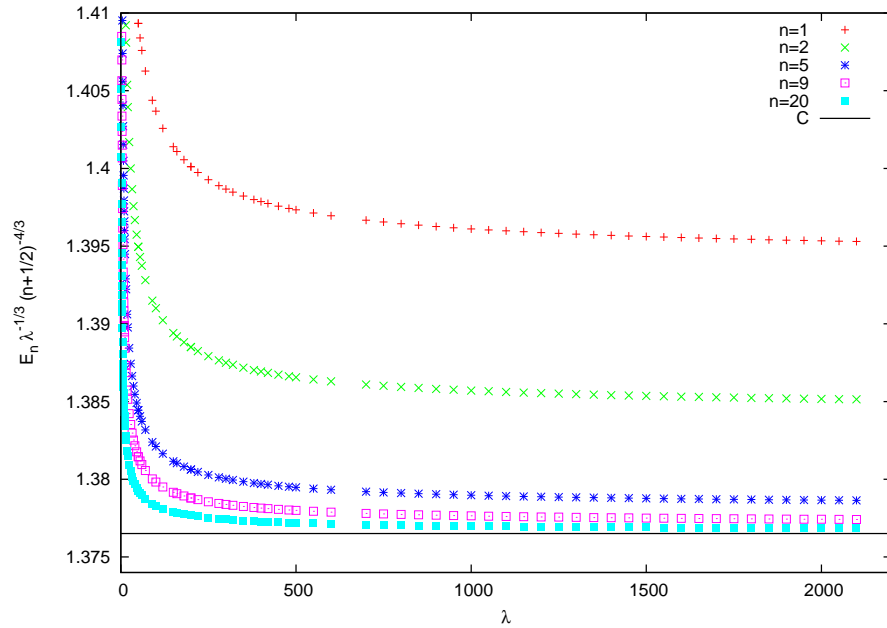
$$x_0 = \pm \frac{1}{\sqrt{\lambda}}, V_{min} = -\frac{1}{4\lambda} \quad (9.34)$$

Όταν το πηγάδι είναι πολύ βαθύ τότε, για τις χαμηλότερες στάθμες, μπορούμε να θεωρήσουμε ότι το δυναμικό προσεγγίζεται αρκετά καλά από αυτό του αρμονικού ταλαντωτή με συχνότητα  $\omega^2 = V''(x_0)$ , οπότε

$$E_{min} \approx V_{min} + \frac{1}{2}\omega \quad (9.35)$$

Πίνακας 9.2: Αριθμητικός υπολογισμός των ενεργειακών επιπέδων του αναρμονικού ταλαντωτή όπως δίνεται στην εργασία [41].

$\lambda$	$E_0$	$E_1$	$E_2$	$E_3$	$E_4$
0.002	0.501 489 66	1.507 419 39	2.519 202 12	3.536 744 13	4.559 955 56
0.006	0.504 409 71	1.521 805 65	2.555 972 30	3.606 186 33	4.671 800 37
0.01	0.507 256 20	1.535 648 28	2.590 845 80	3.671 094 94	4.774 913 12
0.05	0.532 642 75	1.653 436 01	2.873 979 63	4.176 338 91	5.549 297 81
0.1	0.559 146 33	1.769 502 64	3.138 624 31	4.628 882 81	6.220 300 90
0.3	0.637 991 78	2.094 641 99	3.844 782 65	5.796 573 63	7.911 752 73
0.5	0.696 175 82	2.324 406 35	4.327 524 98	6.578 401 95	9.028 778 72
0.7	0.743 903 50	2.509 228 10	4.710 328 10	7.193 265 28	9.902 610 70
1	0.803 770 65	2.737 892 27	5.179 291 69	7.942 403 99	10.963 5831
2	0.951 568 47	3.292 867 82	6.303 880 57	9.727 323 19	13.481 2759
50	2.499 708 77	8.915 096 36	17.436 9921	27.192 6458	37.938 5022
200	3.930 931 34	14.059 2268	27.551 4347	43.005 2709	60.033 9933
1000	3.694 220 85	23.972 2061	47.017 3387	73.419 1140	102.516 157
8000	13.366 9076	47.890 7687	93.960 6046	146.745 512	204.922 711
20000	18.137 2291	64.986 6757	127.508 839	199.145 124	278.100 238
$\lambda$	$E_5$	$E_6$	$E_7$	$E_8$	
0.002	5.588 750 05	6.623 044 60	7.662 759 33	8.707 817 30	
0.006	5.752 230 87	6.846 948 47	7.955 470 29	9.077 353 66	
0.01	5.901 026 67	7.048 326 88	8.215 837 81	9.402 692 31	
0.05	6.984 963 10	8.477 397 34	10.021 9318	11.614 7761	
0.1	7.899 767 23	9.657 839 99	11.487 3156	13.378 9698	
0.3	10.166 4889	12.544 2587	15.032 7713	17.622 4482	
0.5	11.648 7207	14.417 6692	17.320 4242	20.345 1931	
0.7	12.803 9297	15.873 6836	19.094 5183	22.452 9996	
1	14.203 1394	17.634 0492	21.236 4362	24.994 9457	
2	17.514 1324	21.790 9564	26.286 1250	30.979 8830	
50	49.516 4187	61.820 3488	74.772 8290	88.314 3280	
200	78.385 6232	97.891 3315	118.427 830	139.900 400	
1000	133.876 891	167.212 258	202.311 200	239.011 580	
8000	267.628 498	334.284 478	404.468 350	477.855 700	
20000	363.201 843	453.664 875	548.916 140	648.515 330	



Σχήμα 9.6: Επαλήθευση της ασυμπτωτικής σχέσης (9.32). Στον κάθετο άξονα έχουμε υπολογίσει την ποσότητα  $E_n \lambda^{-1/3} (n+1/2)^{-4/3}$  η οποία για αρκετά μεγάλα  $n$  και  $\lambda$  θα πρέπει να προσεγγίζει ασυμπτωτικά την τιμή  $C = 3^{4/3} \pi^2 / \Gamma(1/4)^{8/3} \approx 1.37650740$  (οριζόντια γραμμή).

Στην περίπτωση αυτή το φαινόμενο σήραγγας είναι πολύ ασθενές, με αποτέλεσμα τα ενεργειακά επίπεδα να χωρίζονται ελαφρά μεταξύ τους ανά ζεύγη. Αυτό γίνεται γιατί οι αντίστοιχες ιδιοκαταστάσεις είναι συμμετρικοί και αντισυμμετρικοί συνδυασμοί καταστάσεων που αντιστοιχούν σε καταστάσεις εντοπισμένες στο αριστερό ή δεξιό ελάχιστο της δυναμικής ενέργειας. Π.χ. για τα δύο χαμηλότερα ενεργειακά επίπεδα περιμένουμε ότι

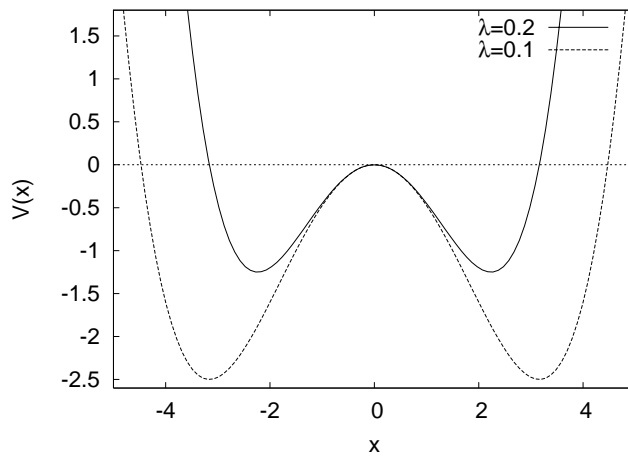
$$E_{0,1} \approx E_{min} \pm \frac{\Delta}{2}, \quad (9.36)$$

όπου  $\Delta \ll |E_{min}|$  και

$$|0\rangle_\lambda \approx \frac{|+\rangle + |-\rangle}{\sqrt{2}}, \quad |1\rangle_\lambda \approx \frac{|+\rangle - |-\rangle}{\sqrt{2}}, \quad (9.37)$$

όπου οι καταστάσεις  $|+\rangle$  και  $|-\rangle$  είναι εντοπισμένες στο αριστερό και δεξί πηγάδι του δυναμικού αντίστοιχα (δείτε και τα σχήματα 10.4 του Κεφαλαίου 10).

Ως βάση για τον υπολογισμό της Χαμιλτονιανής (9.33) θα χρησιμοποιήσουμε τις σχέσεις (9.12). Οι απαραίτητες μεταβολές στον κώδικα



Σχήμα 9.7: Η δυναμική ενέργεια  $V(x)$  για  $\lambda = 0.1, 0.2$ .

μας είναι ελάχιστες. Απλά θα προσθέσουμε μία ρουτίνα που να υπολογίζει τους πίνακες  $p_{nm}$ . Παίρνουμε έτσι τον κώδικα που αποθηκεύουμε στο αρχείο `doublewell.f90`:

```
!=====
program doublewell_elevels
!=====
! H      : Hamiltonian operator H0+(lambda/4)*X^4
! H0     : Hamiltonian H0=1/2 P^2-1/2 X^2
! X,X2,X4: Position operator and its powers
! iP     : i P operator
! P2     : P^2 = -(iP)(iP) operator
! E      : Energy eigenvalues
! WORK   : Workspace for lapack routine DSYEV
!=====
implicit none
integer,parameter :: P=1000
integer,parameter :: LWORK=3*P-1
real(8),dimension(P,P) :: H,H0,X,X4,X2,iP,P2
real(8),dimension(P) :: E
real(8),dimension(LWORK) :: WORK
real(8) :: lambda,lambda0,lambdaf,dlambda
integer :: DIM0,DIMF,dDIM,DIM
integer :: i

!Minimum and maximum values of Hilbert space dimensions:
print *, 'Enter Hilbert Space dimensions (DIM0,DIMF,DDIM): '
read *, DIM0,DIMF,DDIM
!Minimum and maximum values of lambda (step dlambda):
```

```

print *, 'Enter lambda0, lambdaf, dlambda: '
read *, lambda0, lambdaf, dlambda
print *, 'lambda0= ', lambda0
!Print Message:
print *, '# #####'
print *, '# Energy levels of double well potential'
print *, '# using matrix methods.'
print *, '# Hilbert Space Dimensions = ', DIM0, ' - ', DIMF, &
  ' step= ', dDIM
print *, '# lambda coupling = ', lambda0, ' - ', lambdaf, &
  ' step= ', dlambda
print *, '# #####'
print *, '# Output: DIM lambda E_0 E_1 ... E_{N-1}'
print *, '# _____'

do DIM=DIM0, DIMF, dDIM

  call calculate_operators(X, X2, X4, iP, P2, H0, DIM)

  lambda = lambda0
  do while (lambda .le. lambdaf)
    call calculate_evs(H, H0, X4, E, WORK, lambda, DIM)
    write(6,100) 'EV ', DIM, lambda, (E(i), i=1, DIM)
    lambda = lambda+dlambda
  enddo
enddo
100 FORMAT(A3, I5, 1000G25.15)
end program doublewell_elevels
!=====
subroutine calculate_evs(H, H0, X4, E, WORK, lambda, DIM)
!=====
  implicit none
  integer, parameter :: P=1000
  integer, parameter :: LWORK=3*P-1
  real(8), dimension(P,P) :: H, H0, X4
  real(8), dimension(P) :: E
  real(8), dimension(LWORK) :: WORK
  integer :: DIM
  real(8) :: lambda
  character(1) :: JOBZ, UPLO
  integer :: LDA, INFO, i, j

  call calculate_H(H, H0, X4, lambda, DIM)
  JOBZ='V'; UPLO='U'
  call DSYEV(JOBZ, UPLO, DIM, H, P, E, WORK, LWORK, INFO)
  print *, '# ***** EVEC *****'
  do j=1, DIM
    write(6,101) '# EVEC ', DIM, lambda, (H(i, j), i=1, DIM)
  enddo

```



```

print *, '# ***** EVEC ***** '
101 FORMAT(A7,I5,F8.4,1000G14.6)

if(INFO .ne. 0)then
  print *, 'dsyev failed. INFO= ', INFO
  stop
endif

end subroutine calculate_evs
!=====
subroutine calculate_H(H,H0,X4,lambda,DIM)
!=====
  implicit none
  integer ,parameter :: P=1000
  real(8),dimension(P,P) :: H,H0,X4
  integer :: DIM
  real(8) :: lambda
  integer :: i,j

  do j=1,DIM
    do i=1,DIM
      H(i,j)=H0(i,j)+0.25D0*lambda*X4(i,j)
    enddo
  enddo

  print *, '# ***** H ***** '
  do j=1,DIM
    write(6,102) '# HH ',(H(i,j), i=1,DIM)
  enddo
  print *, '# ***** H ***** '

102 FORMAT(A5,1000G14.6)
end subroutine calculate_H
!=====
subroutine calculate_operators(X,X2,X4,iP,P2,H0,DIM)
!=====
  implicit none
  integer ,parameter :: P=1000
  real(8),dimension(P,P) :: X,X4,X2,iP,P2,H0
  integer :: DIM
  integer :: i,j,m,n
  real(8),parameter :: isqrt2=1.0D0/sqrt(2.0D0)

  X =0.0D0;X2=0.0D0;X4=0.0D0
  iP=0.0D0;P2=0.0D0

  do i=1,DIM
    n=i-1 !indices 0,...,DIM-1
    ! The delta_{n,m+1} term, i.e. m=n-1

```

```

m=n-1 !energy level: n -> i=n+1, m-> j=m+1
j=m+1
if(j.ge.1) X (i,j) = isqrt2*sqrt(DBLE(m+1))
if(j.ge.1) iP(i,j) = -isqrt2*sqrt(DBLE(m+1))
! The delta_{n,m-1} term, i.e. m=n+1
m=n+1
j=m+1
if(j.le.DIM) X (i,j) = isqrt2*sqrt(DBLE(m))
if(j.le.DIM) iP(i,j) = isqrt2*sqrt(DBLE(m))
enddo !do i=1,DIM

X2 = MATMUL( X, X)
P2 = -MATMUL(iP,iP)
X4 = MATMUL(X2,X2)

!The Hamiltonian:
H0 = 0.5D0*(P2-X2)
end subroutine calculate_operators

```

Πού προτιμάει να βρίσκεται το σωματίο, όταν βρίσκεται στις καταστάσεις  $|+\rangle$  και  $|-\rangle$ ; Αυτό μπορεί να απαντηθεί αν υπολογίσουμε την αναμενόμενη τιμή του τελεστή θέσης  $\langle x \rangle$  σε κάθε μια από τις καταστάσεις. Γνωρίζουμε ότι, όταν το σωματίδιο είναι σε κάποια από τις ενεργειακές ιδιοκαταστάσεις, έχουμε

$$\langle x \rangle_n(\lambda) = {}_\lambda \langle n | x | n \rangle_\lambda = 0 \quad (9.38)$$

εξαιτίας της αρτιότητας του δυναμικού  $V(x) = V(-x)$ . Άρα,

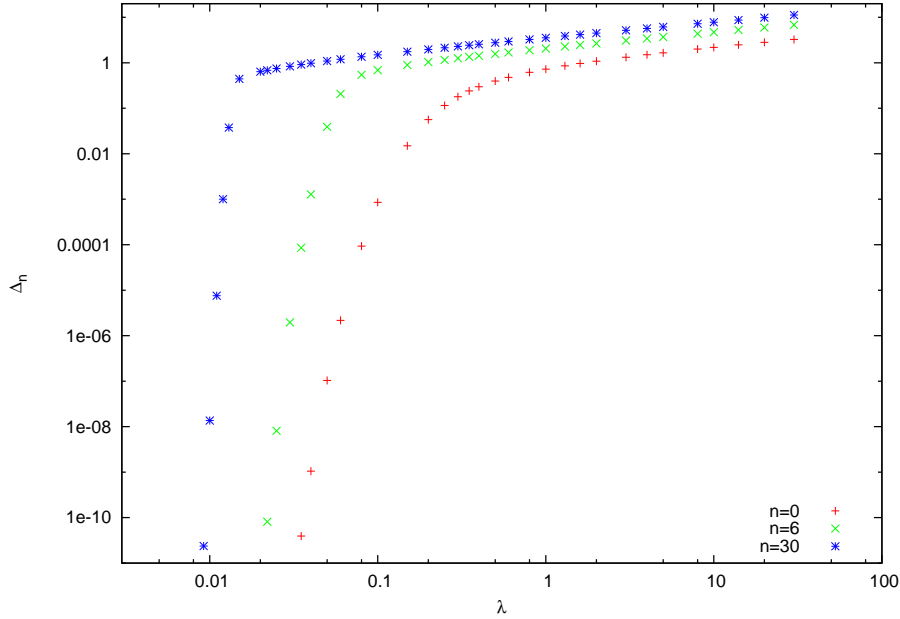
$$\begin{aligned}
 \langle x \rangle_\pm(\lambda) &= \langle \pm | x | \pm \rangle \\
 &= \frac{1}{\sqrt{2}} ({}_\lambda \langle 0 | x | 0 \rangle_\lambda \pm {}_\lambda \langle 1 | x | 0 \rangle_\lambda \pm {}_\lambda \langle 0 | x | 1 \rangle_\lambda + {}_\lambda \langle 1 | x | 1 \rangle_\lambda) \\
 &= \pm \sqrt{2} \langle 1 | x | 0 \rangle_\lambda,
 \end{aligned} \quad (9.39)$$

όπου στην τελευταία γραμμή χρησιμοποιήσαμε τη σχέση (9.38)  ${}_\lambda \langle 0 | x | 0 \rangle_\lambda = {}_\lambda \langle 1 | x | 1 \rangle_\lambda = 0$  και ότι τα πλάτη  ${}_\lambda \langle 1 | x | 0 \rangle_\lambda = {}_\lambda \langle 0 | x | 1 \rangle_\lambda$ . Επίσης<sup>8</sup> ισχύει ότι  ${}_\lambda \langle 1 | x | 0 \rangle_\lambda > 0$ . Αν λοιπόν έχουμε ότι  $|0\rangle_\lambda = \sum_{m=0}^{\infty} c_m^{(0)} |m\rangle$  και  $|1\rangle_\lambda = \sum_{m=0}^{\infty} c_m^{(1)} |m\rangle$ , παίρνουμε

$$\langle x \rangle_\pm(\lambda) = \pm \sqrt{2} \sum_{m,m'=0}^{\infty} c_m^{(0)} c_{m'}^{(1)} X_{mm'}. \quad (9.40)$$

<sup>8</sup>Για να πειστείτε, κοιτάξτε τις κυματοσυναρτήσεις στα σχήματα 10.4 του Κεφαλαίου 10 και υπολογίστε τα σχετικά ολοκληρώματα.

Δεδομένου ότι για πεπερασμένο  $N$ , η DSYEV μας επιστρέφει προσεγγιστικά τους συντελεστές  $c_m^{(n)}$  στις στήλες του πίνακα  $H(\text{DIM}, \text{DIM})$ , έτσι ώστε  $c_m^{(n)} \approx H(m+1, n+1)$ , μπορείτε να συγκρίνετε την τιμή του  $\langle x \rangle_{\pm}(\lambda)$  με τις κλασικές τιμές  $x_0 = \pm 1/\sqrt{\lambda}$ , καθώς το  $\lambda$  αυξάνεται.



Σχήμα 9.8: Υπολογισμός της διαφοράς ενεργειακών επιπέδων  $\Delta_n = E_{n+1} - E_n$  για  $n = 0, 6, 30$  για το διπλό πηγάδι δυναμικού από το πρόγραμμα `doublewell.f90`. Η διαφορά τείνει στο μηδέν, καθώς το πηγάδι δυναμικού γίνεται βαθύτερο, όταν το  $\lambda$  ελαττώνεται. Οι καταστάσεις  $|\pm\rangle = (|n+1\rangle_{\lambda} \pm |n\rangle_{\lambda})/\sqrt{2}$  εντοπίζονται τότε ολοένα και περισσότερο στο δεξί ή αριστερό πηγάδι αντίστοιχα.

## 9.5 Ασκήσεις

- 9.1 Υπολογίστε αναλυτικά τον πίνακα  $H(\lambda)$  για  $\lambda = 2, 3$ . Υπολογίστε τις ιδιοτιμές για  $N = 2$ . Συγκρίνετε με τις τιμές που υπολογίζει το πρόγραμμά σας ως επιβεβαίωση ότι τρέχει σωστά.
- 9.2 Μεταβάλετε τον κώδικα `test.f90`, έτσι ώστε να επιβεβαιώνει ότι τα ιδιοδιανύσματα ικανοποιούν τις σχέσεις  $\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$  και ότι αποτελούν ορθοκανονική βάση  $\mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$ .
- 9.3 Να υπολογίσετε τις ενεργειακές τιμές  $E_5(\lambda)$  και  $E_9(\lambda)$  για  $\lambda = 0.8, 1.2$  με ακρίβεια καλύτερη από 0.01%.
- 9.4 Μέχρι ποιο ενεργειακό επίπεδο μπορείτε να υπολογίσετε με ακρίβεια καλύτερη του 2% όταν  $N = 64$ ; Πάρτε  $\lambda = 1$ .
- 9.5 Να υπολογίσετε τις ενεργειακές τιμές  $E_3$  και  $E_{12}$  για  $0 \leq \lambda \leq 4$  με βήμα  $\delta\lambda = 0.2$  με ακρίβεια καλύτερη του 0.01%. Πόσο μεγάλο  $N$  πρέπει να πάρετε για κάθε ενεργειακό επίπεδο, έτσι ώστε να πετύχετε το στόχο σας;
- 9.6 Υπολογίστε τον πίνακα του τελεστή  $x^4$  αναλυτικά και προγραμματίστε τον στο πρόγραμμά σας. Συγκρίνετε τους χρόνους που τρέχει το πρόγραμμά σας σε σχέση με πριν σαν συνάρτηση του  $N$ .
- 9.7 Μεταβάλετε τον κώδικα του `anharmonic.f90`, έτσι ώστε τα arrays `H`, `X`, `X4`, `E`, `WORK` να είναι `ALLOCATABLE` και να έχουν διάσταση που να καθορίζεται από τη μεταβλητή `DIM` που δίνει ο χρήστης τη στιγμή που τρέχει το πρόγραμμα. (Υπόδειξη: Δείτε το πρόγραμμα `anharmonicSPEED.f90` στο συνοδευτικό λογισμικό.)
- 9.8 Προσπαθήστε να αναπαράγετε τα αποτελέσματα των Hioe και Montroll [41] που δίνονται στον πίνακα 9.2 για τις ενεργειακές στάθμες  $n = 3$  και  $n = 5$ . Μέχρι ποια τιμή του  $\lambda$  μπορείτε να το κάνετε με τους υπολογιστικούς πόρους που διαθέτετε;
- 9.9 Προσπαθήστε να αναπαράγετε το αποτέλεσμα των Hioe και Montroll [41] που δίνεται από τη σχέση (9.31). Υπολογίστε την ενέργεια της θεμελιώδους κατάστασης για  $200 < \lambda < 20000$  και προσαρμόστε τα δεδομένα σας σε συνάρτηση της μορφής  $\lambda^{1/3}(a + b\lambda^{-2/3} + c\lambda^{-4/3})$ . Με πόση ακρίβεια μπορείτε να προσδιορίσετε τους συντελεστές  $a$ ,  $b$  και  $c$  και πόσο καλά συμφωνούν με την εξίσωση (9.31);

- 9.10 Μεταβάλλετε τον κώδικα του `anharmonic.f90` έτσι ώστε να υπολογίζει τις αναμενόμενες τιμές  $\langle x^2 \rangle_n(N, \lambda)$ ,  $\langle p^2 \rangle_n(N, \lambda)$  και τα αντίστοιχα γινόμενα  $\Delta x \cdot \Delta p$ . (Υπόδειξη: Δείτε το πρόγραμμα `anharmonicOBS.f90` στο συνοδευτικό λογισμικό.)
- 9.11 Να αναπαράγετε τα αποτελέσματα του σχήματος 9.4. Στη συνέχεια, να επαναλάβετε τον υπολογισμό για  $\lambda = 2.0, 10.0, 100.0$ . Να κάνετε τους ίδιους υπολογισμούς για  $n = 20$ .
- 9.12 Να αναπαράγετε τα αποτελέσματα του σχήματος 9.5. Στη συνέχεια, να επαναλάβετε τον υπολογισμό  $n = 20$ .
- 9.13 Να αναπαράγετε τα αποτελέσματα του σχήματος 9.6. Στη συνέχεια, να επαναλάβετε τον υπολογισμό  $n = 3, 7, 12, 18, 24$ .
- 9.14 Να γράψετε πρόγραμμα που να υπολογίζει τις ενεργειακές στάθμες για τον αναρμονικό ταλαντωτή

$$H(\lambda, \mu) = \frac{1}{2}p^2 + \frac{1}{2}x^2 + \lambda x^4 + \mu x^6 \quad (9.41)$$

και να υπολογίσετε τις  $E_n(\lambda)$  για  $n = 0, 3, 8, 20$  για  $\lambda = 0.2$  και  $\mu = 0.2, 0.5, 1.0, 2.0, 10.0$ .

- 9.15 Να μεταβάλλετε το πρόγραμμα της προηγούμενης άσκησης, έτσι ώστε να υπολογίζει τις αναμενόμενες τιμές  $\langle x^2 \rangle_n(N, \lambda)$ ,  $\langle p^2 \rangle_n(N, \lambda)$  και τα αντίστοιχα γινόμενα  $\Delta x \cdot \Delta p$ . Να υπολογίσετε τις αναμενόμενες τιμές  $\langle x^2 \rangle_n(\lambda)$ ,  $\langle p^2 \rangle_n(\lambda)$  και  $\Delta x \cdot \Delta p$  για  $n = 0, 3, 8, 20$  για  $\lambda = 0.2$  και  $\mu = 0.2, 0.5, 1.0, 2.0, 10.0$ .
- 9.16 Να υπολογίσετε τα ζεύγη ενεργειακών τιμών  $E_n, E_{n+1}$  για  $n = 0, 4, 20$  χρησιμοποιώντας το πρόγραμμα `doublewell.f90` όταν  $\lambda = 0.2, 0.1, 0.05, 0.02$ . Να υπολογίσετε τη διαφορά  $\Delta_n = E_{n+1} - E_n$ . Τι παρατηρείτε;
- 9.17 Ορίστε τις τιμές ενέργειας

$$\epsilon_n = -\frac{1}{4\lambda} + \left(n + \frac{1}{2}\right).$$

Συγκρίνετε τις τιμές των ενεργειών  $E_n, E_{n+1}$  που υπολογίσατε στην προηγούμενη άσκηση με τις τιμές  $\epsilon_n - \Delta_n/2$  και  $\epsilon_n + \Delta_n/2$  αντίστοιχα. Τι παρατηρείτε και γιατί;

- 9.18 Να κάνετε τις απαραίτητες μετατροπές στο πρόγραμμα `doublewell.f90`, έτσι ώστε να υπολογίζει τις αναμενόμενες τιμές  $\langle x \rangle_{\pm}(\lambda)$  της σχέσης (9.40). Να συγκρίνετε τις τιμές  $\langle x \rangle_{\pm}(\lambda)$  με τις κλασικές τιμές  $x_0 = \pm 1/\sqrt{\lambda}$  για  $\lambda = 0.2, 0.1, 0.05, 0.02, 0.01$ .
- 9.19 Να επαναλάβετε την προηγούμενη άσκηση, όταν οι καταστάσεις  $|\pm\rangle = (1/\sqrt{2})(|n\rangle_{\lambda} \pm |n+1\rangle_{\lambda})$  για  $n = 6$  και  $n = 30$ .
- 9.20 Στον απλό αρμονικό ταλαντωτή, τα ενεργειακά επίπεδα ισαπέχουν μεταξύ τους, δηλ.  $\Delta_n = E_{n+1} - E_n = 1$ ,  $(\Delta_{n+2} - \Delta_n)/\Delta_n = 0$ . Να υπολογίσετε τις παραπάνω ποσότητες για τον αναρμονικό ταλαντωτή και το διπλό πηγάδι δυναμικού για  $\lambda = 1, 10, 100, 1000$  και για  $n = 0, 8, 20$ . Τι συμπεραίνετε σε κάθε περίπτωση;

## ΚΕΦΑΛΑΙΟ 10

### Η Εξίσωση Schrödinger

Στο κεφάλαιο αυτό γίνεται μελέτη της χρονοανεξάρτητης εξίσωσης Schrödinger για ένα μη σχετικιστικό σωματίο μάζας  $m$ , χωρίς σπιν, το οποίο κινείται σε μία διάσταση υπό την επίδραση ενός στατικού δυναμικού πεδίου που δίνεται από τη συνάρτηση δυναμικής ενέργειας (“δυναμικού”)  $V(x)$ . Θα περιοριστούμε σε δέσμιες καταστάσεις. Οι λύσεις της εξίσωσης στην περίπτωση αυτή δίνουν το διακριτό ενεργειακό φάσμα  $\{E_n\}$ , καθώς και τις αντίστοιχες ιδιοκαταστάσεις που δίνονται, σε αναπαράσταση θέσης, από τις κυματοσυναρτήσεις  $\psi_n(x)$ .

Από άποψη αριθμητικής ανάλυσης, το πρόβλημα είναι η λύση απλών διαφορικών εξισώσεων ιδιοτιμών με συνοριακές συνθήκες. Μέρος της λύσης είναι και η ιδιοτιμή της ενέργειας η οποία θα πρέπει, επίσης, να υπολογιστεί. Θα χρησιμοποιήσουμε δύο μεθόδους, μία για τη λύση του προβλήματος του σωματίου μέσα σε απειρόβαθο πηγάδι δυναμικού το οποίο είναι άρτιο  $V(x) = V(-x)$ , και μία που θα εφαρμόσουμε σε γενικότερα δυναμικά και προτιμάται στις εφαρμογές. Η πρώτη είναι μια εισαγωγή στο πρόβλημα και έχει μόνο ακαδημαϊκό και εκπαιδευτικό ενδιαφέρον και ο βιαστικός αναγνώστης μπορεί να παραλείψει τη μελέτη της παραγράφου 10.2 και να προχωρήσει στην παράγραφο 10.3.

#### 10.1 Εισαγωγή

Η εξίσωση Schrödinger που ικανοποιούν οι κυματοσυναρτήσεις  $\psi(x)$  που αναπαριστούν τις ιδιοκαταστάσεις ενέργειας είναι η

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x), \quad (10.1)$$

μαζί με τη συνθήκη κανονικοποίησης

$$\langle \psi | \psi \rangle = \int_{-\infty}^{+\infty} \psi^*(x) \psi(x) dx = 1. \quad (10.2)$$

Θυμίζουμε ότι ο τελεστής  $\hat{H}$  που σε αναπαράσταση θέσης δίνεται από

$$\hat{H} = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(\hat{x}), \quad (10.3)$$

είναι ερμιτιανός, δηλ.  $\hat{H}^\dagger = \hat{H}$ . Η εξίσωση (10.1) είναι μια εξίσωση ιδιοτιμών

$$\hat{H}\psi(x) = E\psi(x), \quad (10.4)$$

η οποία έχει λύσεις ένα διακριτό σύνολο από πραγματικές συναρτήσεις  $\psi_n^*(x) = \psi_n(x)$  τέτοιες, ώστε  $\hat{H}\psi_n(x) = E_n\psi_n(x)$ . Οι αριθμοί  $E_0 \leq E_1 \leq E_2 \leq \dots$  είναι πραγματικοί και αποτελούν το ενεργειακό φάσμα του σωματιδίου στο πεδίο<sup>1</sup>  $V(x)$ . Η ελάχιστη ενέργεια  $E_0$  αντιστοιχεί στη θεμελιώδη κατάσταση που βρίσκεται το σωματίδιο που δίνεται από μία μη τετριμμένη συνάρτηση  $\psi_0(x)$ . Σε συμφωνία με την αρχή απροσδιοριστίας του Heisenberg, στην κατάσταση αυτή  $\Delta p > 0$  και  $\Delta x > 0$ , έτσι ώστε  $\Delta p \cdot \Delta x \geq \hbar/2$ .

Οι ιδιοκαταστάσεις  $\psi_n(x)$  αποτελούν μια ορθοκανονική βάση

$$\langle \psi_n | \psi_m \rangle = \int_{-\infty}^{+\infty} \psi_n^*(x) \psi_m(x) dx = \delta_{n,m}. \quad (10.5)$$

έτσι ώστε οποιαδήποτε κυματοσυνάρτηση  $\phi(x)$  που αναπαριστά την κατάσταση  $|\phi\rangle$  να δίνεται από το γραμμικό συνδυασμό

$$\phi(x) = \sum_{n=0}^{\infty} c_n \psi_n(x). \quad (10.6)$$

Οι αριθμοί  $c_n = \langle \psi_n | \phi \rangle = \int_{-\infty}^{+\infty} \psi_n^*(x) \phi(x) dx$  δίνουν την πιθανότητα  $p_n = |c_n|^2$  να μετρηθεί η ενέργεια  $E_n$  στην κατάσταση  $|\phi\rangle$ .

Επίσης, θυμίζουμε ότι για οποιαδήποτε κατάσταση  $|\phi\rangle$  η συνάρτηση

$$p_\phi(x) = |\phi(x)|^2 = \phi^*(x) \phi(x) \quad (10.7)$$

---

<sup>1</sup>Το γεγονός ότι το ενεργειακό φάσμα του σωματιδίου είναι φραγμένο από κάτω εξαρτάται από τη μορφή του δυναμικού. Υποθέτουμε ότι το  $V(x)$  είναι τέτοιο, ώστε η ενέργεια  $E_0$  να είναι πεπερασμένη. Επίσης, στη μία διάσταση, το ενεργειακό φάσμα ενός σωματιδίου για γενικά δυναμικά είναι μη εκφυλισμένο (δείτε όμως λ.χ. S. Kar, R. Parwani, arXiv:0706.1135.)



είναι η πυκνότητα πιθανότητας εύρεσης του σωματιδίου στη θέση  $x$ , δηλ. η πιθανότητα εύρεσης του σωματιδίου στο διάστημα  $[x_1, x_2]$  δίνεται από τη σχέση

$$\mathcal{P}_\phi(x_1 < x < x_2) = \int_{x_1}^{x_2} \phi^*(x)\phi(x) dx. \quad (10.8)$$

Η σχέση κανονικοποίησης (10.2) σύμφωνα με την παραπάνω ερμηνεία απηχεί τη διατήρηση της πιθανότητας (ανεξάρτητη του χρόνου, ιδιότητα της χρονοεξαρτημένης εξίσωσης Schrödinger) και την πληρότητα (εδώ τη βεβαιότητα παρατήρησης του σωματιδίου κάπου στον άξονα των  $x$ ).

Οι μετρήσιμες ποσότητες του παραπάνω κβαντομηχανικού συστήματος δίνονται από τελεστές  $\hat{A}(\hat{x}, \hat{p})$  και οι αναμενόμενες τιμές τους, όταν το σύστημα είναι σε μια κατάσταση  $|\phi\rangle$ , δίνονται από τη σχέση

$$\langle \hat{A} \rangle_\phi = \langle \phi | \hat{A} | \phi \rangle = \int_{-\infty}^{+\infty} \phi^*(x) \hat{A}(\hat{x}, \hat{p}) \phi(x) dx. \quad (10.9)$$

Από αριθμητικής άποψης, το πρόβλημα ιδιοτιμών (10.1) ανάγεται στη λύση μιας διαφορικής εξίσωσης δεύτερης τάξης. Οι διαφορές σε σχέση με τις περιπτώσεις που μελετήσαμε σε προηγούμενα κεφάλαια είναι:

- Αντί να έχουμε πρόβλημα αρχικών τιμών (τιμές της συνάρτησης και παραγώγου σε ένα σημείο) έχουμε πρόβλημα συνοριακών τιμών (τιμές της συνάρτησης ή της παραγώγου σε δύο σημεία).
- Η ιδιοτιμή (ενέργεια) είναι άγνωστη και πρέπει να προσδιοριστεί σαν μέρος της λύσης.

Θα παρουσιάσουμε μερικές απλές μεθόδους επίλυσης του προβλήματος, ειδικές στη μία διάσταση, ως μια εισαγωγή στον τρόπο αριθμητικής λύσης ενός προβλήματος με τα παραπάνω χαρακτηριστικά.

Για την αριθμητική λύση της παραπάνω εξίσωσης καταφεύγουμε σε επανακανονικοποίηση των συναρτήσεων και των παραμέτρων τους, έτσι ώστε να έχουμε να κάνουμε με αδιάστατες ποσότητες. Για το λόγο αυτό, η (10.1) γράφεται αρχικά στη μορφή:

$$\frac{d^2}{dx^2} \psi(x) + \frac{2m}{\hbar^2} (E - V(x)) \psi(x) = 0. \quad (10.10)$$

Επιπλέον, επιλέγουμε μια χαρακτηριστική κλίμακα μήκους  $L$  στο πρόβλημα και επαναορίζουμε  $\tilde{x} = x/L$ . Ορίζουμε  $\tilde{\psi}(\tilde{x}) = \psi(x)$   $\tilde{\psi}'(\tilde{x}) = d\psi(x)/d\tilde{x} = L d\psi(x)/dx$  και παίρνουμε

$$\tilde{\psi}''(\tilde{x}) + \frac{2mL^2}{\hbar^2} (E - V(\tilde{x}L)) \tilde{\psi}(\tilde{x}) = 0. \quad (10.11)$$

Ορίζουμε  $v(\tilde{x}) = 2mL^2V(x)/\hbar^2 = 2mL^2V(\tilde{x}L)/\hbar^2$ ,  $\epsilon = 2mL^2E/\hbar^2$  και αλ-  
λάζουμε συμβολισμό  $\tilde{x} \rightarrow x$ ,  $\tilde{\psi} \rightarrow \psi$ , οπότε παίρνουμε

$$\psi''(x) = -(\epsilon - v(x))\psi(x). \quad (10.12)$$

Οι λύσεις της (10.1) παίρνονται εύκολα από τις λύσεις της (10.12) χρη-  
σιμοποιώντας το “λεξιλόγιο”<sup>2</sup>:

$$x \rightarrow \frac{x}{L}, \quad E = \frac{\hbar^2}{2mL^2}\epsilon, \quad V(x) = \frac{\hbar^2}{2mL^2}v(x/L). \quad (10.13)$$

Τέλος, να σημειώσουμε ότι, αν πάρουμε  $\tilde{p} = -i\partial/\partial\tilde{x} = -iL\partial/\partial x$  για την  
ορμή, τότε θα έχουμε

$$\tilde{p} = \frac{L}{\hbar}p. \quad (10.14)$$

Η σχέση μετάθεσης  $[x, p] = i\hbar$  γίνεται τώρα  $[\tilde{x}, \tilde{p}] = i$ . Για την κινητική  
ενέργεια  $T = \frac{p^2}{2m}$  έχουμε

$$T = \frac{\hbar^2}{2mL^2}\tilde{p}^2 = -\frac{\hbar^2}{2mL^2}\frac{\partial^2}{\partial\tilde{x}^2}, \quad (10.15)$$

και για τη Χαμιλτονιανή  $H = T + V$

$$H = \frac{\hbar^2}{2mL^2}(\tilde{p}^2 + v(\tilde{x})) = \frac{\hbar^2}{2mL^2}\left(-\frac{\partial^2}{\partial\tilde{x}^2} + v(\tilde{x})\right). \quad (10.16)$$

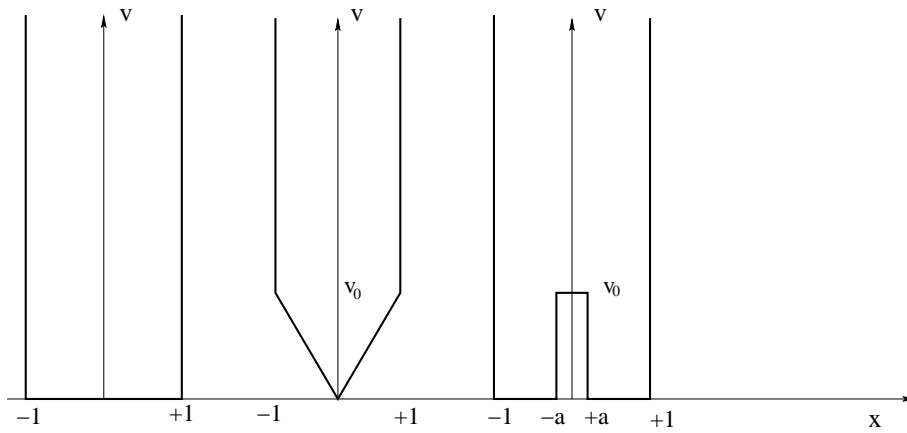
Στις εξισώσεις των επόμενων παραγράφων, η περισπωμένη θα παρα-  
λείπεται και θα γράφουμε  $x$  αντί για  $\tilde{x}$ .

## 10.2 Το απειρόβαθο πηγάδι δυναμικού

Το απλούστερο πρότυπο για τη μελέτη των ποιοτικών χαρακτηριστικών  
των δέσμιων καταστάσεων είναι το απειρόβαθο πηγάδι δυναμικού πλά-  
τους  $L$  όπου το σωματίο είναι περιορισμένο να βρίσκεται στο διάστημα  
 $[-L/2, L/2]$ :

$$v(x) = \begin{cases} 0 & |x| < L/2 \\ +\infty & |x| \geq L/2 \end{cases} \quad (10.17)$$

<sup>2</sup>Φυσικά αν κανονικοποιήσουμε τις λύσεις  $\tilde{\psi}(\tilde{x})$  της (10.12) σύμφωνα με τη σχέση  
 $\int_{-\infty}^{+\infty} \tilde{\psi}^*(\tilde{x})\tilde{\psi}(\tilde{x})d\tilde{x} = 1$ , θα πρέπει να πάρουμε και  $\psi(x) = (1/\sqrt{L})\tilde{\psi}(x/L)$  για να έχουμε  
σωστή κανονικοποίηση  $\int_{-\infty}^{+\infty} \psi^*(x)\psi(x)dx = 1$ .



Σχήμα 10.1: Τα τρία δυναμικά που δίνονται από τις εξισώσεις (10.17), (10.26) και (10.27).

Εδώ, σύμφωνα με τα λεγόμενα στο τέλος της προηγούμενης παραγράφου, έχουμε επιλέξει το  $L$  να είναι το πλάτος του πηγαδιού και η μεταβλητή  $x$  να είναι αδιάστατη και να αντιστοιχεί στο  $x/(L/2)$  όταν το  $x$  έχει διαστάσεις μήκους.

Η λύση της (10.12) υπολογίζεται εύκολα. Τα χαρακτηριστικά που πρέπει να τονίσουμε είναι ότι λόγω της συμμετρίας

$$v(-x) = v(x), \quad (10.18)$$

του δυναμικού (άρτια συνάρτηση της θέσης), οι λύσεις έχουν συγκεκριμένη ομοτιμία (parity), ιδιότητα που θα μας βοηθήσει σημαντικά στην αριθμητική αναζήτηση των λύσεων. Αυτό θα κάνει τη μέθοδο που θα παρουσιάσουμε ειδική για δυναμικά που είναι άρτιες συναρτήσεις της θέσης. Στην επόμενη παράγραφο θα αναπτύξουμε μία πιο γενική μέθοδο που θα περιλαμβάνει και μη άρτια δυναμικά. Οι λύσεις χωρίζονται σε δύο κατηγορίες, μία με άρτια ομοτιμία  $\psi_n(x) \equiv \psi_n^{(+)}(-x) = \psi_n^{(+)}(x)$  για  $n = 1, 3, 5, 7, \dots$  και μία με περιττή ομοτιμία  $\psi_n(x) \equiv -\psi_n^{(-)}(-x) = \psi_n^{(-)}(x)$  για  $n = 2, 4, 6, 8, \dots$

$$\psi_n(x) = \begin{cases} \psi_n^{(+)}(x) = \cos\left(\frac{n\pi}{2}x\right) & |x| < 1 \quad n = 1, 3, 5, 7, \dots \\ \psi_n^{(-)}(x) = \sin\left(\frac{n\pi}{2}x\right) & |x| < 1 \quad n = 2, 4, 6, 8, \dots \end{cases} \quad (10.19)$$

όπου

$$\epsilon_n = \left(\frac{n\pi}{2}\right)^2, \quad (10.20)$$

και η κανονικοποίηση έχει επιλεγεί, έτσι ώστε<sup>3</sup>  $\int_{-1}^1 (\psi_n(x))^2 dx = 1$ .

Οι λύσεις που αναζητάμε είναι δυνατόν να βρεθούνε χρησιμοποιώντας τις ιδιότητες ομοτιμίας των κυματοσυναρτήσεων. Παρατηρούμε ότι για τις λύσεις θετικής ομοτιμίας

$$\psi_n^{(+)}(0) = A \quad \psi_n^{(+)\prime}(0) = 0, \quad (10.21)$$

ενώ για τις λύσεις αρνητικής ομοτιμίας

$$\psi_n^{(-)}(0) = 0 \quad \psi_n^{(-)\prime}(0) = A. \quad (10.22)$$

Η σταθερά  $A$  εξαρτάται από την κανονικοποίηση της κυματοσυνάρτησης. Άρα, μπορούμε να θέσουμε  $A = 1$  και στη συνέχεια, να επανακανονικοποιήσουμε την κυματοσυνάρτηση, έτσι ώστε να ισχύει η (10.2). Οι σχέσεις (10.21) και (10.22) μπορούν να θεωρηθούν ως οι αρχικές συνθήκες στην (10.12), αν η ενέργεια είναι γνωστή, οπότε με έναν αλγόριθμο της αρεσκείας μας (λ.χ. Runge–Kutta 4ης τάξης) να προωθήσουμε τη λύση προς τα  $x = \pm 1$ . Φυσικά, το πρόβλημα είναι ότι η ενέργεια  $\epsilon$  δεν είναι γνωστή. Αν η ενέργεια δεν είναι η επιτρεπτή από την κβαντική θεωρία, τότε θα βρούμε ότι παραβιάζονται οι συνοριακές συνθήκες

$$\psi_n^{(\pm)}(\pm 1) = 0. \quad (10.23)$$

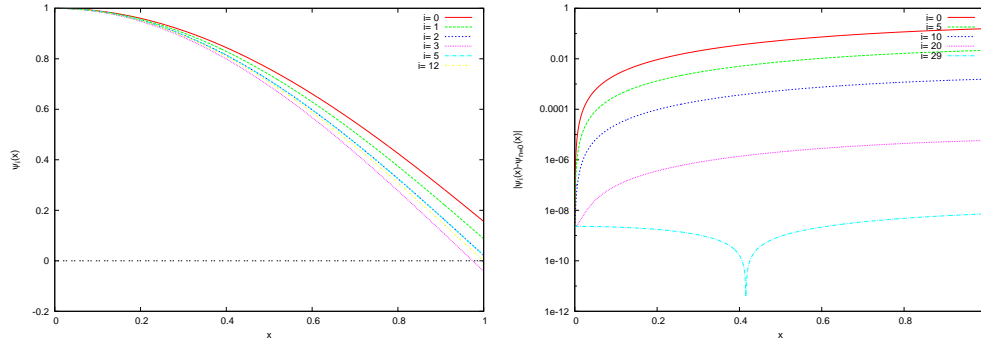
Όσο πλησιάζουμε τη σωστή ενέργεια, τόσο  $\psi_n^{(\pm)}(\pm 1) \rightarrow 0$ .

Οπότε ακολουθούμε την παρακάτω διαδικασία:

- Επιλέγουμε ενέργεια  $\epsilon$  η οποία είναι χαμηλότερη από τη ζητούμενη. Αν το δυναμικό δεν είναι απλό τετραγωνικό, χρησιμοποιούμε τις λύσεις ανάλογου τετραγωνικού προβλήματος για την εκτίμηση τάξης μεγέθους ή ξεκινάμε από ενέργεια λίγο μεγαλύτερη από την ελάχιστη δυναμική ενέργεια.
- Επιλέγουμε την ομοτιμία της ζητούμενης λύσης και θέτουμε αρχικές συνθήκες σύμφωνα με τις (10.21) και (10.22).
- Χρησιμοποιούμε τη μέθοδο Runge–Kutta για να προωθήσουμε τη λύση από<sup>4</sup>  $x = 0$  σε  $x = +1$ .

<sup>3</sup>Σύμφωνα με το “λεξικό” που αναφέραμε στο τέλος της προηγούμενης παραγράφου, για πηγάδι δυναμικού όπου  $x \in [-L/2, L/2]$  έχουμε επιλέξει ως αδιάστατη μεταβλητή την  $x/(L/2) \in [-1, 1]$ . Τότε  $E_n = \frac{\hbar^2}{2m(L/2)^2} \epsilon_n = \frac{\hbar^2 \pi^2}{2mL^2} n^2$  και  $\psi_n^{(+)}(x) = \sqrt{2/L} \cos(n\pi x/L)$ ,  $\psi_n^{(-)}(x) = \sqrt{2/L} \sin(n\pi x/L)$ . Επίσης, παρατηρείστε πως  $\epsilon_n = p_n^2$  όπως προκύπτει από τις σχέσεις (10.13) και (10.14).

<sup>4</sup>Από την ομοτιμία της συνάρτησης, συμπεραίνουμε την τιμή της συνάρτησης στο διάστημα  $[-1, 0)$ .



Σχήμα 10.2: Η διαδικασία σύγκλισης της λύσης  $\psi_i(x)$  της (10.12) με το δυναμικό (10.17) σαν συνάρτηση του αριθμού επαναλήψεων  $i$  στο πρόγραμμα `well.f90`. Αρχικά επιλέγουμε  $\text{energy} = 2.0$  και θετική ομοτιμία  $\text{parity} = 1$ . Μετά από 29 επαναλήψεις, η λύση συγκλίνει στην θεμελιώδη κατάσταση  $\psi_1(x) = \cos(\pi x/2)$  με ενέργεια  $\epsilon = (\pi/2)^2$  με σχετική ακρίβεια  $\sim 10^{-9}$ . Στο κάτω σχήμα βλέπουμε (σε λογαριθμική κλίμακα) το σχετικό σφάλμα και τη μείωση του με τον αριθμό των επαναλήψεων. Η ενέργεια σε κάθε επανάληψη είναι για  $i \equiv \text{iter} = 1, 2, 3, 5, 10, 12, 20$   $\text{energy} = 2.4, 2.6, 2.4, 2.4625, 2.46875, 2.4673828125$ .

- Αν δεν εκπληρώνεται η (10.23), αυξάνουμε την ενέργεια κατά  $\delta\epsilon$  (επιλεγμένη παράμετρος) και επαναλαμβάνουμε.
- Επαναλαμβάνουμε μέχρι η  $\psi_n^{(\pm)}(1)$  να αλλάξει πρόσημο. Τότε, μειώνουμε την ενέργεια κατά  $\delta\epsilon = -\delta\epsilon/2$ .
- Η διαδικασία τερματίζεται, όταν  $|\psi_n^{(\pm)}(1)| < \delta$  για κατάλληλα επιλεγμένο  $\delta$ .

Για την οργάνωση του κώδικα, αναφέρουμε πρώτα την εξέλιξη της κυματοσυνάρτησης από  $x = 0$  σε  $x = 1$  με χρήση της μεθόδου Runge-Kutta 4ης τάξης. Για τα βήματα ολοκλήρωσης χρησιμοποιούμε τον κώδικα από το Κεφάλαιο 4 που βρίσκεται στο αρχείο `rk.f90`. Απομονώνουμε μόνο την subroutine `RKSTEP`, που θα βρείτε στο πρόγραμμα της σελίδας 226, και την αποθηκεύουμε σε ένα αρχείο `rk.f90`. Για την ολοκλήρωση της (10.12) χρησιμοποιούμε τη συνάρτηση  $\phi(x) \equiv \psi'(x)$  και παίρνουμε:

$$\begin{aligned}\psi'(x) &= \phi(x) \\ \phi'(x) &= (v(x) - \epsilon)\psi(x),\end{aligned}\tag{10.24}$$

μαζί με τις αρχικές συνθήκες

$$\begin{aligned}\psi(0) = 1 \quad , \quad \phi(0) \equiv \psi'(0) = 0 & \quad \text{άρτια ομοτιμία} \\ \psi(0) = 0 \quad , \quad \phi(0) \equiv \psi'(0) = 1 & \quad \text{περιττή ομοτιμία}.\end{aligned}\tag{10.25}$$

Χρησιμοποιούμε το συμβολισμό  $\psi(x) \rightarrow \text{psi}$ ,  $\phi(x) \rightarrow \text{psip}$ . Οι συναρτήσεις  $f1$  και  $f2$  που αντιστοιχούν στο δεξί μέλος της (10.24), είναι οι παράγωγοι των  $\psi(x)$  και  $\phi(x)$  αντίστοιχα. Άρα απλά  $f1=\text{psip}$  και  $f2=(V-\text{energy})*\text{psi}$ . Ο προγραμματισμός τους γίνεται σε ξεχωριστό αρχείο, έτσι ώστε να είναι εύκολη η μελέτη και άλλων δυναμικών  $v(x)$ . Στο αρχείο `wellInfSq.f90` προγραμματίζουμε το δυναμικό (10.17):

```
!=====
!file: wellInfSq.f
!
!Functions used in RKSTEP routine. Here:
!f1 = psip(x) = psi(x)'
!f2 = psip(x)'' = psi(x)''
!
!All one has to set is V, the potential
!=====
!----- trivial function: derivative of psi
real(8) function f1(x,psi,psip)
  real(8) :: x,psi,psip
  f1=psip
end function f1
!=====
!----- the second derivative of wavefunction:
!psip(x)' = psi(x)'' = -(E-V) psi(x)
real(8) function f2(x,psi,psip)
  implicit none
  real(8) :: x,psi,psip,energy,V
  common /params/energy
!----- potential, set here:
  V = 0.0D0
!----- Schroedinger eq: RHS
  f2 = (V-energy)*psi
end function f2
!=====
```

Τονίζουμε πως η ενέργεια  $\epsilon = \text{energy}$  τοποθετείται σε common block, ώστε να μπορούμε να τη μεταβάλλουμε από το κυρίως πρόγραμμα.

Το κυρίως πρόγραμμα βρίσκεται στο αρχείο `well.f90`. Αφού ζητήσουμε τα δεδομένα από το χρήστη (`energy`, `parity`, `Nx`) ξεκινάει η αναζήτηση της σωστής ενέργειας ξεκινώντας από την αρχικά επιλεγμένη τιμή

```
do while (iter .lt. 10000)
  .....
  if(DABS(psinew) .le. epsilon) EXIT
```

```

    if(psinew*psiold .lt. 0.0D0 ) de = -0.5D0*de
    energy = energy + de
    .....
enddo                                ! do while

```

η οποία σταματάει όταν  $\psi(1) = \text{psinew}$  έχει απόλυτη τιμή μικρότερη από  $\epsilon$ , δηλ. όταν θεωρήσουμε ότι η συνθήκη (10.23) ικανοποιείται με την επιθυμητή ακρίβεια. Αν το πρόσημο της κυματοσυνάρτησης στο  $x = 1$  αλλάξει ( $\text{psinew} * \text{psiold} < 0$ ), τότε έχουμε ξεπεράσει τη σωστή τιμή της ενέργειας, οπότε αλλάζουμε το πρόσημο του βήματος  $de$  και μειώνουμε το μέγεθός του στο μισό. Μέσα στο βρόχο εκτελείται ο αλγόριθμος που περιγράφεται στη σελίδα 428. Μετά την έξοδο από τον παραπάνω βρόχο, η ενέργεια έχει προσδιοριστεί με τη ζητούμενη ακρίβεια και το υπόλοιπο πρόγραμμα απλά καταγράφει τη λύση στο array `psifinal(STEPS)`. Τα αποτελέσματα καταγράφονται στο αρχείο `psi.dat`. Παρατηρήστε τη χρήση της μεταβλητής `parity` για την ενοποιημένη αντιμετώπιση των περιπτώσεων  $\text{parity} = \pm 1$ . Ολόκληρος ο κώδικας παρατίθεται παρακάτω:

```

!=====
! file: well.f
!
! Computation of energy eigenvalues and eigenfunctions
! of a particle in an infinite well with  $V(-x)=V(x)$ 
!
! Input:  energy: initial guess for energy
!         parity: desired parity of solution (+/- 1)
!         Nx-1 : Number of RK4 steps from x=0 to x=1
! Output: energy: energy eigenvalue
!         psi.dat: final psi(x)
!         all.dat: all psi(x) for trial energies
!=====
program even_potential_well
  implicit none
  integer, parameter :: P=10000
  real(8) :: energy, dx, x, epsilon, de
  common /params/energy
  integer :: parity, Nx, iter, i
  real(8) :: psi, psip, psinew, psiold
  real(8) :: psifinal(-P:P), xstep(-P:P)
!----- Input:
  print *, 'Enter energy, parity, Nx: '
  read *, energy, parity, Nx
  if(Nx .gt. P) stop 'Nx > P'
  if(parity .gt. 0) then
    parity = 1

```

```

else
    parity = -1
endif
print *,'# #####'
print *,'# Estart= ',energy,' parity= ',parity
dx      = 1.0D0/(Nx-1)
epsilon = 1.0D-6
print *,'# Nx= ',Nx,' dx = ',dx,' eps= ',epsilon
print *,'# #####'

!----- Calculate:
open(unit=11,file='all.dat')
iter     = 0
psiold   = 0.0D0 ! calculated values of psi at x=1
psinew   = 1.0D0
de       = 0.1D0*DABS(energy) ! original change in energy
do while (iter .lt. 10000)
!----- Initial conditions at x=0
    x      = 0.0D0
    if(parity .eq. 1)then
        psi   = 1.0D0
        psip  = 0.0D0
    else
        psi   = 0.0D0
        psip  = 1.0D0
    endif
    write(11,*) iter,energy, x, psi,psip
! ----- Use Runge-Kutta to forward to x=1
    do i=2,Nx
        x      = (i-2)*dx
        call RKSTEP(x,psi,psip,dx)
        write(11,*) iter,energy,x,psi,psip
    enddo
    psinew = psi
    print *,iter, energy, de,psinew
! ----- Stop if value of psi close to 0
    if(DABS(psinew) .le. epsilon) EXIT
! ----- Change direction of energy search:
    if(psinew*psiold .lt. 0.0D0 ) de = -0.5D0*de
    energy = energy + de
    psiold = psinew
    iter   = iter + 1
enddo
close(11)
!We found the solution: calculate it once again and store it
if(parity .eq. 1)then
    psi   = 1.0D0
    psip  = 0.0D0
    node  = 0 ! count number of nodes of function

```



```

else
  psi      = 0.0D0
  psip     = 1.0D0
  node     = 1
endif
x          = 0.0D0
xstep     (0) = x
psifinal(0) = psi ! array that stores psi(x)
psiold     = 0.0D0
!----- Use Runge-Kutta to move to x=1
do i=2,Nx
  x          = (i-2)*dx
  call RKSTEP(x,psi,psip,dx)
  xstep     (i-1) = x
  psifinal(i-1) = psi
!----- Use parity to compute psi(-x)
  xstep     (1-i) = -x
  psifinal(1-i) = parity*psi
!----- Print final solution:
  open(unit=11,file='psi.dat')
  print *, 'Final result: E= ',energy, ' n= ',node,&
    ' parity= ',parity
  write(11,*) '# E= ',energy, ' n= ',node,&
    ' parity= ',parity
  do i=-(Nx-1),(Nx-1)
    write(11,*) xstep(i),psifinal(i)
  enddo
  close(11)
end program even_potential_well

```

Η μεταγλώττιση και το τρέξιμο γίνεται εύκολα με τις εντολές

```

> gfortran well.f90 wellInfSq.f90 rk.f90 -o well
> ./well
Enter energy,parity,Nx:
2.0 1 400
# #####
# Estart= 2.0000000000000000 parity= 1
# Nx= 400 dx = 2.50626566416E-003 eps= 9.99999999999E-007
# #####
0 2.0000000000000000 0.200000000000 0.15594369476721
1 2.2000000000000000 0.200000000000 8.74448016806986E-2
.....
28 2.4674072265624 1.220703125000E-5 -1.95005436858826E-6
29 2.4674011230468 -6.103515625000E-6 -7.24621589476086E-9
Final result: E= 2.4674011230468746 parity= 1

```

Η τιμή της ενέργειας προσδιορίζεται σε  $\epsilon = 2.467401123$  που μπορεί να

$n$	$(n\pi/2)^2$	Τετραγωνικό	Τριγωνικό	Διπλό Πηγάδι
1	2.467401100	2.467401123	5.248626709	15.294378662
2	9.869604401	9.869604492	14.760107422	15.350024414
3	22.2066099	22.2066040	27.0690216	59.1908203
4	39.47841	39.47839	44.51092	59.96887
5	61.6850275	61.6850242	66.6384315	111.3247375
6	88.82643	88.82661	93.84588	126.37628
7	120.902653	120.902664	125.878830	150.745215
8	157.91367	157.91382	162.92569	194.07578
9	199.859489	199.859490	204.845026	235.017471
10	246.74011	246.74060	251.74813	275.67383
11	298.555533	298.555554	303.545814	331.428306
12	355.3057	355.3064	360.3107	388.7444

Πίνακας 10.1: Οι ιδιοτιμές της ενέργειας για το τετραγωνικό, τριγωνικό και διπλό πηγάδι απειρόβαθου δυναμικού (Εξισώσεις (10.17), (10.26) με  $v_0 = 10$  και (10.27) με  $v_0 = 100$ ,  $a = 0.3$ ). Παρατηρούμε την εξαιρετική ακρίβεια προσδιορισμού των ενεργειακών σταθμών για το τετραγωνικό δυναμικό. Επίσης, για τα άλλα δυναμικά, παρατηρούμε ότι καθώς απομακρυνόμαστε από τον πάτο του δυναμικού, οι ενεργειακές στάθμες τείνουν να γίνουν οι ίδιες: Το σωματίο παύει να επηρεάζεται από τις λεπτομέρειες του δυναμικού στον πάτο, καθώς αυξάνεται η ενέργειά του. Στο διπλό πηγάδι δυναμικού παίρνουμε  $E_1 \approx E_2$  και  $E_3 \approx E_4$  σύμφωνα με την ανάλυση στη σελίδα 435.

συγκριθεί με την ακριβή τιμή  $\epsilon = (\pi/2)^2 \approx 2.467401100$ . Το ποσοστό σφάλματος είναι  $\sim 10^{-8}$ , άρα μάλλον καλά πήγαμε! Η διαδικασία της σύγκλισης φαίνεται στο σχήμα 10.2.

Για την εύρεση των διεγερμένων καταστάσεων αλλάζουμε την ομοτιμία και διαλέγουμε κάθε φορά ενέργεια ελαφρά μεγαλύτερη από τη λύση που έχουμε ήδη βρει<sup>5</sup>. Τα αποτελέσματα δίνονται στον πίνακα 10.1. Παρατηρούμε την εξαιρετική συμφωνία του αριθμητικού υπολογισμού συγκρινόμενου με το αναλυτικά γνωστό αποτέλεσμα  $\epsilon_n = (n\pi/2)^2$ .

Τελειώνουμε την παράγραφο με δύο ακόμα παραδείγματα δυναμικών. Πρώτα του δυναμικού με τριγωνικό σχήμα στον πάτο

$$v(x) = \begin{cases} v_0|x| & |x| < 1 \\ +\infty & |x| > 1 \end{cases} \quad (10.26)$$

<sup>5</sup>Προσοχή: αν τα επίπεδα ενέργειας είναι πολύ κοντά, μπορούμε να κρατάμε την αρχική ενέργεια σταθερή και να αλλάζουμε μόνο την ομοτιμία.

καθώς και ένα διπλό πηγάδι δυναμικού με

$$v(x) = \begin{cases} v_0 & |x| < a \\ 0 & a < |x| < 1 \\ +\infty & 1 < |x| \end{cases} \quad (10.27)$$

όπου οι παράμετροι  $v_0, a$  είναι θετικοί αριθμοί. Η μορφή των δυναμικών δείχνεται πρόχειρα στο σχήμα 10.1.

Για το τριγωνικό δυναμικό επιλέγουμε  $v_0 = 10$ , ενώ για το διπλό πηγάδι  $v_0 = 100$  και  $a = 0.3$ . Οι μεταβολές στον κώδικα γίνονται στο αρχείο `wellInfSq.f90` και αποθηκεύονται μέσα στα αρχεία `wellInfTr.f90` και `wellInfDbl.f90` αντίστοιχα. Αλλάζουμε μόνο τη γραμμή του κώδικα που αφορά τη συνάρτηση του δυναμικού μέσα στη συνάρτηση `f2`. Λ.χ. στο αρχείο `wellInfTr.f90`

```
!----- potential , set here:
V = 10.0D0*DABS(x)
```

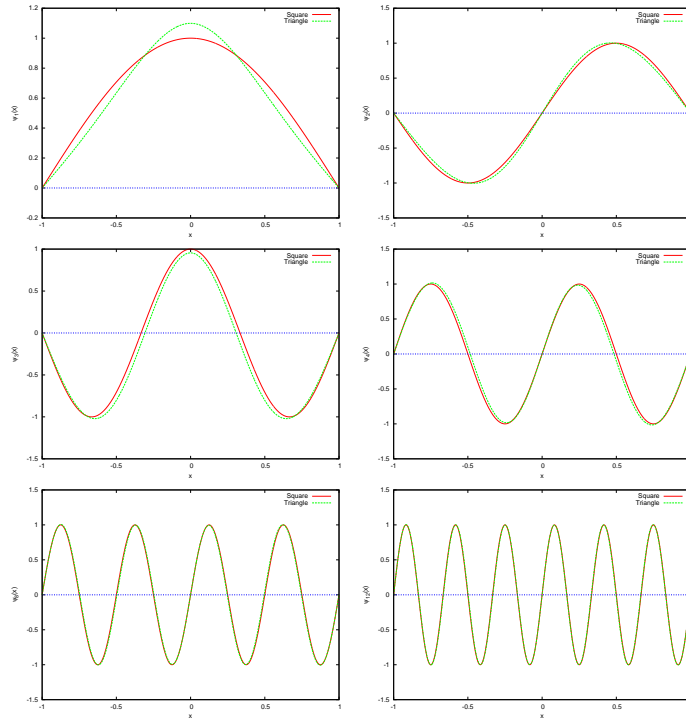
ενώ στο αρχείο `wellInfDbl.f90`

```
!----- potential , set here:
if( DABS(x) .le. 0.3D0)then
  V = 100.0D0
else
  V = 0.0D0
endif
```

Η ανάλυση γίνεται με τον ίδιο ακριβώς τρόπο και τα αποτελέσματα για το ενεργειακό φάσμα δίνονται στον πίνακα 10.1. Παρατηρούμε ότι οι υψηλές ενεργειακές στάθμες των τριών δυναμικών τείνουν να γίνουν οι ίδιες, καθώς το  $n$  γίνεται πολύ μεγάλο. Ο λόγος είναι ότι όταν το σωματίο έχει πολύ υψηλή ενέργεια σε σχέση με το  $v_0$ , επηρεάζεται πολύ λίγο από τις λεπτομέρειες της μορφής του δυναμικού στον πάτο και ουσιαστικά βλέπει ένα απειρόβαθο πηγάδι δυναμικού. Στην περίπτωση του τριγωνικού δυναμικού, οι πρώτες ενεργειακές στάθμες είναι υψηλότερες από αυτές του τετραγωνικού δυναμικού, αφού κατά μέσο όρο η δυναμική ενέργεια είναι μεγαλύτερη και η μορφή του δυναμικού τείνει να περιορίσει το σωματίο σε μικρότερη περιοχή ( $\Delta x$  μειώνεται, άρα  $\Delta p$  αυξάνεται). Το τελευταίο φαίνεται και στο σχήμα 10.3, όπου συγκρίνονται οι κυματοσυναρτήσεις των δύο δυναμικών.

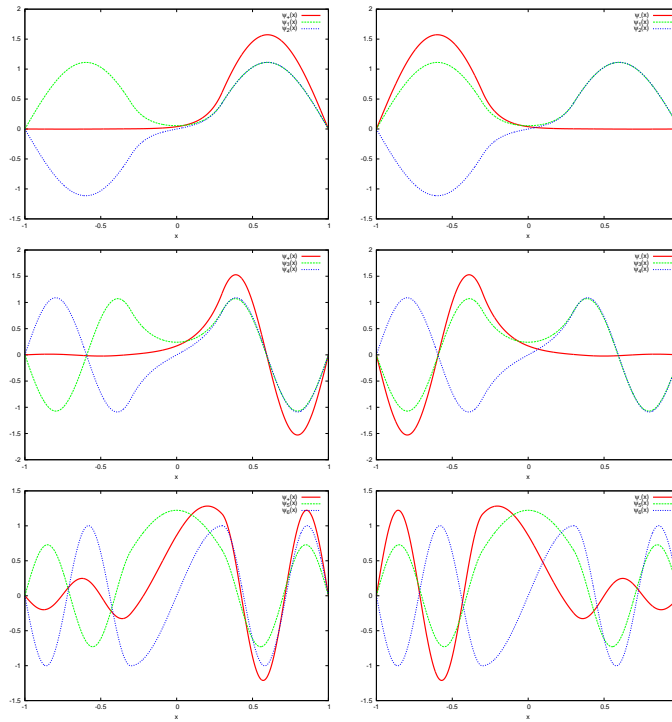
Το ίδιο παρατηρούμε και για το διπλό πηγάδι δυναμικού. Επιπλέον, βλέπουμε και τον προσεγγιστικό εκφυλισμό των 4 πρώτων ενεργειακών σταθμών ανά ζεύγη, κάτι που αναμένεται σε δυναμικά της μορφής

αυτής. Αυτό σας θυμίζω μπορεί να κατανοηθεί ποιοτικά παρατηρώντας λ.χ. ότι οι κυματοσυναρτήσεις  $\psi_+(x) = (1/\sqrt{2})(\psi_1(x) + \psi_2(x))$  και  $\psi_-(x) = (1/\sqrt{2})(\psi_1(x) - \psi_2(x))$  αντιστοιχούν σε κυματοσυναρτήσεις καταστάσεων στις οποίες το σωματίο σχεδόν περιορίζεται στο αριστερό και δεξί πηγάδι αντίστοιχα. Αυτό μπορείτε να το δείτε στο σχήμα 10.4. Καθώς  $v_0 \rightarrow +\infty$ , τα δύο πηγάδια αποσυζεύγγνυνται και οι  $\psi_{\pm}(x)$  τείνουν προς τις κυματοσυναρτήσεις θεμελιώδους κατάστασης δύο ανεξάρτητων απειρόβαθων πηγαδιών πλάτους  $1 - a$  και οι αντίστοιχες ενέργειες σε  $\epsilon_{+,1} = \epsilon_{-,1} = (\pi/(1-a))^2$ . Η διαφορά των  $\epsilon_1$  και  $\epsilon_2$  από τις τιμές αυτές οφείλεται στην πεπερασμένη τιμή του δυναμικού  $v_0$  (δείτε άσκηση 4).



Σχήμα 10.3: Οι κυματοσυναρτήσεις των ενεργειακών ιδιοκαταστάσεων για  $n = 1, 2, 3, 4, 8, 12$  για το απειρόβαθο τετραγωνικό και τριγωνικό δυναμικό των εξισώσεων (10.17) και (10.26) με  $v_0 = 10$ . Παρατηρούμε την επίδραση του τριγωνικού δυναμικού στις κυματοσυναρτήσεις μικρού  $n$ , ενώ για  $n \geq 8$  η διάκριση γίνεται ολοένα και μικρότερη.

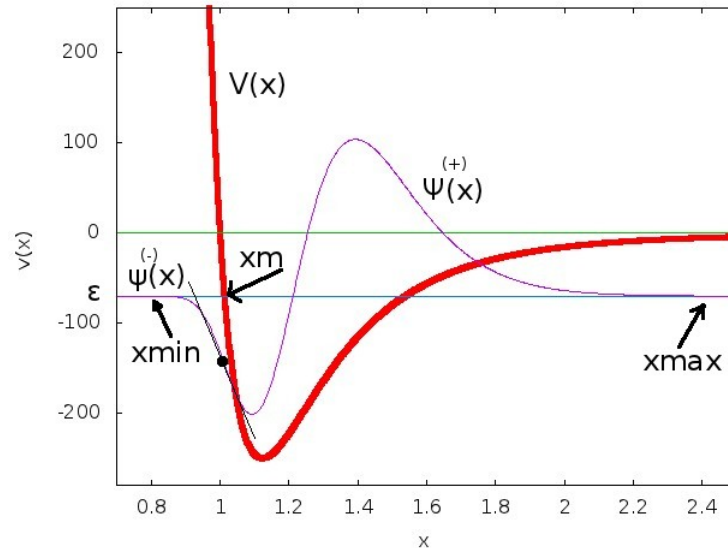
Τέλος, να τονίσουμε τους περιορισμούς της μεθόδου αυτής. Καταρχήν, θυμίζουμε ότι μπορούμε να την χρησιμοποιήσουμε μόνο σε απειρόβαθα πηγάδια δυναμικού που είναι άρτια  $v(x) = v(-x)$ . Αυτό χρησιμοποιήθηκε στις αρχικές συνθήκες (10.21) και (10.22) που ισχύουν για καταστάσεις δεδομένης ομοτιμίας. Όταν το δυναμικό είναι άρτιο,



Σχήμα 10.4: Οι συναρτήσεις  $\psi_{\pm}(x) = (1/\sqrt{2})(\psi_n(x) \pm \psi_{n+1}(x))$  για  $n = 1, 3, 5$  του διπλού πηγαδιού δυναμικού (Εξ. (10.27) με  $v_0 = 100, a = 0.3$ ) φαίνονται με έντονο χρώμα. Παρατηρούμε ότι στις καταστάσεις αυτές όσο πιο έντονος είναι ο προσεγγιστικός εκφυλισμός, τόσο εντονότερος είναι και ο εντοπισμός του σωματιδίου στο δεξί ή αριστερό μέρος του διπλού πηγαδιού. Με αχνότερο χρώμα δείχνονται οι κυματοσυναρτήσεις των ιδιοκαταστάσεων της ενέργειας για  $n = 1, 2, 3, 4, 5, 6$ .

οι ιδιοκαταστάσεις της ενέργειας έχουν καλά καθορισμένη ομοτιμία. Το άλλο πρόβλημα γίνεται αντιληπτό, αν κάνετε την άσκηση 4: Αν η κυματοσυνάρτηση είναι σχεδόν μηδέν για  $x = 0$ , όπως συμβαίνει όταν  $v(0) \gg \epsilon$ , τότε τα αριθμητικά σφάλματά μας εμποδίζουν να ολοκληρώσουμε με μεγάλη ακρίβεια από  $x = 0$  σε  $x = 1$ . Το ίδιο θα συμβαίνει και όταν έχουμε να περάσουμε μέσα από ψηλά φράγματα δυναμικού. Παρόλα αυτά είναι μια απλή μέθοδος που μπορεί να χρησιμοποιηθεί και στην περίπτωση που έχουμε δέσμιες καταστάσεις σε ένα άρτιο δυναμικό που δεν είναι απειρόβαθο: Στην περίπτωση αυτή παίρνουμε το δυναμικό που μας δίνεται και απλά τοποθετούμε τους αδιαπέραστους τοίχους σε σημεία όπου η κυματοσυνάρτηση αναμένεται να είναι στην περιοχή τους πρακτικά παντού μηδέν. Τότε η επίδραση του τείχους θα είναι πολύ μικρή πάνω στις λύσεις του αρχικού προβλήματος. Δείτε το πρόβλημα 3.

### 10.3 Δέσμιες Καταστάσεις



Σχήμα 10.5: Ολοκλήρωση της εξίσωσης Schrödinger σύμφωνα με τον αλγόριθμο της παραγράφου 10.3. Οι κυματοσυναρτήσεις και οι παράγωγοί τους ορίζονται να έχουν μικρές τιμές στις κλασικά απαγορευμένες τιμές του  $x$   $x_{\min}$  και  $x_{\max}$ . Το σημείο  $x_m$  υπολογίζεται από τη σχέση  $v(x_m) = \epsilon$ . Οι κυματοσυναρτήσεις προωθούνται μέχρι το  $x_m$  σύμφωνα με τις (10.24) και παίρνουμε τις  $\psi^{(+)}(x)$  και  $\psi^{(-)}(x)$ . Αφού ορίσουμε  $\psi^{(+)}(x_m) = \psi^{(-)}(x_m)$  επανακανονικοποιώντας την  $\psi^{(-)}(x)$ , μεταβάλλουμε την ενέργεια μέχρι οι παράγωγοι  $\psi^{(+)}'(x_m) \approx \psi^{(-)}'(x_m)$ .

Το σπουδαιότερο πρόβλημα με την απλή μέθοδο που παρουσιάσαμε στην παράγραφο 10.2 είναι ότι παρουσιάζει αριθμητική αστάθεια, όταν προσπαθούμε να λύσουμε ένα πρόβλημα δέσμιων καταστάσεων. Αυτό θα το συναντήσατε ήδη αν προσπαθήσατε να λύσετε την άσκηση 3 όπου μετακινώντας τα τείχη μακρύτερα από  $|x| = 3$  η σύγκλιση του αλγόριθμου γίνεται ιδιαίτερα δυσχερής. Για να το καταλάβουμε αυτό, όταν ολοκληρώνουμε την εξίσωση Schrödinger από την ελεύθερη περιοχή προς την κλασικά απαγορευμένη, περνάμε από ταλαντούμενη κυματοσυνάρτηση μέτρου της τάξης της μονάδας σε κυματοσυνάρτηση η οποία έχει εκθετική απόσβεση. Δεν πρέπει όμως να ξεχνάμε ότι για  $|x| \rightarrow +\infty$ , μαζί με τη φυσικά αποδεκτή λύση  $\psi(x) \sim e^{-k|x|}$  έχουμε και τη λύση  $\psi(x) \sim e^{+k|x|}$  η οποία αποκλίνει εκθετικά γρήγορα και την απορρίπτουμε λόγω της (10.2). Έτσι, χρειάζεται πολύ λεπτή ρύθμιση της τιμής της ενέργειας για την επίτευξη σύγκλισης, ειδικά αν ολοκληρώνουμε για μεγάλα  $|x|$ . Για το λόγο αυτό, είναι προτιμότερο να ολο-

κληρώνουμε από την περιοχή εκθετικής απόσβεσης προς την κλασικά επιτρεπόμενη περιοχή. Η ιδέα είναι να ξεκινήσουμε από τέτοιες περιοχές και να ταιριάζουμε τις τιμές των λύσεων και των παραγώγων τους σε κατάλληλα επιλεγμένα σημεία. Το ταίριασμα γίνεται προσπαθώντας να εντοπίσουμε την ενέργεια που η τιμή της κάνει το λόγο

$$f(\epsilon) = \frac{\psi^{(+)\prime}(x_m)/\psi^{(+)}(x_m) - \psi^{(-)\prime}(x_m)/\psi^{(-)}(x_m)}{\psi^{(+)\prime}(x_m)/\psi^{(+)}(x_m) + \psi^{(-)\prime}(x_m)/\psi^{(-)}(x_m)} \quad (10.28)$$

να είναι μηδέν μέσα στα όρια της αριθμητικής ακρίβειας που θέτουμε σε κατάλληλα επιλεγμένο σημείο  $x_m$ . Το σημείο  $x_m$  είναι καλό να βρίσκεται μέσα στην κλασικά επιτρεπόμενη περιοχή ( $\epsilon > v(x)$ ) και, συνήθως, το παίρνουμε στο σημείο όπου  $\epsilon = v(x)$ . Με κατάλληλη επανακανονικοποίηση των  $\psi^{(\pm)}(x)$ , επιλέγουμε  $\psi^{(+)}(x_m) = \psi^{(-)}(x_m)$ , οπότε  $f(\epsilon) \ll 1$  σημαίνει ότι  $\psi^{(+)\prime}(x_m) \approx \psi^{(-)\prime}(x_m)$ . Ο παρονομαστής της (10.28) απλά θέτει την κλίμακα στην ακρίβεια που θέλουμε να πετύχουμε<sup>6</sup>.

Η ιδέα σκιαγραφείται γραφικά στο σχήμα 10.5. Ο αλγόριθμος έχει ως εξής:

- Επιλέγουμε το διάστημα ολοκλήρωσης  $[x_{\min}, x_{\max}]$ .
- Επιλέγουμε τις αρχικές συνθήκες ολοκλήρωσης  $\psi^{(-)}(x_{\min})$ ,  $\psi^{(-)\prime}(x_{\min})$ ,  $\psi^{(+)}(x_{\max})$ ,  $\psi^{(+)\prime}(x_{\max})$ . Η επιλογή γίνεται ανάλογα με το πρόβλημα, δηλ. με το δυναμικό  $v(x)$  που θα μελετήσουμε. Συνήθως τα  $x_{\min}$ ,  $x_{\max}$  είναι αρκετά βαθιά μέσα στην κλασικά απαγορευμένη περιοχή και παίρνουμε τα  $\psi^{(-)}(x_{\min})$ ,  $\psi^{(+)}(x_{\max})$  μηδέν ή εκθετικά μικρά ( $\sim e^{-k|x|}$ ,  $k^2 = v(x) - \epsilon$ ). Οι παράγωγοι  $\psi^{(-)\prime}(x_{\min})$ ,  $\psi^{(+)\prime}(x_{\max})$  παίρνονται ανάλογα μικρές. Η ελευθερία επιλογής της σταθεράς κανονικοποίησης της  $\psi(x)$  μας επιτρέπει οι επιλογές αυτές να είναι σχετικά αυθαίρετες. Το σχετικό πρόσημο των παραγώγων (που καθορίζεται λ.χ. από την ομοτιμία σε περίπτωση συμμετρικού δυναμικού) δεν έχει αρχικά σημασία και λαμβάνεται υπόψη στην επανακανονικοποίηση της  $\psi^{(-)}(x)$ , όταν παρακάτω θα κάνουμε την ταύτιση των τιμών στο σημείο  $x_m$ . Σε ένα απειρόβαθο πηγάδι, τα  $x_{\min}, x_{\max}$  είναι τα σημεία απειρισμού του δυναμικού και φυσικά  $\psi^{(-)}(x_{\min}) = \psi^{(+)}(x_{\max}) = 0$ .
- Επιλέγουμε αρχική ενέργεια  $\epsilon$  και βήμα μεταβολής της  $\delta\epsilon$ .

<sup>6</sup>Προσοχή: αν έχουμε την ατυχία να επιλέξουμε  $x_m$  τέτοιο ώστε  $\psi'(x_m) = 0$ , τότε το παραπάνω κριτήριο μάλλον θα αποτύχει.

- Από την τιμή της ενέργειας υπολογίζουμε το σημείο  $x_m$ , το οποίο βρίσκεται κοντά στο πιο αριστερό σημείο που λύνει την εξίσωση<sup>7</sup>  $v(x) = \epsilon$ .
- Προωθούμε τις εξισώσεις (10.24) από το  $x_{\min}$  στο  $x_m$  και παίρνουμε τις  $\psi^{(-)}(x), \psi^{(-)'}(x)$ .
- Προωθούμε τις εξισώσεις (10.24) από το  $x_{\max}$  στο  $x_m$  και παίρνουμε τις  $\psi^{(+)}(x), \psi^{(+)'}(x)$ .
- Επανακανονικοποιούμε την  $\psi^{(-)}(x) \rightarrow \psi^{(-)}(x)(\psi^{(+)}(x_m)/\psi^{(-)}(x_m))$ , έτσι ώστε να έχουμε  $\psi^{(+)}(x_m) = \psi^{(-)}(x_m)$ .
- Υπολογίζουμε το λόγο  $f(\epsilon)$  της (10.28).
- Αν η  $|f(\epsilon)|$  είναι μικρότερη από μια μικρή σταθερά ανοχής, σταματάμε και θεωρούμε τον υπολογισμό της ιδιοενέργειας και ιδιοκατάστασης ικανοποιητική.
- Αν η  $f(\epsilon)$  άλλαξε πρόσημο από την προηγούμενη επανάληψη, σημαίνει πως μόλις περάσαμε την αναζητούμενη τιμή της ιδιοενέργειας. Μεταβάλλουμε τη φορά αναζήτησης  $\delta\epsilon \rightarrow -\delta\epsilon/2$ .
- Μεταβάλλουμε την τιμή της ενέργειας  $\epsilon \rightarrow \epsilon + \delta\epsilon$  και επαναλαμβάνουμε από το τέταρτο βήμα.

Μετά από τον παραπάνω αλγόριθμο, η κυματοσυνάρτηση αποτελεί καλή προσέγγιση της ιδιοκατάστασης  $\psi_n(x)$  με ιδιοενέργεια  $\epsilon_n$ . Περαιτέρω επεξεργασία αφορά την κανονικοποίηση σύμφωνα με την (10.2) και ο υπολογισμός των αναμενόμενων τιμών σύμφωνα με την (10.9). Ενδιαφέρον παρουσιάζει και ο αριθμός  $n_0$  των σημείων μηδενισμού της κυματοσυνάρτησης που δίνει την τάξη του ενεργειακού επιπέδου  $n$  της ενέργειας  $\epsilon_n$  ( $n = n_0 + 1$ ).

Παρατίθεται το πρόγραμμα που κωδικοποιεί τον παραπάνω αλγόριθμο. Κατ' αρχήν, όπως και στην προηγούμενη παράγραφο, χρησιμοποιούμε τη ρουτίνα RKSTEP (βλ. σελίδα 226) που πραγματοποιεί ένα βήμα Runge-Kutta 4ης τάξης την οποία προγραμματίζουμε στο αρχείο rk.f90.

Το δυναμικό προγραμματίζεται σε μια συνάρτηση  $V(x)$ . Οι αρχικές συνθήκες στα σημεία  $x_{\min}$  και  $x_{\max}$  προγραμματίζονται στη ρουτίνα `boundary(xmin, xmax, psixmin, psipxmin, psixmax, psipxmax)` η οποία επιστρέφει στο κυρίως πρόγραμμα τις τιμές  $psixmin = \psi^{(-)}(x_{\min})$ ,

<sup>7</sup>Προσοχή, το σημείο αυτό αλλάζει όταν αλλάζουμε την  $\epsilon$ .



$\text{psipxmin} = \psi^{(-)'}(\text{xmin})$ ,  $\text{psixmax} = \psi^{(+)}(\text{xmax})$ ,  $\text{psipxmax} = \psi^{(-)'}(\text{xmax})$ . Αυτές τις τοποθετούμε σε ένα αρχείο του οποίου το όνομα έχει σχέση με το δυναμικό  $v(x)$ . Για παράδειγμα, για το απειρόβαθο πηγάδι δυναμικού (10.17) δημιουργούμε το αρχείο `schInfSq.f90`. Στο ίδιο αρχείο προγραμματίζουμε και τις συναρτήσεις των παραγώγων `f1`, `f2`, όπως κάναμε και στην προηγούμενη παράγραφο.

```
!=====
!file: schInfSq.f
!
!Functions used in RKSTEP routine. Here:
!f1 = psip(x) = psi(x)'
!f2 = psip(x)' = psi(x)''
!
!One has to set:
! 1. V(x), the potential
! 2. The boundary conditions for psi,psip at x=xmin and x=xmax
!
!=====
!----- potential:
real(8) function V(x)
  implicit none
  real(8) :: x
  V = 0.0D0
end function V
!----- boundary conditions:
subroutine boundary(xmin,xmax,&
                   psixmin,psipxmin,psixmax,psipxmax)
  implicit none
  real(8) :: xmin,xmax,psixmin,psipxmin,psixmax,psipxmax,V
!for infinite square well we set psi=0
!at boundary and psip=+/-1
  psixmin = 0.0D0
  psipxmin = 1.0D0
  psixmax = 0.0D0
  psipxmax = -1.0D0
!----- Initial values at xmin and xmax
end subroutine boundary
!=====
!----- trivial function: derivative of psi
real(8) function f1(x,psi,psip)
  real(8) :: x,psi,psip
  f1=psip
end function f1
!=====
!----- the second derivative of wavefunction:
```

```

!psip(x)' = psi(x)'' = -(E-V) psi(x)
real(8) function f2(x,psi,psip)
  implicit none
  real(8) :: x,psi,psip,energy,V
  common /params/energy
!----- Schroedinger eq: RHS
  f2 = (V(x)-energy)*psi
end function f2
!=====

```

Παρατηρούμε ότι τα σημεία απειρισμού του δυναμικού καθορίζονται από τις συνοριακές συνθήκες στα  $x_{\min}$ ,  $x_{\max}$ .

Το κυρίως πρόγραμμα παρουσιάζεται στο αρχείο `sch.f90`. Παρακάτω παραθέτουμε τον κώδικα ο οποίος συμπεριλαμβάνει κανονικοποίηση της κυματοσυνάρτησης και υπολογισμό του αριθμού των δεσμών της κυματοσυνάρτησης. Η κανονικοποίηση γίνεται από τη συνάρτηση `integrate(psi, dx, Nx)` η οποία ολοκληρώνει αριθμητικά με τη μέθοδο Simpson το τετράγωνο μιας συνάρτησης που οι τιμές της  $\psi(i)$   $i=1, \dots, Nx$  δίνονται σε ένα περιττό αριθμό από  $Nx$  σημεία τα οποία ισαπέχουν απόσταση  $dx$ .

```

!=====
!
! File: sch.f90
!
! Integrate 1d Schrodinger equation from xmin to xmax.
! Determine energy eigenvalue and eigenfunction by matching
! evolving solutions from xmin and from xmax at a point xm.
! Matching done by equating values of functions and their
! derivatives at xm. The point xm chosen at the left most
! turning point of the potential at any given value of the
! energy. The potential and boundary conditions chosen in
! different file.
!
! Input:  energy: Trial value of energy
!         de: energy step, if matching fails de -> e+de, if
!             logderivative changes sign      de -> -de/2
!         xmin, xmax, Nx
!
! Output: Final value of energy, number of nodes of
!         wavefunction in stdout
!         Final eigenfunction in file psi.dat
!         All trial functions and energies in file all.dat
!=====
program schroedinger_equation_1D
  implicit none

```

```

integer,parameter :: P=20001
integer :: Nx,NxL,NxR
real(8) :: psi(P),psip(P)
real(8) :: dx
real(8) :: xmin,xmax,xm      !left/right/matching points
real(8) :: psixmin,psipxmin,psixmax,psipxmax
real(8) :: psileft,psiright,psistep,psinorm
real(8) :: psipleft,psipright,psipstep
real(8) :: energy,de,epsilon,integrate
common/params/energy
real(8) :: matchlogd,matchold,psiold,norm,x
integer :: iter,i,imatch,nodes
real(8) :: v
!----- Input:
print *,'# Enter energy,de,xmin,xmax,Nx'
read *,energy,de,xmin,xmax,Nx
!----- need even intervals for normalization integration
if( mod(Nx,2).eq.0)Nx=Nx+1
if( Nx .gt. P ) stop 'Fatal Error: Nx>P'
if( xmin .ge. xmax) stop 'Error: xmin >= xmax'
dx      = (xmax - xmin)/(Nx-1)
epsilon = 1.0D-6
call boundary(xmin,xmax,psixmin,psipxmin,psixmax,psipxmax)
print *,'# #####'
print *,'# Estart= ',energy, ' de= ',de
print *,'# Nx= ',Nx, ' eps= ',epsilon
print *,'# xmin= ',xmin, ' xmax= ',xmax, ' dx= ',dx
print *,'# psi(xmin)= ',psixmin, ' psip(xmin)= ',psipxmin
print *,'# psi(xmax)= ',psixmax, ' psip(xmax)= ',psipxmax
print *,'# #####'
!----- Calculate:
open(unit=11,file='all.dat')
matchold = 0.0d0
do iter=1,10000
!----- Determine matching point at turning point from the left:
    imatch = -1
    do i=1,Nx
        x = xmin + (i-1)*dx
        if( imatch .lt. 0 .and. (energy-V(x)) .gt. 0.0D0) imatch = i
    enddo
    if( imatch .le. 100 .or. imatch .ge. Nx-100) imatch = Nx/5
    xm      = xmin + (imatch-1)*dx
    NxL = imatch
    NxR = Nx-imatch+1
!----- Evolve wavefunction from the left:
    psi (1) = psixmin
    psip (1) = psipxmin
    psistep = psixmin
    psipstep = psipxmin

```

```

do i=2,NxL
  x      = xmin + (i-2)*dx !this is x before the step
  call RKSTEP(x,psistep,psipstep, dx)
  psi (i) = psistep
  psip(i) = psipstep
enddo
! use this to normalize eigenfunction to match at xm
psinorm  = psistep
psipleft = psipstep
!----- Evolve wavefunction from the right:
psi (Nx) = psixmax
psip(Nx) = psipxmax
psistep  = psixmax
psipstep = psipxmax
do i=2,NxR
  x      = xmax - (i-2)*dx
  call RKSTEP(x,psistep,psipstep,-dx)
  psi (Nx-i+1) = psistep
  psip(Nx-i+1) = psipstep
enddo
psinorm      = psistep/psinorm
psipright    = psipstep
!----- Renormalize psil so that psil(xm)=psir(xm)
do i=1,NxL-1
  psi (i)      = psinorm * psi (i)
  psip(i)      = psinorm * psip(i)
enddo
psipleft      = psinorm * psipleft
!----- print current solution:
do i=1,Nx
  x = xmin + (i-1)*dx
  write(11,*)iter,energy,x,psi(i),psip(i)
enddo
!----- matching using derivatives:
!Careful: this can fail if psi'(xm) = 0 !! (use also |del|<1e-6
!criterion)
matchlogd = &
  (psipright-psipleft)/(DABS(psipright)+DABS(psipleft))
print *, '# iter,energy,de,xm,logd: ',&
  iter,energy,de,xm,matchlogd
!----- Exit condition:
if(DABS(matchlogd).le.epsilon.or.DABS(de/energy).lt.1.0D-12)&
  EXIT
if( matchlogd * matchold .lt. 0.0D0) de = -0.5D0*de
energy = energy + de
matchold = matchlogd
enddo ! do iter=1,10000
close(11)
!

```

```

!----- Solution has been found and now it is stored:
norm      = integrate(psi,dx,Nx)
norm      = 1.0D0/sqrt(norm)
do i=1,Nx
  psi(i)   = norm*psi(i)
enddo
!----- Count number of zeroes, add one and get energy level:
nodes     = 1
psiold    = psi(1)
do i=2,Nx-1
  !should be 0 within epsilon:
  if( DABS(psi(i)) .gt. epsilon)then
    if( psiold*psi(i).lt.0.0D0 )nodes = nodes+1
    psiold = psi(i)
  endif
enddo !i=2,Nx-1
!----- Print final solution:
open(unit=11,file='psi.dat')
print *, 'Final result: E= ',energy, ' n= ',nodes,&
      ' norm = ', norm
if( DABS(matchlogd) .gt. epsilon) print * &
      'Final result: SOS: logd>epsilon. logd= ',matchlogd
write(11,*) '# E= ', energy, ' n= ',nodes,&
      ' norm = ', norm
do i=1,Nx
  x = xmin + (i-1)*dx
  write(11,*) x,psi(i)
enddo
close(11)
end program schroedinger_equation_1D

!=====
!Simpson's rule to integrate psi(x)*psi(x) for proper
!normalization. For n intervals of width dx (n even)
!Simpson's rule is:
!int(f(x)dx) =
! (dx/3)*(f(x_0)+4 f(x_1)+2 f(x_2)+...+4 f(x_{n-1})+f(x_n))
!
!Input:   Discrete values of function psi(Nx)
!         Integration step dx
!Returns: Integral(psi(x)psi(x) dx)
!=====
real(8) function integrate(psi,dx,Nx)
  implicit none
  integer :: Nx
  !----- Note: we need P due to geometry of array
  real(8) :: psi(Nx),dx
  !-----
  real(8) :: int
  integer :: i

```

```

!----- zeroth order point:
i      = 1
int    = psi(i)*psi(i)
!----- odd order points (i=k+1 is even):
do i=2,Nx-1,2
  int = int + 4.0D0*psi(i)*psi(i)
enddo
!----- even order points:
do i=3,Nx-2,2
  int = int + 2.0D0*psi(i)*psi(i)
enddo
!----- last point:
i      = Nx
int    = int + psi(i)*psi(i)
!----- measure normalization:
int    = int*dx/3.0D0
!----- final result:
integrate = int
end function integrate
!=====

```

Αφήνουμε ως άσκηση στον αναγνώστη να αναπαράγει τα αποτελέσματα της προηγούμενης παραγράφου για το απειρόβαθο πηγάδι δυναμικού. Για το τρέξιμο του προγράμματος, μεταγλωττίστε τον κώδικα και τρέξτε τον με την εντολή:

```

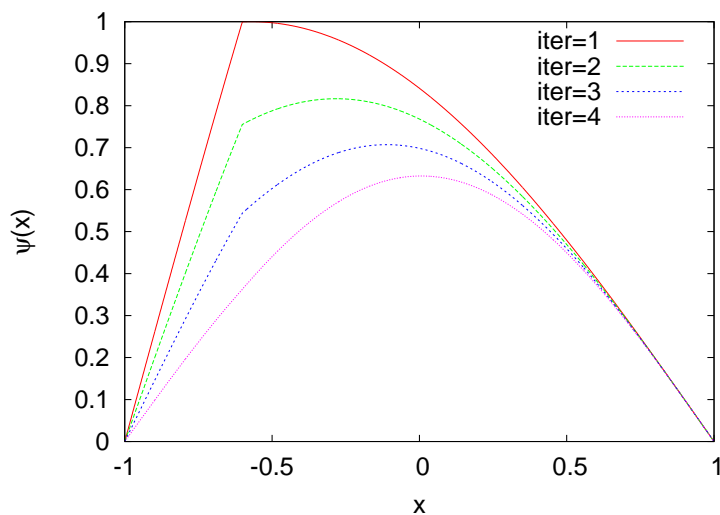
> gfortran sch.f90 schInfSq.f90 rk.f90 -o s
> ./s
# Enter energy ,de ,xmin ,xmax ,Nx
1 0.5 -1 1 2000
# #####
# Estart=      1.000  de=      0.5
# Nx=          2001  eps=     1.0E-006
# xmin=       -1.000  xmax=     1.000  dx=     1.000E-003
# psi(xmin)= 0.000  psip(xmin)=      1.000
# psi(xmax)= 0.000  psip(xmax)=     -1.000
# #####
# iter ,energy ,de ,xm ,logd:  1 1.0000  0.500    -0.60 -0.9748
# iter ,energy ,de ,xm ,logd:  2 1.5000  0.500    -0.60 -0.6412
# .....
# iter ,energy ,de ,xm ,logd: 30 2.4674 -3.82E-6  -0.60 -1.0E-6
# iter ,energy ,de ,xm ,logd: 31 2.4674  1.91E-6  -0.60  2.8E-7
Final result: E= 2.467401504516602 n= 1 norm = 1.5707965025

```

Παραπάνω θέσαμε  $x_{\min} = -1$ ,  $x_{\max} = 1$ ,  $N_x = 2000$ , καθώς και  $\epsilon = 1$ ,  $\delta\epsilon = 0.5$ . Τελικά, πήραμε ότι η ενέργεια της πρώτης (θεμελιώδους,  $n=1$ ) ενεργειακής κατάστασης είναι  $\epsilon_1 = 2.4674015045166016$ . Η κυματοσυ-

νάρτηση βρίσκεται στο αρχείο `psi.dat` και μπορούμε να τη δούμε γραφικά με το `gnuplot` με την εντολή

```
gnuplot> plot "psi.dat" using 1:2 with lines
```



Σχήμα 10.6: Η διαδικασία σύγκλισης των λύσεων της εξίσωσης Schrödinger όπως περιγράφεται στη σελίδα 447 για τη θεμελιώδη κατάσταση στο άπειρο πηγάδι δυναμικού.

Η διαδικασία σύγκλισης στην κυματοσυνάρτηση είναι αποθηκευμένη στο αρχείο `all.dat`. Η πρώτη στήλη έχει τον αριθμό προσπάθειας σύγκλισης (εδώ `iter = 0, ... 31`) και μπορούμε εύκολα να φιλτράρουμε κάθε προσπάθεια με τις εντολές

```
gnuplot> plot "<awk '$1==1' all.dat" using 3:4 w l t "iter=1"
gnuplot> replot "<awk '$1==2' all.dat" using 3:4 w l t "iter=2"
gnuplot> replot "<awk '$1==3' all.dat" using 3:4 w l t "iter=3"
gnuplot> replot "<awk '$1==4' all.dat" using 3:4 w l t "iter=4"
.....
```

οι οποίες παράγουν το σχήμα 10.6.

## 10.4 Μετρήσεις

Όταν για μια κατάσταση  $|\psi\rangle$  γνωρίζουμε την κυματοσυνάρτηση σε αναπαράσταση θέσης  $\psi(x)$ , είναι εύκολο να υπολογίσουμε τη δράση τελε-

στών που αντιστοιχούν σε παρατηρήσιμες ποσότητες  $\mathcal{A}(x, p)$  που είναι συνάρτηση της θέσης και της ορμής. Η δράση των τελεστών

$$\hat{x}\psi(x) = x\psi(x) \quad \hat{p}\psi(x) = -i\frac{\partial}{\partial x}\psi(x) \quad (10.29)$$

δίνουν<sup>8</sup>

$$\hat{\mathcal{A}}(\hat{x}, \hat{p})\psi(x) = \mathcal{A}(x, -i\frac{\partial}{\partial x})\psi(x). \quad (10.30)$$

Χρησιμοποιώντας την (10.9) μπορούμε να υπολογίσουμε την αναμενόμενη (ή μέση) τιμή  $\langle \mathcal{A} \rangle$  του τελεστή  $\mathcal{A}$ , όταν το σύστημα βρίσκεται στην κατάσταση  $|\psi\rangle$ . Ενδιαφέροντα παραδείγματα αποτελούν οι παρατηρήσιμες ποσότητες “θέση”  $x$ , “θέση τετράγωνο”  $x^2$ , “ορμή”  $p$ , “ορμή τετράγωνο”  $p^2$ , “κινητική ενέργεια”  $T$ , “δυναμική ενέργεια”  $V$ , “ενέργεια” ή “Χαμιλτονιανή”  $H = T + V$  των οποίων οι μέσες τιμές δίνονται από τις σχέσεις

$$\begin{aligned} \langle x \rangle &= \int_{-\infty}^{+\infty} \psi^*(x) x \psi(x) dx \\ \langle x^2 \rangle &= \int_{-\infty}^{+\infty} \psi^*(x) x^2 \psi(x) dx \\ \langle p \rangle &= \int_{-\infty}^{+\infty} \psi^*(x) \left( -i\frac{\partial}{\partial x} \right) \psi(x) dx \\ \langle p^2 \rangle &= \int_{-\infty}^{+\infty} \psi^*(x) \left( -\frac{\partial^2}{\partial x^2} \right) \psi(x) dx \\ \langle T \rangle &= \frac{\hbar^2}{2mL^2} \int_{-\infty}^{+\infty} \psi^*(x) \left( -\frac{\partial^2}{\partial x^2} \right) \psi(x) dx \\ \langle V \rangle &= \frac{\hbar^2}{2mL^2} \int_{-\infty}^{+\infty} \psi^*(x) v(x) \psi(x) dx \\ \langle H \rangle &= \frac{\hbar^2}{2mL^2} \int_{-\infty}^{+\infty} \psi^*(x) \left( -\frac{\partial^2}{\partial x^2} + v(x) \right) \psi(x) dx. \end{aligned} \quad (10.31)$$

Στις παραπάνω σχέσεις χρησιμοποιήσαμε αδιάστατες  $x, p$  και τις σχέσεις (10.15) και (10.16). Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι “αβεβαιότητες”  $\Delta x^2 = \langle x^2 \rangle - \langle x \rangle^2$ ,  $\Delta p^2 = \langle p^2 \rangle - \langle p \rangle^2$  οι οποίες πρέπει να ικανοποιούν τη σχέση (“αρχή αβεβαιότητας Heisenberg”)

$$\Delta x \cdot \Delta p \geq \frac{1}{2}. \quad (10.32)$$

<sup>8</sup>Εδώ δε λαμβάνουμε υπόψη προβλήματα διάταξης τελεστών που δεν μετατίθενται λ.χ.  $xp^2$ .



Στις προηγούμενες παραγράφους εξηγούμε πώς να υπολογίζουμε αριθμητικά τις κυματοσυναρτήσεις των ιδιοκαταστάσεων της ενέργειας. Σε αυτές, επειδή  $\hat{H}\psi(x) = E\psi(x)$ , παίρνουμε  $\langle H \rangle = (1/2mL^2)\epsilon$ , αλλά οι υπόλοιποι τελεστές θέλουν κάποια αριθμητική προσέγγιση για τον υπολογισμό των μέσων τιμών τους. Αν οι τιμές της κυματοσυνάρτησης δίνονται σε  $N$  ισαπέχοντα σημεία  $x_1, x_2, \dots, x_N$ , θα πάρουμε

$$\frac{\partial\psi(x_i)}{\partial x} \approx \frac{\psi(x_{i+1}) - \psi(x_{i-1}))}{2h} \quad (10.33)$$

όπου  $h = x_{i+1} - x_i$  και

$$\frac{\partial^2\psi(x_i)}{\partial x^2} \approx \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1}))}{h^2}. \quad (10.34)$$

Και οι δύο τύποι έχουν ακρίβεια  $\mathcal{O}(h^2)$ . Θα πρέπει να προσέξουμε λίγο την εφαρμογή των τύπων στα άκρα του διαστήματος  $[x_1, x_N]$ . Αρχικά, θα χρησιμοποιήσουμε τις προσεγγίσεις<sup>9</sup>

$$\begin{aligned} \frac{\partial\psi(x_1)}{\partial x} &\approx \frac{\psi(x_2) - \psi(x_1)}{h} \\ \frac{\partial\psi(x_N)}{\partial x} &\approx \frac{\psi(x_N) - \psi(x_{N-1}))}{h} \end{aligned} \quad (10.35)$$

και

$$\begin{aligned} \frac{\partial^2\psi(x_1)}{\partial x^2} &\approx \frac{\psi(x_3) - 2\psi(x_2) + \psi(x_1)}{h^2} \\ \frac{\partial^2\psi(x_N)}{\partial x^2} &\approx \frac{\psi(x_N) - 2\psi(x_{N-1}) + \psi(x_{N-2}))}{h^2}. \end{aligned} \quad (10.36)$$

Το σχετικό πρόγραμμα υπολογισμού των  $\langle x \rangle$ ,  $\langle x^2 \rangle$ ,  $\langle p \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x$ ,  $\Delta p$  προγραμματίζεται στο αρχείο `observables.f90` και δίνεται παρακάτω:

```
!=====
!  
! File observables.f90  
! Compile: gfortran observables.f90 -o o  
! Usage:   ./o <psi.dat>  
!  
! Read in a file with a wavefunction in the format of psi.dat:
```

<sup>9</sup>Στο συνοδευτικό λογισμικό `observables.f90`, `Derivatives.nb` εξηγείται πώς να χρησιμοποιήσετε σχέσεις με ακρίβεια  $\mathcal{O}(h^2)$ . Στα παραδείγματα που θα χρησιμοποιήσουμε η επίδραση της μειωμένης ακρίβειας  $\mathcal{O}(h)$  στα αποτελέσματα είναι περίπου στο 4ο δεκαδικό ψηφίο.

```

! # E= <energy> ....
! x1  psi(x1)
! x2  psi(x2)
! .....
!
! Outputs expectation values:
! normalization Energy <x> <p> <x^2> <p^2> Dx Dp DxDP
! where Dx = sqrt(<x^2>-<x>^2) Dp = sqrt(<p^2>-<p>^2)
! DxDP = Dx * Dp
!
!=====
program observables_expectation
  implicit none
  integer ,parameter :: P=50000
  integer Nx,i
  real(8) :: xstep(P),psi(P),obs(P)
  real(8) :: xav, pav, x2av, p2av, Dx, Dp, DxDP,energy,h,norm
  real(8) :: integrate
  character(20) :: psifile,scratch

!the first argument of the command line must be the path
!to the file with the wavefunction. (GNU fortran extension...)
  if( iargc() .ne. 1) stop 'Usage: o <filename>'
  call getarg(1,psifile)
!If the file does not exist, we go to label 100 (stop):
  open(unit=11,file=psifile,status='OLD',err=100)
  print *, "# reading wavefunction from file:", psifile
!we read the first comment line from the file:
  read(11,*) scratch,scratch,energy
!-----
!Input data: psi(x)
  Nx = 1
  do while(.TRUE.)
    if(Nx .ge. P) stop 'Too many points'
    read(11,*,end=101) xstep(Nx),psi(Nx)
    Nx = Nx+1
  enddo !do while(.TRUE.)
101 continue
  Nx = Nx - 1
  if(mod(Nx,2) .eq. 0) Nx = Nx - 1
  h = (xstep(Nx)-xstep(1))/(Nx-1)
!-----
!Calculate:
!----- norm:
  do i=1,Nx
    obs(i) = psi(i)*psi(i)
  enddo
  norm = integrate(obs,h,Nx)
!----- <x> :

```

```

do i=1,Nx
  obs(i) = xstep(i)*psi(i)*psi(i)
enddo
xav = integrate(obs,h,Nx)/norm
!----- <p>/i :
obs(1) = psi(1)*(psi(2)-psi(1))/h
do i=2,Nx-1
  obs(i) = psi(i)*(psi(i+1)-psi(i-1))/(2.0D0*h)
enddo
obs(Nx) = psi(Nx)*(psi(Nx)-psi(Nx-1))/h
pav = -integrate(obs,h,Nx)/norm
!----- <x^2>
do i=1,Nx
  obs(i) = xstep(i)*xstep(i)*psi(i)*psi(i)
enddo
x2av = integrate(obs,h,Nx)/norm
!----- <p^2>
obs(1) = psi(1)*(psi(3)-2.0D0*psi(2)+psi(1))/(h*h)
do i=2,Nx-1
  obs(i) = psi(i)*(psi(i+1)-2.0D0*psi(i)+psi(i-1))/(h*h)
enddo
obs(Nx) = psi(Nx)*&
  (psi(Nx)-2.0D0*psi(Nx-1)+psi(Nx-2))/(h*h)
p2av = -integrate(obs,h,Nx)/norm
!----- Dx
Dx = sqrt(x2av - xav*xav)
!----- Dp
Dp = sqrt(p2av - pav*pav)
!----- Dx . Dp
DxDp = Dx*Dp
!print results:
print *, '# norm E <x> <p>/i <x^2> <p^2> Dx Dp DxDp'
print '(10G25.17)', norm, energy, xav, pav, x2av, p2av, Dx, Dp, DxDp
stop !normal execution ends here. Error messages follow
100 stop 'Cannot open file'
end program observables_expectation
!=====
!
!Simpson's rule to integrate psi(x)*psi(x) for proper
!normalization. For n intervals of width dx (n even)
!Simpson's rule is:
!int(f(x)dx) =
! (dx/3)*(f(x_0)+4 f(x_1)+2 f(x_2)+...+4 f(x_{n-1})+f(x_n))
!
!Input: Discrete values of function psi(Nx)
! Integration step dx
!Returns: Integral(psi(x)psi(x) dx)
!=====
real(8) function integrate(psi,dx,Nx)

```

```

implicit none
integer :: Nx
real(8) :: psi(Nx),dx
real(8) :: int
integer i
!----- zeroth order point:
i      = 1
int    = psi(i)
!----- odd order points (i=k+1 is even):
do i=2,Nx-1,2
  int = int + 4.0D0*psi(i)
enddo
!----- even order points:
do i=3,Nx-2,2
  int = int + 2.0D0*psi(i)
enddo
!----- last point:
i      = Nx
int    = int + psi(i)
!----- measure normalization:
int    = int*dx/3.0D0
!----- final result:
integrate = int
end function integrate
!=====

```

Για τη χρήση του προγράμματος χρειάζεστε την κυματοσυνάρτηση στα σημεία  $x_1, \dots, x_{N_x}$  στο φορμά που παράγουν τα προγράμματα που γράψαμε μέχρι τώρα. Στην πρώτη γραμμή θα πρέπει να είναι καταγεγραμμένη η ενέργεια στην 3η στήλη, ενώ από τη 2η γραμμή και μετά έχουμε δύο στήλες με τα ζευγάρια  $(x_i, \psi(x_i))$ . Η κυματοσυνάρτηση δεν είναι αναγκαίο να είναι κανονικοποιημένη, το κάνει το πρόγραμμα. Αν τα παραπάνω δεδομένα είναι καταγεγραμμένα στο αρχείο `psi.dat`, τότε η χρήση του προγράμματος γίνεται με τις εντολές

```

> gfortran observables.f90 -o obs
> ./obs psi.dat

```

Το πρόγραμμα τυπώνει στο `stdout` τη σταθερά κανονικοποίησης της  $\psi(x)$ , την ενέργεια (διαβασμένη από το αρχείο - όχι υπολογισμένη), και στη συνέχεια οι  $\langle x \rangle$ ,  $\langle x^2 \rangle$ ,  $\langle p \rangle/i$ ,  $\langle p^2 \rangle$ ,  $\Delta x$ ,  $\Delta p$  και το γινόμενο  $\Delta x \cdot \Delta p$ .

Μερικές διευκρινήσεις πάνω στις λεπτομέρειες του προγράμματος. Για να διαβάσουμε τα δεδομένα από το αρχείο `psi.dat` χρησιμοποιούμε τις συναρτήσεις `iargc()`, `getarg(n,string)`. Η πρώτη επιστρέφει τον αριθμό των arguments στη γραμμή εντολών και η δεύτερη αποθηκεύει το

n-οστό argument στην CHARACTER μεταβλητή string. Οπότε, οι εντολές

```
character(20) :: psifile, scratch
if( iargc().ne. 1) stop 'Usage: o <filename>'
call getarg(1,psifile)
```

σταματάνε το πρόγραμμα, αν η εντολή δεν έχει ακριβώς ένα argument, αλλιώς το πρώτο argument αποθηκεύεται στη μεταβλητή file.

Η εντολή

```
open(unit=11,file=psifile,status='OLD',err=100)
100 stop 'Cannot open file'
```

ανοίγει ένα αρχείο που πρέπει να υπάρχει ήδη (status='OLD'), αλλιώς παράγεται σφάλμα. Ο προσδιορισμός err=100 μεταφέρει τότε τον έλεγχο του προγράμματος στο statement με label '100'. Στο παραπάνω παράδειγμα, σταματάει το πρόγραμμα με μήνυμα σφάλματος 'Cannot open filename'.

Οι εντολές

```
Nx = 1
do while(.TRUE.)
  read(11,*,end=101) xstep(Nx),psi(Nx)
  Nx = Nx+1
enddo !do while(.TRUE.)
101 continue
```

διαβάζουν το αρχείο που ανοίγουμε γραμμή-γραμμή. Ο προσδιορισμός end=101 στο statement read(11,\*,end=101) μεταφέρει τον έλεγχο του προγράμματος στο statement με label 101 (δηλ. εκτός του do loop), όταν φτάσουμε στο τέλος του αρχείου.

Οι υπόλοιπες εντολές είναι εφαρμογές των σχέσεων (10.33), (10.34), (10.35) και (10.36) στους τύπους (10.31) και ο αναγνώστης παρακαλείται να τις μελετήσει προσεκτικά. Επίσης, χρησιμοποιείται και η ρουτίνα integrate για τα απαραίτητα ολοκληρώματα.

## 10.5 Ο Αναρμονικός Ταλαντωτής - Ξανά...

Στο Κεφάλαιο 9 μελετήσαμε τον αρμονικό και αναρμονικό ταλαντωτή στην αναπαράσταση των ιδιοκαταστάσεων ενέργειας του αρμονικού ταλαντωτή  $|n\rangle$ . Στην παράγραφο αυτή, θα επανεξετάσουμε το πρόβλημα

στην αναπαράσταση θέσης. Θα υπολογίσουμε τις κυματοσυναρτήσεις  $\psi_{n,\lambda}(x)$  που διαγωνιοποιούν τη Χαμιλτονιανή (9.15), είναι δηλ. λύσεις της εξίσωσης Schrödinger. Θέτοντας  $L = \sqrt{\hbar/m\omega}$  στην (10.13), η (10.12) γίνεται:

$$\psi''(x) = -(\epsilon - v(x))\psi(x) \quad (10.37)$$

με  $v(x) = x^2 + 2\lambda x^4$ . Για  $\lambda = 0$  παίρνουμε τον αρμονικό ταλαντωτή με

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-x^2/2} H_n(x), \epsilon_n = 2 \left( n + \frac{1}{2} \right), \quad (10.38)$$

όπου  $H_n(x)$  είναι τα πολυώνυμα Hermite.

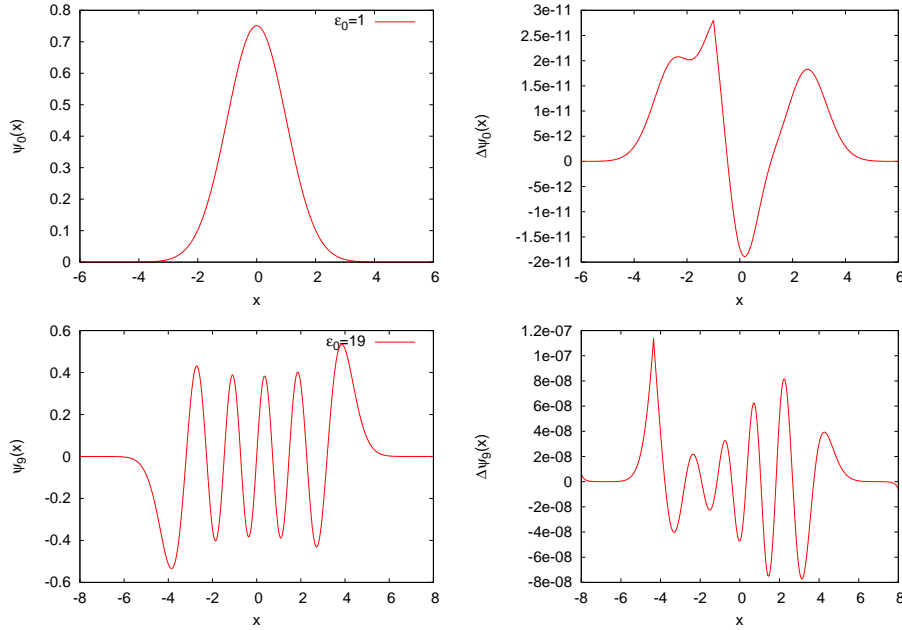
Για έλεγχο του προγράμματος και της ακρίβειας της διαδικασίας, ξεκινάμε από τον αρμονικό ταλαντωτή. Το δυναμικό και οι αρχικές συνθήκες προγραμματίζονται στο αρχείο schHOC.f90. Οι αλλαγές που γίνονται αφορούν τις συναρτήσεις  $V(x)$ , boundary(xmin, xmax, psixmin, psipxmin, psixmax, psipxmax):

```
!=====
!file: schHOC.f
!.....
!----- potential:
real(8) function V(x)
  implicit none
  real(8) :: x
  V = x*x
end function V
!----- boundary conditions:
subroutine boundary(xmin,xmax,&
                   psixmin,psipxmin,psixmax,psipxmax)
  implicit none
  real(8) :: xmin,xmax,psixmin,psipxmin,psixmax,psipxmax,V

  psixmin    = exp(-0.5D0*xmin*xmin)
  psipxmin   = -xmin*psixmin
  psixmax    = exp(-0.5D0*xmax*xmax)
  psipxmax   = -xmax*psixmax
end subroutine boundary
!=====
.....
```

όπου ο κώδικας στις τελείες παραλείπεται, επειδή είναι ίδιος με αυτόν της προηγούμενης παραγράφου. Οι αρχικές συνθήκες λαμβάνουν υπόψη τη γνωστή ασυμπτωτική συμπεριφορά των λύσεων της εξίσω-

σης Schrödinger<sup>10</sup>  $\psi_0(x) \sim e^{-x^2/2}$ ,  $\psi'_n(x) \sim -x\psi_n(x)$ . Δοκιμάστε να τις αλλάξετε για να δείτε αν υπάρχει επίδραση στα αποτελέσματα. Τα



Σχήμα 10.7: Ο υπολογισμός των κυματοσυναρτήσεων  $\psi_0(x)$ ,  $\psi_9(x)$  από το πρόγραμμα sch.f90, schH0C.f90. Στα διαγράμματα δεξιά φαίνεται η διαφορά των τιμών τους από τις αναμενόμενες τιμές (10.38).

αποτελέσματα δίνονται γραφικά στο σχήμα 10.7 όπου πέρα από την ποιοτική συμφωνία, διαγράφεται και η απόκλιση από τις αναλυτικά υπολογισμένες τιμές (10.38) που είναι της τάξης  $10^{-11}$ – $10^{-7}$ . Οι τιμές των ενεργειών  $\epsilon_n$  βρίσκονται σε συμφωνία με την (10.38) με σχετική ακρίβεια καλύτερη από  $10^{-9}$  για  $n \leq 14$ .

Στη συνέχεια, υπολογίζουμε τις τιμές των μέσων τιμών  $\langle x \rangle$ ,  $\langle x^2 \rangle$ ,  $\langle p \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x$  και  $\Delta p$ . Αυτές υπολογίζονται πολύ εύκολα από τις σχέσεις (9.4) και (9.8). Βλέπουμε ότι  $\langle x \rangle = \langle n | (a^\dagger + a) / \sqrt{2} | n \rangle = 0$ ,  $\langle p \rangle = \langle n | i(a^\dagger - a) / \sqrt{2} | n \rangle = 0$ , ενώ

$$\langle x^2 \rangle = \langle p^2 \rangle = \langle n | \frac{1}{2}(a^\dagger a + a a^\dagger) | n \rangle = \left( n + \frac{1}{2} \right). \quad (10.39)$$

Το πρόγραμμα observables.f90 μας δίνει  $\langle x \rangle = 0$  με ακρίβεια  $\sim 10^{-6}$

<sup>10</sup>Κανονικά  $\psi_n(x) \sim x^n e^{-x^2/2}$  το οποίο αγνοούμε και βρίσκουμε ότι δεν παίζει σημαντικό ρόλο στα τελικά αποτελέσματα για τα  $n$  που μελετάμε. Δοκιμάστε αν αυτό είναι αναγκαίο να το λάβουμε υπόψη για πολύ μεγαλύτερα  $n$ .

$n$	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x \cdot \Delta p$
0	0.500000000	0.4999977	0.4999989
1	1.500000284	1.4999883	1.4999943
2	2.499999747	2.4999711	2.4999854
3	3.499999676	3.4999441	3.4999719
4	4.499999607	4.4999082	4.4999539
5	5.499999520	5.4998633	5.4999314
6	6.499999060	6.4998098	6.4999044
7	7.499999642	7.4995484	7.4997740
8	8.499999715	8.4994203	8.4997100
9	9.499999837	9.4992762	9.4996380
10	10.500000012	10.4991160	10.4995580
11	11.499999542	11.4994042	11.4997019
12	12.499999610	12.4992961	12.4996479
13	13.499999705	13.4991791	13.4995894
14	14.499999835	14.4990529	14.4995264

Πίνακας 10.2: Οι αναμενόμενες τιμές των  $\langle x^2 \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x \cdot \Delta p$  για τον απλό αρμονικό ταλαντωτή για τις καταστάσεις  $|n\rangle$ ,  $n = 0, \dots, 14$ .

και  $\langle p \rangle = 0$  με ακρίβεια  $\sim 10^{-11}$ . Οι τιμές των  $\langle x^2 \rangle$ ,  $\langle p^2 \rangle$  δίνονται στον πίνακα 10.2.

Στη συνέχεια, επαναλαμβάνουμε τον υπολογισμό μας για τον αναρμονικό ταλαντωτή και για  $\lambda = 0.5, 2.0$ . Αυτό γίνεται μετατρέποντας το αρχείο `schHOC.f90` που αποθηκεύεται στο αρχείο `schUOC.f90`. Απλά αλλάζουμε το δυναμικό σε:

```
!=====
!file : schUOC.f
!.....
!----- potential:
real(8) function V(x)
  implicit none
  real(8) :: x, lambda
  lambda = 2.0D0
  V = x*x+2.0D0*lambda*x*x*x*x
end function V
.....
```

Οι κυματοσυναρτήσεις δείχνονται στο σχήμα 10.8 όπου φαίνεται ότι η αύξηση του  $\lambda$  οδηγεί σε επιπλέον περιορισμό του σωματιδίου στο χώρο όπως αναμένεται. Στον πίνακα 10.3 καταχωρούνται οι τιμές της  $\epsilon_n$  για

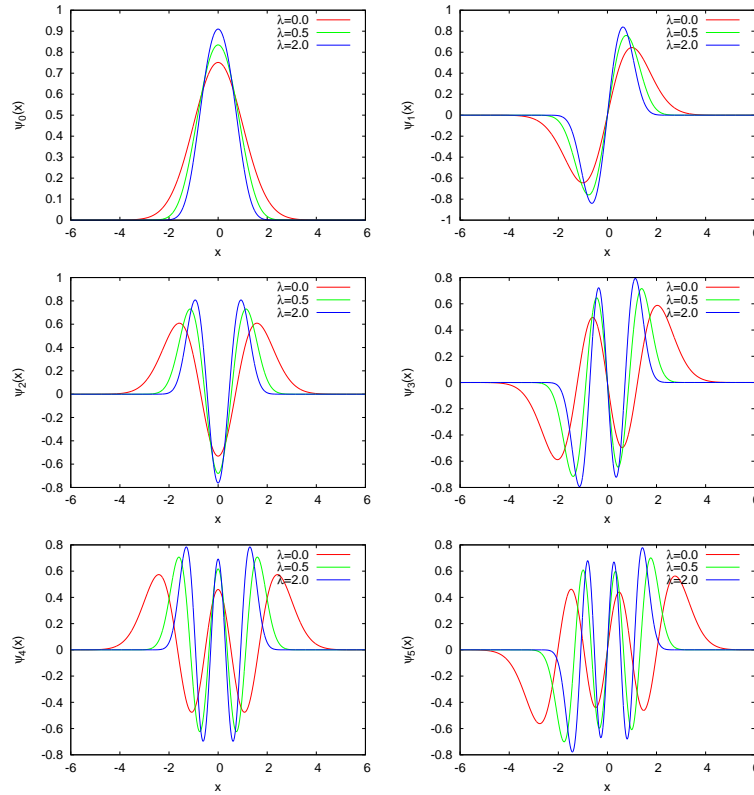


$n$	$\epsilon_n$	$\epsilon_{n,\lambda=0.5}$	$\epsilon_{n,\lambda=2.0}$
0	1.0000	1.3924	1.9031
1	3.0000	4.6488	6.5857
2	5.0000	8.6550	12.6078
3	7.0000	13.1568	19.4546
4	9.0000	18.0576	26.9626
5	11.0000	23.2974	35.0283
6	13.0000	28.8353	43.5819
7	15.0000	34.6408	52.5723
8	17.0000	40.6904	61.9598
9	19.0000	46.9650	71.7129

Πίνακας 10.3: Οι τιμές της ενέργειας  $\epsilon_n$  για τον αρμονικό ταλαντωτή καθώς και τον αναρμονικό ταλαντωτή για  $\lambda = 0.5, 2.0$ . Παρατηρείται η αύξηση των τιμών της ενέργειας για αυξανόμενο  $\lambda$ .

$n$	$\lambda = 0.5$			$\lambda = 2.0$		
	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x \cdot \Delta p$	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x \cdot \Delta p$
0	0.3058	0.8263	0.5027	0.2122	1.1980	0.5042
1	0.8013	2.8321	1.5064	0.5408	4.2102	1.5089
2	1.1554	5.3848	2.4944	0.7612	8.1513	2.4909
3	1.4675	8.2819	3.4862	0.9582	12.6501	3.4816
4	1.7509	11.4545	4.4784	1.1370	17.5955	4.4728
5	2.0141	14.8599	5.4707	1.3029	22.9169	5.4643
6	2.2617	18.4691	6.4631	1.4590	28.5668	6.4560
7	2.4970	22.2607	7.4555	1.6074	34.5103	7.4478
8	2.7220	26.2184	8.4478	1.7492	40.7206	8.4397
9	2.9384	30.3289	9.4402	1.8856	47.1762	9.4316

Πίνακας 10.4: Οι αναμενόμενες τιμές των  $\langle x^2 \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x \cdot \Delta p$  για τον αναρμονικό ταλαντωτή για τις καταστάσεις  $|n\rangle$ ,  $n = 0, \dots, 9$ . Παρατηρούμε τη μείωση της  $\Delta x = \sqrt{\langle x^2 \rangle}$  και αύξηση της  $\Delta p = \sqrt{\langle p^2 \rangle}$  καθώς αυξάνεται το  $\lambda$ . Το γινόμενο  $\Delta x \cdot \Delta p$  φαίνεται να είναι πολύ κοντά στις τιμές που παίρνουμε από τον αρμονικό ταλαντωτή και για τις δύο τιμές του  $\lambda$ . Συγκρίνετε τα αποτελέσματα που καταγράφονται εδώ με αυτά του πίνακα 9.1.



Σχήμα 10.8: Οι κυματοσυναρτήσεις του αναρμονικού ταλαντωτή  $\psi_{n,\lambda}(x)$  για  $n = 0, 1, 2, 3, 4, 5$  και  $\lambda = 0.5, 2.0$  συγκρινόμενες με αυτές του αρμονικού ταλαντωτή. Φαίνεται ότι η αύξηση του  $\lambda$  οδηγεί σε περιορισμό του σωματιδίου στο χώρο.

$n = 0, \dots, 9$ . Παρατηρείται η αύξηση των τιμών της ενέργειας για αυξανόμενο  $\lambda$ . Στον πίνακα 10.4 καταχωρούνται οι αναμενόμενες τιμές των  $\langle x^2 \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x \cdot \Delta p$  για τον αναρμονικό ταλαντωτή για τις καταστάσεις  $|n\rangle$ ,  $n = 0, \dots, 9$ . Παρατηρούμε τη μείωση της  $\Delta x = \sqrt{\langle x^2 \rangle}$  και αύξηση  $\Delta p = \sqrt{\langle p^2 \rangle}$ , καθώς αυξάνεται το  $\lambda$ . Το γινόμενο  $\Delta x \cdot \Delta p$  φαίνεται να είναι πολύ κοντά στις τιμές που παίρνουμε από τον αρμονικό ταλαντωτή και για τις δύο τιμές του  $\lambda$ . Τα αποτελέσματα μπορείτε να τα συγκρίνετε με αυτά που πήραμε στο κεφάλαιο 9, στον πίνακα 9.1.

## 10.6 Το Δυναμικό Lennard–Jones

Το δυναμικό Lennard–Jones είναι ένα απλό φαινομενολογικό μοντέλο που περιγράφει την αλληλεπίδραση δύο ουδέτερων ατόμων σε ένα δια-

τομικό μόριο. Αυτό δίνεται από

$$V(x) = 4V_0 \left\{ \left( \frac{\sigma}{x} \right)^{12} - \left( \frac{\sigma}{x} \right)^6 \right\}. \quad (10.40)$$

Ο απωστικός όρος περιγράφει την άπωση επικάλυψης των ηλεκτρονικών νεφών κατά Pauli, ενώ ο ελκτικός όρος τη δύναμη Van der Waals. Επιλέγουμε  $L = \sigma$  στην (10.13) και ορίζουμε  $v_0 = 2m\sigma^2 V_0 / \hbar^2$ . Η (10.40) γίνεται

$$v(x) = 4v_0 \left\{ \left( \frac{1}{x} \right)^{12} - \left( \frac{1}{x} \right)^6 \right\}, \quad (10.41)$$

ενώ οι υπολογιζόμενες ιδιοτιμές  $\epsilon_n$  συνδέονται με την ενέργεια μέσω της σχέσης

$$\epsilon_n = 4v_0 \left( \frac{E_n}{V_0} \right). \quad (10.42)$$

Το δυναμικό αναπαρίσταται γραφικά στο σχήμα 10.5 για  $v_0 = 250$ . Το ελάχιστο του δυναμικού βρίσκεται στη θέση  $x_m = 2^{1/6} \approx 1.12246$  και η τιμή του είναι  $-v_0$ . Ο προγραμματισμός του δυναμικού γίνεται στο αρχείο schLJ.f90. Το κομμάτι του κώδικα που μεταβάλλουμε από τα προηγούμενα αρχεία δίνεται παρακάτω:

```
!=====
! file: schLJ.f90 (Lennard-Jones)
! .....
!——— potential:
real(8) function V(x)
  implicit none
  real(8) :: x, V0

  V0 = 250.0D0
  V = 4.0D0*V0*(1.0D0/x**12 - 1.0D0/x**6)

end function V
!——— boundary conditions:
subroutine boundary(xmin, xmax, psixmin, psipxmin, psixmax, psipxmax ←
)
  implicit none
  real(8) :: xmin, xmax, psixmin, psipxmin, psixmax, psipxmax, V
  real(8) :: energy
  common/params/energy
!——— Initial values at xmin and xmax
  psixmin = exp(-xmin*sqrt(DABS(energy-V(xmin))))
  psipxmin = sqrt(DABS(energy-V(xmin)))*psixmin
  psixmax = exp(-xmax*sqrt(DABS(energy-V(xmax))))
```

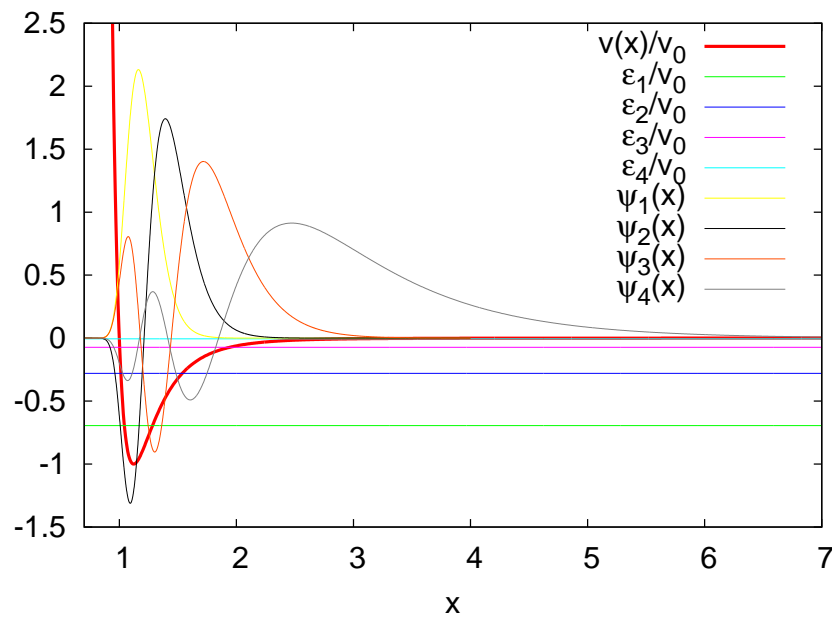
$n$	$\epsilon_n$	$\langle x \rangle$	$\langle p \rangle$	$\langle x^2 \rangle$	$\langle p^2 \rangle$	$\Delta x$	$\Delta p$	$\Delta x \cdot \Delta p$
0	-173.637	1.186	1.0e-10	1.415	34.193	0.091	5.847	0.534
1	-70.069	1.364	6.0e-11	1.893	56.832	0.178	7.539	1.338
2	-18.191	1.699	-4.5e-08	2.971	39.480	0.291	6.283	1.826
3	-1.317	2.679	-2.6e-08	7.586	9.985	0.638	3.160	2.016

Πίνακας 10.5: Τα αποτελέσματα των μετρήσεων για το δυναμικό Lennard-Jones με  $v_0 = 250$ . Έχουμε 4 δέσμιες καταστάσεις.

```

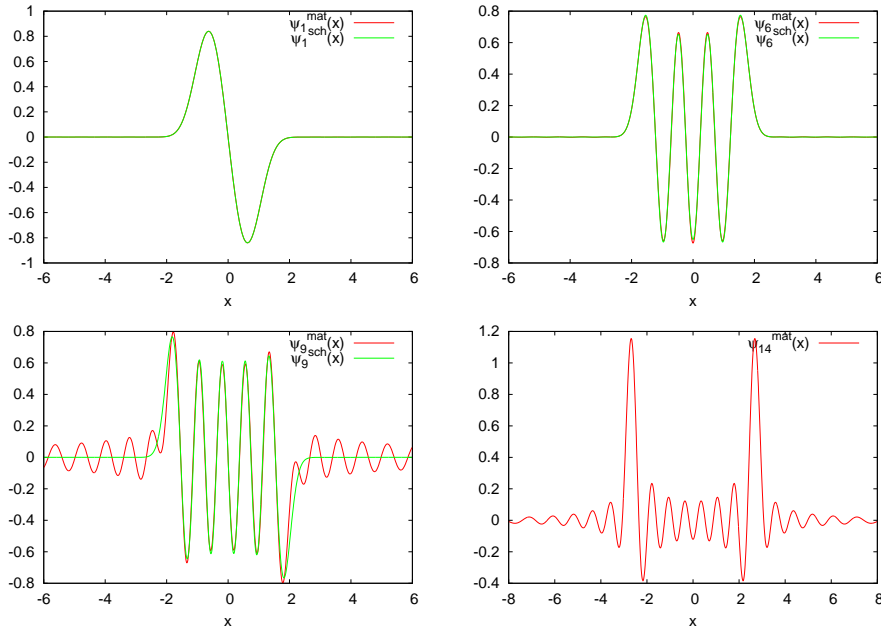
psipxmax = -sqrt(DABS(energy-V(xmax)))*psixmax
end subroutine boundary
.....

```



Σχήμα 10.9: Οι τέσσερις δέσμιες καταστάσεις για το δυναμικό Lennard-Jones με  $v_0 = 250$ . Φαίνεται το δυναμικό  $v(x)/v_0$  με παχιά κόκκινη γραμμή, τα ενεργειακά επίπεδα  $\epsilon_n/v_0$  και οι αντίστοιχες κυματοσυναρτήσεις.

Για την ολοκλήρωση επιλέγουμε τις παραμέτρους  $v_0 = 250$  και  $x_{\min} = 0.7$ ,  $4 < x_{\max} < 10$ . Τα αποτελέσματα παρουσιάζονται στο σχήμα 10.9, όπου φαίνονται τα τέσσερα ενεργειακά επίπεδα των δέσμιων καταστάσεων μαζί με τις κυματοσυναρτήσεις τους. Αυτές είναι περιορισμένες μέσα στο πηγάδι του δυναμικού για τις δύο πρώτες στάθμες, ενώ αρ-



Σχήμα 10.10: Σύγκριση των αποτελεσμάτων για τον υπολογισμό των κυματοσυναρτήσεων  $\psi_{n,\lambda}(x)$  του αναρμονικού ταλαντωτή για  $\lambda = 2.0$  με τις μεθόδους που περιγράφονται στην άσκηση 12. Οι κυματοσυναρτήσεις  $\psi^{\text{sch}}(x)$  αναφέρονται στις  $\psi_{n,\lambda}(x)$  που υπολογίζονται χρησιμοποιώντας τις μεθόδους που περιγράφονται στο κεφάλαιο αυτό. Οι κυματοσυναρτήσεις  $\psi^{\text{mat}}(x)$  αναφέρονται στις  $\psi_{n,\lambda}(x)$  που υπολογίζονται χρησιμοποιώντας τις μεθόδους που περιγράφονται στο Κεφάλαιο 9 με διάσταση χώρου Hilbert  $N = 40$ . Παρατηρούμε ότι οι τελευταίες παρουσιάζουν αποκλίσεις για μεγάλα  $x$ . Αυτό οφείλεται στο ότι στα πλάτη  $\psi_{n,\lambda}(x) = \langle x|n \rangle_\lambda$  για μεγάλα  $x$  συνεισφέρουν καταστάσεις  $|m\rangle$  μεγάλης ενέργειας (γιατί;).

χίζουν να “ξεχειλίζουν” για τις δύο τελευταίες. Στον πίνακα 10.5 παραθέτουμε τις μετρήσεις μας. Παρατηρούμε ότι  $\langle p \rangle = 0$  μέσα στα όρια της ακρίβειας που έχουμε θέσει, όπως περιμένουμε για πραγματικές, δέσμιες κυματοσυναρτήσεις<sup>11</sup>.

<sup>11</sup>Για  $\psi(+\infty) = \psi(0) = 0$  και  $\psi^*(x) = \psi(x)$  έχουμε  $i\langle p \rangle/\hbar = \int_0^{+\infty} \psi(x)(d/dx)\psi(x) dx = -\int_0^{+\infty} (d/dx)\psi(x)\psi(x) dx = 0$ .

## 10.7 Ασκήσεις

- 10.1 Προσθέστε τον κατάλληλο κώδικα στο πρόγραμμα `well.f90`, έτσι ώστε η τελική κυματοσυνάρτηση να τυπώνεται στο `psi.dat` σωστά κανονικοποιημένη. Για το ολοκλήρωμα  $\int_{-1}^1 \psi(x)\psi(x) dx$  χρησιμοποιήστε τον κανόνα του Simpson:

$$\int_a^b f(x) dx = (h/3) (f(x_0) + 4f(x_1) + 2f(x_2) + \dots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n))$$

όπου το διάστημα  $[a, b]$  έχει χωριστεί σε διαστήματα πλάτους  $h$  από  $n$  σημεία  $x_0 = a, x_1, x_2, \dots, x_n = b$  που το  $n$  είναι άρτιος αριθμός.

- 10.2 Προσθέστε τον κατάλληλο κώδικα στο πρόγραμμα `well.f90`, έτσι ώστε να υπολογίζετε τον αριθμό των σημείων μηδενισμού της κυματοσυνάρτησης. Από αυτά, το πρόγραμμα να τυπώνει το ενεργειακό επίπεδο  $n$  της υπολογιζόμενης κυματοσυνάρτησης  $\psi_n(x)$ .
- 10.3 Υπολογίστε τις κυματοσυναρτήσεις των ενεργειακών ιδιοκαταστάσεων στο δυναμικό (10.27) με  $v_0 < 0$ . Αυτό είναι το πρόβλημα του πεπερασμένου πηγαδιού δυναμικού. Λύστε το για  $v_0 = -100$  και  $a = 0.3$ . Πόσες δέσμιες καταστάσεις έχετε μέσα στο πεπερασμένο πηγάδι; Στη συνέχεια, μελετήστε την επίδραση του τοίχους πάνω στις λύσεις. Εισάγετε την παράμετρο  $b$ , έτσι ώστε  $v(x \geq b) = +\infty$  και να μελετήσετε την επίδραση του τείχους πάνω στις λύσεις. Θέστε  $b = 0.35, 0.4, 0.5, 0.6, 0.8, 1.0, 1.5, 2.0, 2.5, 3.0$  και υπολογίστε τη μεταβολή της ενέργειας στις δύο πρώτες ενεργειακές στάθμες. Εκτιμήστε την ακρίβεια επιτυχίας της μεθόδου σας. Στη συνέχεια, μειώστε την τιμή του  $|v_0|$  μέχρι να εξαφανιστεί η δέσμια κατάσταση μέσα στο πεπερασμένο πηγάδι. Ποια είναι η σχέση μεταξύ του  $a$  και  $v_0$  όταν αυτό συμβαίνει; Συγκρίνετε με τη θεωρητικά αναμενόμενη σχέση που μάθατε στο μάθημα της κβαντομηχανικής. Υπόδειξη: Για τις μεγαλύτερες τιμές του  $b$  να αυξήσετε αρκετά την παράμετρο  $Nx > 1000$  και αν δεν πετύχετε σύγκλιση να μειώσετε την `epsilon`.
- 10.4 Στο διπλό πηγάδι δυναμικού βάλτε  $v_0 = 1000, 5000$ . Παρατηρήστε τον (σχεδόν) εκφυλισμό των καταστάσεων και παραστήστε γραφικά τις κυματοσυναρτήσεις  $\psi_{\pm, n} = (1/\sqrt{2})(\psi_n(x) \pm \psi_{n+1}(x))$ , όπου  $n$  περιττός. Συγκρίνετε με τις αντίστοιχες ενεργειακές στάθμες και

κυματοσυναρτήσεις του απειρόβαθου πηγαδιού. Δοκιμάστε πόσο μεγάλο μπορεί να γίνει το  $v_0$ , ώστε να μην μπορείτε πια να λύσετε το πρόβλημα με ανεκτή ακρίβεια.

Υπόδειξη: Για μεγάλα  $v_0$  έχουμε αύξηση αριθμητικής δυσκολίας. Για  $|x| < a$  η κυματοσυνάρτηση είναι σχεδόν μηδέν, και από αυτή θα πρέπει να προκύψει η μη τετριμμένη κυματοσυνάρτηση για  $a < |x| < 1$ . Άρα, η ακρίβεια θα μειώνεται και θα πρέπει να αυξήσουμε το epsilon μέσα στον κώδικά, ώστε να πετύχουμε σύγκλιση αρκετά γρήγορα.

10.5 Επαναλάβετε τις ασκήσεις 3 και 4 χρησιμοποιώντας το πρόγραμμα sch.f90. Συγκρίνετε τα αποτελέσματά σας.

10.6 Μελετήστε τις δέσμιες καταστάσεις στα δυναμικά

$$v(x) = \begin{cases} 0 & a < |x| \\ -V_0 & b < |x| < a \\ -V_1 & |x| < b \end{cases}$$

για  $a = 1, b = 0.2, V_0 = 100, V_1 = 0, 50$  και

$$v(x) = \begin{cases} V_1 & x < 0 \\ -V_0 & 0 < x < a \\ 0 & a < x \end{cases}$$

για  $a = 1, V_0 = 100, V_1 = +\infty, 10, 100$  και

$$v(x) = \begin{cases} V_1 & a < |x| \\ -V_0 & b < |x| < a \\ 0 & c < |x| < b \\ -V_0 & |x| < c \end{cases}$$

για  $a = 1, b = 0.7, c = 0.6, 0.3, V_0 = 100, V_1 = +\infty, 10, 0$ . Για κάθε περίπτωση, υπολογίστε τα  $\langle x \rangle, \langle x^2 \rangle, \langle p \rangle, \langle p^2 \rangle, \Delta x, \Delta p, \Delta x \cdot \Delta p$ .

10.7 Γράψτε πρόγραμμα που από την κυματοσυνάρτηση που δίνει το πρόγραμμα sch.f90, να υπολογίζει την πιθανότητα το σωματίο να βρίσκεται μέσα σε ένα πεπερασμένο διάστημα  $[x_1, x_2]$ . Στα αποτελέσματα που πήρατε από την προηγούμενη άσκηση, προσδιορίστε τα διαστήματα  $[-x_1, x_1]$  μέσα στα οποία έχουμε πιθανότητα 1/3 να βρούμε το σωματίο.

10.8 Συμπληρώστε τους Πίνακες 10.3 και 10.4 για  $\lambda = 0.2, 0.7, 1.0, 1.3, 1.6, 2.5, 3.0$  και παραστήστε γραφικά κάθε αναμενόμενη τιμή συναρτήσει του  $\lambda$ .

10.9 Θεωρήστε το σωματίο που κινείται μέσα στο δυναμικό

$$V(x) = \frac{\hbar^2}{2m} \alpha^2 \lambda(\lambda - 1) \left\{ \frac{1}{2} - \frac{1}{\cosh^2(\alpha x)} \right\}.$$

Το ενεργειακό φάσμα δίνεται από τη σχέση

$$E_n = \frac{\hbar^2}{2m} \alpha^2 \left\{ \frac{\lambda(\lambda - 1)}{2} - (\lambda - 1 - n)^2 \right\}$$

για τις τιμές του  $n = 0, 1, 2, \dots$  για τις οποίες  $E_n > V_{\min}$ . Υπολογίστε αριθμητικά τις ενεργειακές ιδιοτιμές  $\epsilon_n$  των δέσμιων καταστάσεων θέτοντας  $L = 1/\alpha$  στην (10.13) και  $\lambda = 4$ . Κάνετε τις γραφικές παραστάσεις του δυναμικού  $v(x)$  και των αντίστοιχων κυματοσυναρτήσεων. Υπολογίστε τις μέσες τιμές της θέσης, ορμής, αβεβαιότητες στη θέση, ορμή και το γινόμενο τους. Επαναλάβετε για  $\lambda = 2, 6, 8, 10$ .

10.10 Να γράψετε πρόγραμμα που από την κυματοσυνάρτηση να υπολογίζει τη μέση τιμή της ενέργειας

$$\langle \hat{H} \rangle = \int_{-\infty}^{+\infty} \psi(x) \left( -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) \right) \psi(x) dx,$$

υποθέτοντας πως η  $\psi(x)$  είναι πραγματική. Στη συνέχεια να υπολογίσετε τις  $\psi_n(x)$  για τον αρμονικό ταλαντωτή για  $n = 1, \dots, 10$  και να δείξετε (αριθμητικά) ότι  $\langle \hat{H} \rangle_n = E_n$ .

10.11 Θεωρήστε το σωματίο που κινείται μέσα στο δυναμικό Morse

$$V(x) = D_e \left\{ \left( 1 - e^{-a(r-r_e)} \right)^2 - 1 \right\}.$$

Υπολογίστε το φάσμα των δέσμιων καταστάσεων. Επιλέξτε  $L = 1/a$ ,  $x = ar$ ,  $x_e = ar_e$ ,  $\lambda^2 = 2mD_e/a^2\hbar^2$  και πάρτε

$$v(x) = \lambda^2 \left( e^{-2(x-x_e)} - 2e^{-(x-x_e)} \right).$$

Συγκρίνετε με τις αναλυτικά υπολογισμένες λύσεις

$$\epsilon_n = \left( \lambda - n - \frac{1}{2} \right)^2$$

$$\psi_n(z) = N_n z^{\lambda-n-1/2} e^{-z/2} L_n^{2\lambda-2n-1}(z)$$



με  $z = 2\lambda e^{-(x-x_e)}$ ,  $N_n = n! \sqrt{(2\lambda - 2n - 1)/(\Gamma(n+1)\Gamma(2\lambda - n))}$ , και  $L_n^\alpha(z)$  είναι πολυώνυμο Laguerre που δίνεται από τη σχέση  $L_n^\alpha(z) = (z^{-\alpha} e^z / n!) (d^n / dz^n) (z^{n+\alpha} e^{-z}) = (\Gamma(\alpha+2)/(\Gamma(n+2)\Gamma(\alpha-n+2))) {}_1F_1(-n, \alpha+1, z)$ . Ενδεικτικά μπορείτε να πάρετε  $\lambda = 4$ ,  $x_e = 1$ . Για κάθε περίπτωση υπολογίστε τα  $\langle x \rangle$ ,  $\langle x^2 \rangle$ ,  $\langle p \rangle$ ,  $\langle p^2 \rangle$ ,  $\Delta x$ ,  $\Delta p$ ,  $\Delta x \cdot \Delta p$ .

- 10.12 Να υπολογίσετε τις κυματοσυναρτήσεις των ιδιοκαταστάσεων της Χαμιλτονιανής του αναρμονικού ταλαντωτή για  $\lambda = 2.0$  και  $n = 0, \dots, 15$ . Στη συνέχεια να υπολογίσετε τις κυματοσυναρτήσεις που δίνει το πρόγραμμα `anharmonic.f90` από το Κεφάλαιο 9 για  $N = 15, 40, 100$  και να τις συγκρίνετε με αυτές που υπολογίσατε προηγουμένως.

Υπόδειξη: Να γράψετε πρόγραμμα που να υπολογίζει τις ιδιοσυναρτήσεις ενέργειας του αρμονικού ταλαντωτή

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-x^2/2} H_n(x)$$

όπου τα πολυώνυμα Hermite ικανοποιούν τις σχέσεις

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \quad H_0(x) = 1, \quad H_1(x) = 2x.$$

Το πρόγραμμα `anharmonic.f90` υπολογίζει τις ιδιοκαταστάσεις του αναρμονικού ταλαντωτή

$$|n\rangle_\lambda = \sum_{m=0}^{N-1} H(m+1, n+1) |m\rangle$$

βάζοντας τους γραμμικούς συντελεστές στα στοιχεία του array  $H(N, N)$ . Για τις κυματοσυναρτήσεις τους  $\psi_{n,\lambda}(x)$ ,  $\psi_n(x)$  ισχύει η ίδια σχέση. Από τις  $\psi_n(x)$  και τα  $H(i, j)$  υπολογίστε τις  $\psi_{n,\lambda}(x)$  για  $-8 < x < 8$  για κάθε  $N$  και μελετήστε την ακρίβεια υπολογισμού. Για ποιες τιμές του  $x$  η ακρίβεια δεν είναι καλή; Θυμηθείτε ότι για μεγάλα  $x$  οι καταστάσεις μεγάλης ενέργειας παίζουν σημαντικό ρόλο από ότι για  $x$  μικρά. Το σχήμα 10.10 μπορεί να σας βοηθήσει.



## ΚΕΦΑΛΑΙΟ 11

### Ο Τυχαίος Περιπατητής

Στο κεφάλαιο αυτό θα μελετήσουμε τη διαδρομή που ακολουθεί ένας ... μεθυσμένος, όταν αποφασίσει να περπατήσει από το σημείο που βρίσκεται. Λόγω της τύφλας του, τα βήματα που εκτελεί είναι ασυσχέτιστα μεταξύ τους και προς τυχαία διεύθυνση. Αυτά τα δυο χαρακτηριστικά θα ορίσουν το απλούστερο πρότυπο που θα μελετήσουμε. Τα πρότυπα αυτά και οι γενικεύσεις τους έχουν άμεση σχέση με τη συμπεριφορά φυσικών συστημάτων (κίνηση Brown, διάχυση, κίνηση impurities στο πλέγμα, ιδιότητες μακρών ελαστικών μακρομορίων σε μακροσκοπικές αποστάσεις, στοχαστικές διαδικασίες). Στη φυσική στοιχειωδών σωματιδίων παρουσιάζονται στη γεωμετρία των τυπικών διαδρομών των μποζονίων στα ολοκληρώματα διαδρομών του Feynman. Οι έννοιες που αναπτύσσονται από τη μελέτη του τυχαίου περιπατητή γενικεύονται στη θεωρία των τυχαίων επιφανειών που σχετίζονται με τη δισδιάστατη κβαντική βαρύτητα και τη θεωρία χορδών [43].

Η γεωμετρία μιας τυπικής διαδρομής του απλού τυχαίου περιπατητή παύει να είναι κλασική και αυτό μπορεί να φανεί καθαρά από δύο μη κλασικές ιδιότητες της. Το μήκος της διαδρομής που διανύει ο περιπατητής είναι ανάλογος της **τετραγωνικής ρίζας** του χρόνου, με αποτέλεσμα να μην ισχύει η κλασική σχέση  $r = vt$ . Ο δεύτερος λόγος είναι ότι η γεωμετρία του χώρου που πατάει ο περιπατητής (με τη στατιστική έννοια) έχει δομή fractal με διάσταση μεγαλύτερη της μονάδας<sup>1</sup>, δηλαδή δεν είναι ένα απλό μονοδιάστατο μονοπάτι. Παρόμοια φαινόμενα παρουσιάζονται στις τυχαίες επιφάνειες και κβαντικές θεωρίες πεδίου όπου η απόκλιση από την κλασική συμπεριφορά μπορεί να γίνει κατανοητή με κατάλληλες γενικεύσεις των παραπάνω ιδεών. Για περαιτέρω μελέτη παραπέμπουμε στα συγγράμματα [7, 42, 43, 44].

---

<sup>1</sup>Για την ακρίβεια η διάσταση Hausdorff  $d_H = 2$ .

Για να προσομοιώσουμε ένα τέτοιο σύστημα στον υπολογιστή είναι αναγκαία η χρήση γεννητριών τυχαίων αριθμών. Αυτές, τις περισσότερες φορές, είναι αλγόριθμοι που παράγουν μια ακολουθία από ψευδοτυχαίους αριθμούς οι οποίοι κατανέμονται ομοιόμορφα. Από αυτούς είναι δυνατόν να παράξουμε πιο πολύπλοκες κατανομές πιθανοτήτων και έτσι να προσομοιώσουμε συστήματα που έχουν στοχαστική συμπεριφορά. Στο κεφάλαιο αυτό θα μελετήσουμε μερικές απλές τέτοιες γεννήτριες και τις βασικές τους ιδιότητες, καθώς και θα εξασκηθούμε στη χρήση ποιοτικών γεννητριών που είναι μεταφερόμενες (portable) σε οποιοδήποτε υπολογιστικό περιβάλλον.

## 11.1 (Ψευδο)Τυχαίοι Αριθμοί

Η παραγωγή ψευδοτυχαίων αριθμών είναι στην καρδιά της προσομοίωσης Μόντε Κάρλο. Η παραγωγή των ψευδοτυχαίων αριθμών είναι κατά κανόνα ντετερμινιστική: Στη γεννήτρια τυχαίων αριθμών (δηλ. στον αλγόριθμο) δίνονται κάποιες αρχικές συνθήκες από τις οποίες η παραγωγή των αριθμών αφήνεται να εξελιχθεί στο “χρόνο”. Ο επόμενος αριθμός προκαθορίζεται από την κατάσταση της γεννήτριας εξ’ ου και η ντετερμινιστική εξέλιξη. Ίδιες αρχικές συνθήκες δίνουν την ίδια ακριβώς ακολουθία αριθμών. Η εξέλιξη αυτή όμως είναι χαοτική. Ελάχιστα διαφορετικές αρχικές συνθήκες φτιάχνουν ακολουθίες που αποκλίνουν εκθετικά στο χρόνο ή μία από την άλλη. Για παρόμοιο λόγο, ο αριθμός που παράγεται σε κάθε βήμα θεωρείται πως είναι ασυσχέτιστος από τον προηγούμενο. Εδώ βρίσκεται και το αδύνατο σημείο που προκαλεί τα δυσκολότερα προβλήματα, γιατί στις προβληματικές γεννήτριες βρίσκονται λεπτοί συσχετισμοί που είναι δύσκολο να προσδιοριστούν. Πραγματικά τυχαίοι αριθμοί (χρήσιμοι στην κρυπτογραφία) μπορούν να παραχθούν από ειδικές συσκευές που βασίζονται στο χρόνο διάσπασης ραδιενεργών υλικών ή τον ατμοσφαιρικό θόρυβο<sup>2</sup>. Σχεδόν τυχαίοι αριθμοί κρυπτογραφικής ποιότητας παράγονται από τα ειδικά αρχεία /dev/random και /dev/urandom, τα οποία διαβάζουν bits από μια δεξαμενή εντροπίας που φτιάχνεται από διάφορους εξωτερικούς παράγοντες (θερμοκρασία υπολογιστή, θόρυβο από οδηγούς συσκευών κλπ) που είναι διαθέσιμα σε αρκετά λειτουργικά συστήματα και θα δούμε τη χρήση τους σε επόμενη παράγραφο.

Οι πιο δημοφιλείς γεννήτριες, λόγω της απλότητάς τους, είναι οι γεννήτριες modulo (D.H. Lehmer, 1951). Αυτές χρειάζονται μόνο έναν

<sup>2</sup>Υπάρχουν υπηρεσίες στο διαδίκτυο που παρέχουν πραγματικά τυχαίους αριθμούς όπως οι [www.random.org](http://www.random.org), [www.fourmilab.ch/hotbits/](http://www.fourmilab.ch/hotbits/) και άλλες.

αριθμό  $x_0$  (seed) ως αρχική συνθήκη και από τον αριθμό  $x_{i-1}$  παράγουν τον  $x_i$  από τη σχέση

$$x_i = a x_{i-1} + c \pmod{m} \quad (11.1)$$

για κατάλληλα επιλεγμένες τιμές των  $a$ ,  $c$  και  $m$ . Η κατάσταση της γεννήτριας προσδιορίζεται πλήρως από ένα μόνο αριθμό, την τρέχουσα τιμή του  $x_i$ . Υπάρχει βιβλιογραφία σχετικά με τη σωστή επιλογή των  $a$ ,  $c$  και  $m$ , εδώ ας σημειώσουμε πως αυτή είναι διαφορετική για γλώσσα C ή FORTRAN και για συστήματα που είναι 32-bit ή 64-bit. Για λεπτομέρειες παραπέμπουμε στο ειδικό κεφάλαιο των Numerical Recipes [8].

Η τιμή του  $m$  καθορίζει την μέγιστη περίοδο της γεννήτριας. Είναι σαφές πως αν σε κάποια στιγμή της ακολουθίας πάρω τον ίδιο αριθμό που πήρα πριν από  $k$  βήματα, η γεννήτρια θα αρχίσει να παράγει ακριβώς την ίδια ακολουθία και αυτό θα συνεχίζεται περιοδικά. Αφού έχω  $m$  δυνατούς διαφορετικούς αριθμούς, η περίοδος είναι το πολύ  $m$ . Για μια κακή επιλογή των παραμέτρων  $a$ ,  $c$  και  $m$  η περίοδος θα είναι πολύ μικρότερη. Η τιμή του  $m$  όμως δεν μπορεί να είναι αυθαίρετα μεγάλη, αφού σε οποιοδήποτε υπολογιστή υπάρχει ένας μέγιστος ακέραιος που καθορίζεται από τον αριθμό των bits που χρησιμοποιεί να τον αναπαραστήσει. Για ακεραίους 4-bytes (32-bits) χωρίς πρόσημο ο μέγιστος αριθμός είναι  $2^{32} - 1$ , ενώ αν έχουν πρόσημο  $2^{31} - 1$ . Καλή επιλογή των  $a$ ,  $c$  και  $m$  μπορεί να αποδειχθεί (βλ. Knuth [45]) ότι δίνει ακολουθία που είναι αναδιάταξη  $\{\pi_1, \pi_2, \dots, \pi_m\}$  των αριθμών  $1, 2, \dots, m$ . Μια τέτοια επιλογή δίνει αρκετά καλούς τυχαίους αριθμούς για απλές εφαρμογές, αλλά για σοβαρούς υπολογισμούς θα πρέπει να ανατρέξουμε σε μια προσεκτικά επιλεγμένη γεννήτρια τυχαίων αριθμών. Σε αυτές, η κατάσταση της γεννήτριας είναι πιο πολύπλοκη και καθορίζεται από περισσότερους από έναν ακέραιους αριθμούς. Για κώδικα ο αναγνώστης μπορεί να ανατρέξει στις αναφορές [4], [5], [8], [46]. Για portable προγράμματα προτείνονται οι γεννήτριες RANLUX [46] και η γεννήτρια των Marsaglia, Zaman and Tsang. Πρόγραμμα Fortran για την πρώτη θα βρείτε στο συνοδευτικό λογισμικό, ενώ για την τελευταία μπορεί να βρεθεί στο βιβλίο/site του Berg [5].

Για να γίνει κατανοητή η χρήση των γεννητριών, αλλά και να αναδειχθεί ένα βασικό πρόβλημα των modulo γεννητριών παραθέτουμε παρακάτω τις συναρτήσεις `naiveran()` και `drandom()`. Η πρώτη είναι προβληματική και θα μελετήσουμε τους συσχετισμούς που κρύβονται στις ακολουθίες τυχαίων αριθμών που παράγει, καθώς και την επίδρασή τους στις προσομοιώσεις του τυχαίου περιπατητή. Η δεύτερη μπορεί να χρησιμοποιηθεί και για μη τετριμμένες εφαρμογές, όπως οι προσομοιώσεις

του τυχαίου περιπατητή και του πρότυπου Ising που θα κάνουμε σε επόμενες παραγράφους.

Η `naiveran()` είναι απλή εφαρμογή της σχέσης (11.1) με  $a = 1266$ ,  $c = 0$  και  $m = 2^{17}$ :

```
!=====
!File: naiveran.f90
!Program to demonstrate the usage of a modulo
!generator with a bad choice of constants
!resulting in strong pair correlations between
!generated numbers
!=====
real(8) function naiveran()
  implicit none
  integer          :: iran=13337
  common /naiveranpar/ iran
  integer,parameter :: m = 131072 ! equal to 2**17
  integer,parameter :: a = 1277

  iran = a*iran
  iran = MOD(iran,m)

  naiveran = iran/DBLE(m)
end function naiveran
```

Η `drandom()` είναι πάλι εφαρμογή της ίδιας σχέσης, αλλά με  $a = 7^5$ ,  $c = 0$  και  $m = 2^{31} - 1$ . Η επιλογή έγινε από τους Lewis, Goodman και Miller (1969) και δίνει μια γεννήτρια που έχει περάσει από πολλά τεστ και, ακόμα πιο σημαντικό, έχει δοκιμαστεί με επιτυχία πολλές φορές. Το πρόβλημα που παρουσιάζεται είναι ότι πολλαπλασιασμός του  $x_{i-1}$  με το  $a$  θα δώσει πολλές φορές αριθμό που είναι εκτός του διαστήματος που καλύπτουν οι ακέραιοι τεσσάρων bytes και θα έχουμε integer overflow. Για να έχουμε κώδικα που να μεταφέρεται παντού, αλλά και για να είναι γρήγορος, είναι θεμιτό να παραμείνουμε στον χώρο των  $2^{31} - 1$ , θετικών 32-μπιτων (4 byte) ακεραίων με πρόσημο. Η πρόταση του Schrage είναι να χρησιμοποιήσουμε τη σχέση

$$(ax_{i-1}) \bmod m = \begin{cases} a(x_{i-1} \bmod q) - r \left\lfloor \frac{x_{i-1}}{q} \right\rfloor & \text{αν είναι} \geq 0 \\ a(x_{i-1} \bmod q) - r \left\lfloor \frac{x_{i-1}}{q} \right\rfloor + m & \text{αν είναι} < 0 \end{cases} \quad (11.2)$$

όπου  $m = aq + r$ ,  $q = \lfloor m/a \rfloor$  και  $r = m \bmod a$ . Μπορεί ναδειχθεί ότι αν  $r < q$  και αν  $0 < x_{i-1} < m - 1$ , τότε  $0 \leq a(x_{i-1} \bmod q) \leq m - 1$ ,

$0 \leq r[x_{i-1}/q] \leq m-1$  και ότι ισχύει η (11.2). Η περίοδος της γεννήτριας είναι  $2^{31} - 2 \approx 2 \times 10^9$ . Η ισχύς των παραπάνω ισχυρισμών αφήνεται ως αριθμητική άσκηση για τον αναγνώστη.

```

=====
! File: drandom.f90
! Implementation of the Schrage algorithm for a
! portable modulo generator for 32 bit signed integers
! (from numerical recipes)
!
! returns uniformly distributed pseudorandom numbers
! 0.0 < x < 1.0 (0 and 1 excluded)
=====
real(8) function drandom()
  implicit none
  integer, parameter :: a = 16807      ! a = 7**5
  integer, parameter :: m = 2147483647 ! m = a*q+r = 2**31-1
  integer, parameter :: q = 127773     ! q = [m/a]
  integer, parameter :: r = 2836       ! r = MOD(m,a)
  real(8), parameter :: f = (1.0D0/m)
  integer :: p
  integer :: seed
  real(8) :: dr
  common /randoms/seed

101 continue
  p      = seed/q      ! = [seed/q]
  seed   = a*(seed- q*p) - r*p ! = a*MOD(seed,q)-r*[seed/q]
  if(seed .lt. 0) seed = seed + m
  dr     = f*seed
  if( dr .le. 0.0D0 .or. dr .ge. 1.0D0) goto 101
  drandom = dr
end function drandom

```

Η γραμμή ελέγχου των τιμών της γεννήτριας είναι αναγκαία μόνο για την τιμή 0 που παρουσιάζεται μια φορά στην ακολουθία και βάζει ένα επιπλέον φορτίο χρόνου της τάξης του 10 – 20% εξαρτώμενο από το μεταγλωττιστή. Αν δε σας ενδιαφέρει, μπορείτε να το βγάλετε. Επίσης, προσέξτε πως ο αριθμός seed είναι σε common block και μπορεί να αλλάξει από το καλών πρόγραμμα.

Παρακάτω, παραθέτουμε ένα απλό πρόγραμμα χρήσης των παραπάνω γεννητριών το οποίο και θα αναδείξει το πρόβλημα συσχετισμού ζευγαριών των τιμών της `naiveran()`. Θα παράξουμε ζεύγη ακέραιων τιμών  $(i, j)$  με  $0 \leq i, j < 10000$  τα οποία στη συνέχεια θα απεικονίσουμε στο επίπεδο. Αυτό γίνεται παίρνοντας το ακέραιο μέρος των αριθμών

$Lu$  με  $L = 10000$  και  $0 \leq u < 1$  ο τυχαίος αριθμός που δίνει η κάθε γεννήτρια:

```

=====
!Program that produces N random points (i,j) with
!0<= i,j < 10000. Simple qualitative test of serial
!correlations of random number generators on the plane.
!
!compile:
!gfortran correlations2ran.f90 naiveran.f90 drandom.f90
=====
program correlations2
  implicit none
  integer,parameter :: L = 10000
  integer            :: i,N
  character(10)      :: arg
  real(8)            :: naiveran,drandom
  integer            :: seed
  common /randoms/   seed
!Read the number of points from first command argument
if(IARGC() .EQ. 1)then
  call GETARG(1,arg); read(arg,*)N !convert string->integer
else !default value, if no N given by user:
  N=1000
endif
seed = 348325
do i=1,N
  print *,INT(L * naiveran()),INT(L * naiveran())
! print *,INT(L * drandom ()),INT(L * drandom ())
enddo

end program correlations2

```

Το πρόγραμμα βρίσκεται στο αρχείο correlations2ran.f90. Για να δοκιμάσουμε τη naiveran() το μεταγλωττίζουμε

```
> gfortran correlations2ran.f90 naiveran.f90 -o naiveran
```

ενώ για να δοκιμάσουμε την drandom(), μεταβάλλουμε τις γραμμές print ως εξής

```

! print *,INT(L * naiveran()),INT(L * naiveran())
print *,INT(L * drandom ()),INT(L * drandom ())

```

και ξαναμεταγλωττίζουμε



```
> gfortran correlations2ran.f90 drandom.f90 -o drandom
```

παράγοντας έτσι τα δύο εκτελέσιμα αρχεία `naiveran` και `drandom`. Για να δούμε τα αποτελέσματα τρέχουμε με τις εντολές

```
> ./naiveran 100000 > naiveran.out
> ./drandom 100000 > drandom.out
> gnuplot
gnuplot> plot "naiveran.out" using 1:2 with dots
gnuplot> plot "drandom.out" using 1:2 with dots
```

με τις οποίες παράγουμε  $10^5$  σημεία τα οποία, στη συνέχεια, βλέπουμε με το `gnuplot`.

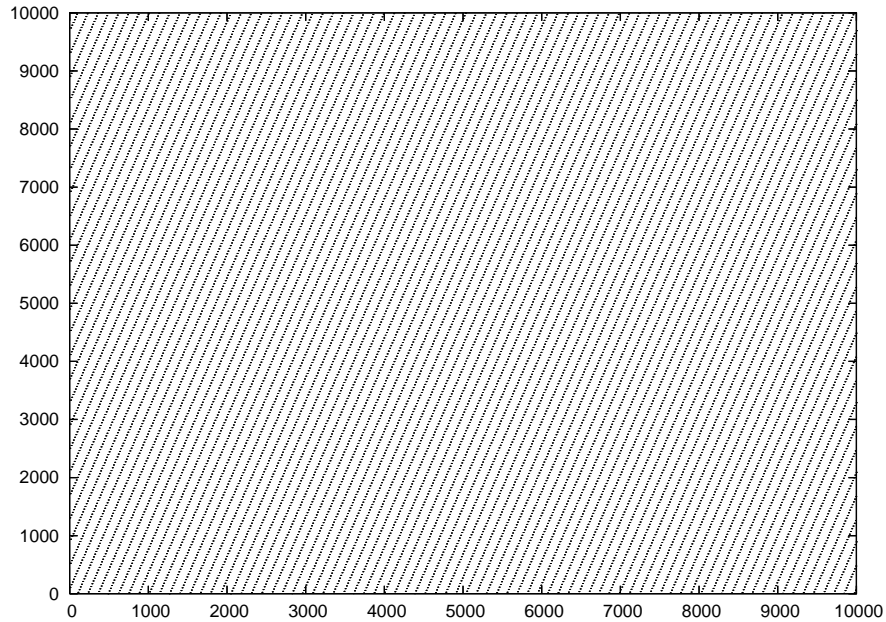
Οι παραπάνω εντολές παράγουν τα σχήματα 11.1 και 11.2 στα οποία βλέπουμε καθαρά το συσχετισμό των ζευγών που παράγονται από την `naiveran()`. Τα σχήματα βέβαια δίνουν αρκετά ποιοτική αναπαράσταση. Προσεκτική μελέτη της `drandom()` δείχνει πως τα σημεία  $(u_i, u_{i+1})$  που παράγει δεν περνάνε το τεστ  $\chi^2$ , όταν έχουμε περισσότερα από  $10^7$  σημεία, που είναι πολύ λιγότερα από την περίοδο της γεννήτριας. Για να αποφύγουμε τέτοια προβλήματα έχουν προταθεί πολλές λύσεις, οι πιο απλές “ανακατεύουν” τα αποτελέσματα, ώστε να εξαφανίζουν τους σειριακούς συσχετισμούς χαμηλής τάξης. Τέτοιες γεννήτριες θα συζητήσουμε στην επόμενη παράγραφο. Η ομοιόμορφη κατανομή των τυχαίων αριθμών μπορεί ναδειχθεί γραφικά φτιάχνοντας το ιστόγραμμα της σχετικής συχνότητας των τυχαίων αριθμών. Για να κάνουμε ιστογράμματα χρησιμοποιούμε το script `histogram` γραμμένο σε `awk`<sup>3</sup> ως εξής:

```
> histogram -v f=0.01 drandom.out > drandom.hst
> gnuplot
gnuplot> plot "drandom.hst" using 1:3 with histeps
gnuplot> plot [:][0:] "drandom.hst" using 1:3 with histeps
```

Με την εντολή `histogram -v f=0.01` φτιάχνεται ιστόγραμμα των δεδομένων στο αρχείο `drandom.out`, έτσι ώστε το πλάτος των περιοχών τιμών (bin width) να είναι  $1/0.01 = 100$ . Γενικά, το αντίστροφο του αριθμού μετά το `-f` ορίζει το πλάτος των περιοχών τιμών του ιστογράμματος.

Τα αποτελέσματα φαίνονται στα σχήματα 11.3 και 11.4. Στη συνέχεια, μελετούμε τη διασπορά των μετρήσεων μας, οι οποίες είναι εμφανείς ήδη από το σχήμα 11.3. Η διασπορά αυτή μειώνεται με το μέγεθος

<sup>3</sup>Βλ. συνοδευτικό λογισμικό στον κατάλογο `Tools`. Δώστε την εντολή `histogram --h` για να τυπωθούν στο τερματικό σας οδηγίες χρήσης.



Σχήμα 11.1: Τα ζεύγη ψευδοτυχαίων αριθμών από τη συνάρτηση `naiveran()`. Φαίνεται καθαρά ο συσχετισμός των ζευγών που παράγονται, αφού τα σημεία κάθονται πάνω σε εμφανές διακριτό πλέγμα.

του δείγματος των τυχαίων αριθμών που συλλέγουμε. Αυτό φαίνεται στο ιστόγραμμα του σχήματος 11.5. Για να δούμε ποσοτικά την εξάρτηση των διακυμάνσεων από το μέγεθος του δείγματος  $n$ , υπολογίζουμε την ποσότητα ( $x_i$  είναι η ακολουθία των τυχαίων αριθμών)

$$\sigma = \sqrt{\frac{1}{n-1} \left( \frac{1}{n} \sum_{i=1}^n x_i^2 - \left( \frac{1}{n} \sum_{i=1}^n x_i \right)^2 \right)}, \quad (11.3)$$

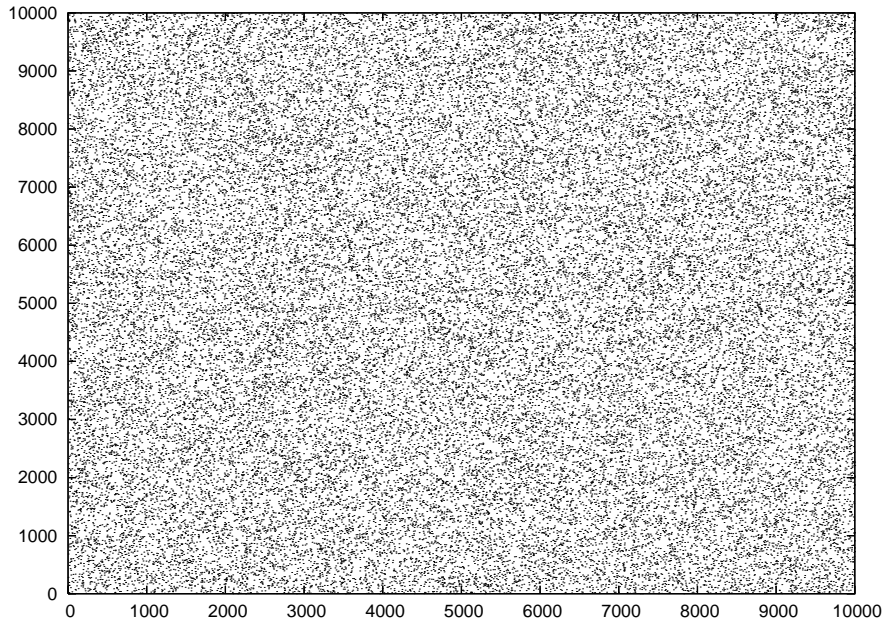
την οποία αποδίδουμε γραφικά στο σχήμα 11.6. Η σχέση

$$\ln \sigma \sim \frac{1}{2} \ln(n), \quad (11.4)$$

όπως φαίνεται από την ευθεία που προσαρμόζουμε στα σημεία, δείχνει ότι

$$\sigma \sim \frac{1}{\sqrt{n}}. \quad (11.5)$$

Θα κλείσουμε το κεφάλαιο αυτό αναφέροντας σύντομα την παραγωγή ψευδοτυχαίων αριθμών με κατανομή διαφορετική από την ομοιό-



Σχήμα 11.2: Τα ζεύγη ψευδοτυχαίων αριθμών από τη συνάρτηση `drandom()`. Φαίνεται ότι τα σημεία αυτά κατανέμονται τυχαία σε σχέση με αυτά της `naiveran()`.

μορφή. Για μια τυχαία κατανομή με συνάρτηση πυκνότητας πιθανότητας  $f(x)$  μπορούμε να πάρουμε ψευδοτυχαίους αριθμούς που ακολουθούν την  $f(x)$  από ομοιόμορφα κατανεμημένους αριθμούς στο διάστημα  $(0, 1)$ . Έστω η συνάρτηση κατανομής

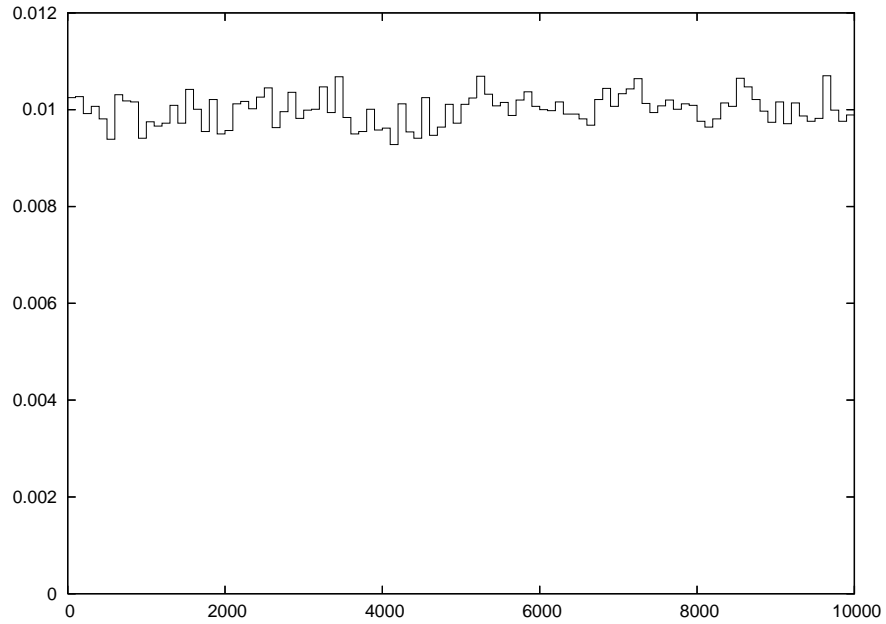
$$0 \leq u = F(x) = \int_{-\infty}^x f(x') dx' \leq 1, \quad (11.6)$$

που δεν είναι άλλο από το εμβαδόν κάτω από την  $f(x)$  στο  $(-\infty, x]$  και ισούται με την πιθανότητα  $P(x' < x)$ . Αν επιλέξουμε το  $u$  ομοιόμορφα κατανεμημένο στο διάστημα  $(0, 1)$ , τότε θα έχουμε  $P(u' < u) = u$ . Τότε το  $x = F^{-1}(u)$  είναι τέτοιο, ώστε  $P(x' < x) = u = F(x)$ , άρα κατανέμεται σύμφωνα με την  $f(x)$ . Οπότε αν  $u_i$  είναι ομοιόμορφα κατανεμημένοι ψευδοτυχαίοι αριθμοί, τότε οι

$$x_i = F^{-1}(u_i) \quad (11.7)$$

είναι μια ακολουθία ψευδοτυχαίων αριθμών που κατανέμεται σύμφωνα με την  $f(x)$ . Για παράδειγμα, θεωρούμε την κατανομή κατά Cauchy

$$f(x) = \frac{1}{\pi} \frac{c}{c^2 + x^2} \quad c > 0. \quad (11.8)$$



Σχήμα 11.3: Η σχετική συχνότητα κατανομής των ψευδοτυχαίων αριθμών που παράγονται από την συνάρτηση `drandom()`. Φαίνεται η ομοιόμορφη κατανομή και η διασπορά γύρω από τη μέση τιμή.

Τότε

$$F(x) = \int_{-\infty}^x f(x') dx' = \frac{1}{2} + \frac{1}{\pi} \tan^{-1} \left( \frac{x}{c} \right). \quad (11.9)$$

Σύμφωνα με τα παραπάνω η γεννήτρια τυχαίων αριθμών θα δίνεται από τη σχέση

$$x_i = c \tan(\pi u_i - \pi/2) \quad (11.10)$$

ή ισοδύναμα (για πιο γρήγορο πρόγραμμα)

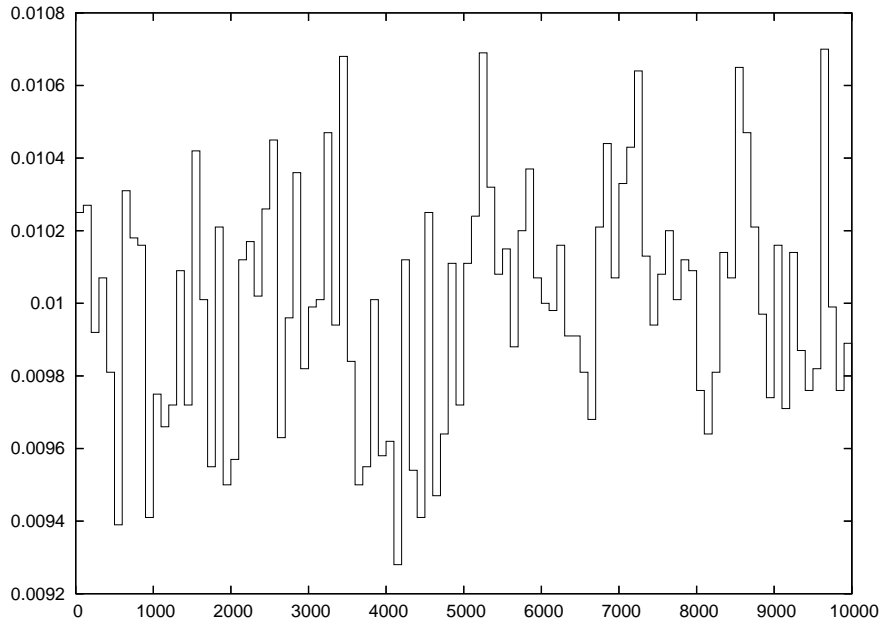
$$x_i = c \tan(2\pi u_i). \quad (11.11)$$

Περισσότερες εφαρμογές έχει η γεννήτρια Gaussian τυχαίων αριθμών. Η κατανομή αυτή δίνεται από τη συνάρτηση πυκνότητας πιθανότητας

$$g(x) = \frac{1}{\sqrt{2\pi} \sigma} e^{-x^2/(2\sigma^2)} \quad (11.12)$$

Η συνάρτηση κατανομής είναι η

$$G(x) = \int_{-\infty}^x g(x') dx' = \frac{1}{2} + \frac{1}{2} \operatorname{erf} \left( \frac{x}{\sqrt{2} \sigma} \right) \quad (11.13)$$



Σχήμα 11.4: Το προηγούμενο σχήμα σε μεγαλύτερη κλίμακα για να φανεί καθαρότερα η διασπορά.

όπου η  $\text{erf}(x) = \int_{-\infty}^x \exp\{-(x')^2\} dx'$  είναι η error function η οποία μπορεί να υπολογιστεί αριθμητικά, καθώς και η αντίστροφή της. Αυτό θα οδηγούσε σε πολύ αργό αλγόριθμο για τη γεννήτριά μας, οπότε χρησιμοποιούμε την πυκνότητα πιθανότητας  $\rho(x, y)$  δύο τυχαίων μεταβλητών  $x, y$

$$\rho(x, y) dx dy = \frac{1}{\sqrt{2\pi} \sigma} e^{-x^2/(2\sigma^2)} \frac{1}{\sqrt{2\pi} \sigma} e^{-y^2/(2\sigma^2)} dx dy = \frac{1}{2\pi\sigma^2} e^{-r^2/(2\sigma^2)} r dr d\phi \quad (11.14)$$

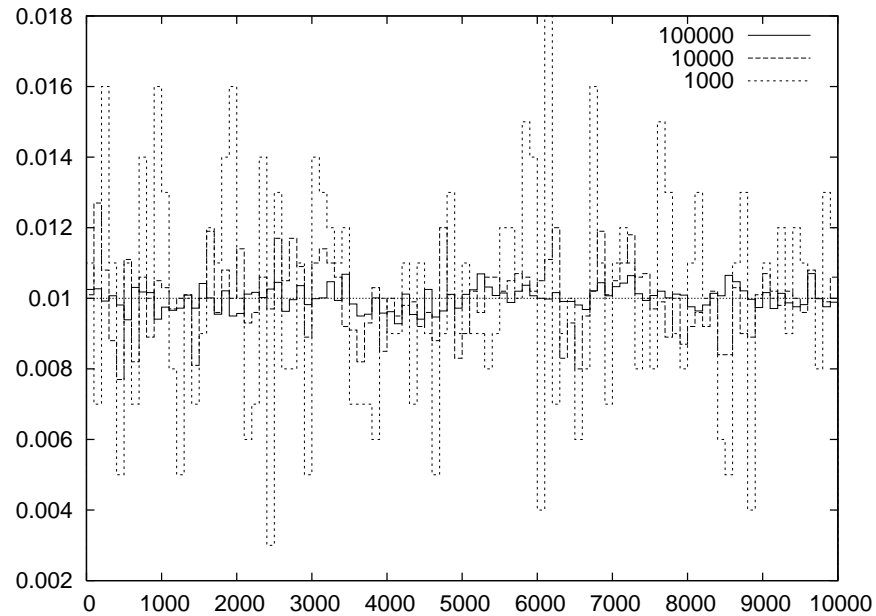
όπου  $x = r \cos \phi$ ,  $y = r \sin \phi$ . Τότε, έχουμε ότι

$$u = G(r) = \int_0^r \int_0^{2\pi} dr d\phi \rho(r, \phi) = 1 - e^{-r^2/(2\sigma^2)} \quad (11.15)$$

η οποία όταν αντιστραφεί θα μας δώσει

$$r = \sigma \sqrt{-2 \ln(1 - u)}. \quad (11.16)$$

Οπότε, αρκεί να παράγουμε ακολουθία  $\{u_i\}$  ομοιόμορφα κατανεμημέ-



Σχήμα 11.5: Η σχετική συχνότητα κατανομής των ψευδοτυχαίων αριθμών που παράγονται από την συνάρτηση `drandom()` σαν συνάρτηση του μεγέθους του δείγματος των τυχαίων αριθμών  $n$  για  $n = 1000, 10000, 100000$ .

νων τυχαίων αριθμών και να πάρουμε:

$$r_i = \sigma \sqrt{-2 \ln(u_i)} \quad (11.17)$$

$$\phi_i = 2\pi u_{i+1} \quad (11.18)$$

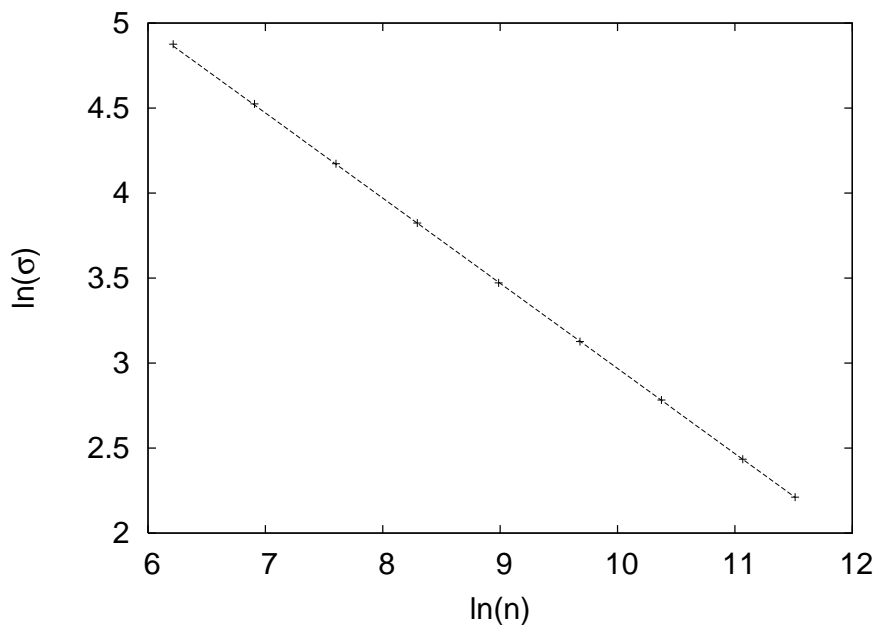
$$x_i = r_i \cos \phi_i \quad (11.19)$$

$$x_{i+1} = r_i \sin \phi_i, \quad (11.20)$$

οπότε προκύπτει η ακολουθία  $\{x_i\}$  που είναι Gaussian τυχαίοι αριθμοί<sup>4</sup>. Ο προγραμματισμός της σχετικής συνάρτησης για  $\sigma = 1$  δίνεται παρακάτω:

```
!=====
!Function to produce random numbers distributed
!according to the gaussian distribution
!g(x) = 1/(sigma*sqrt(2*pi))*exp(-x**2/(2*sigma**2))
!=====
real(8) function gaussran()
  implicit none
```

<sup>4</sup>Μπορεί ναδειχθεί ότι τα  $x_i, x_{i+1}$  είναι στατιστικά ανεξάρτητα.



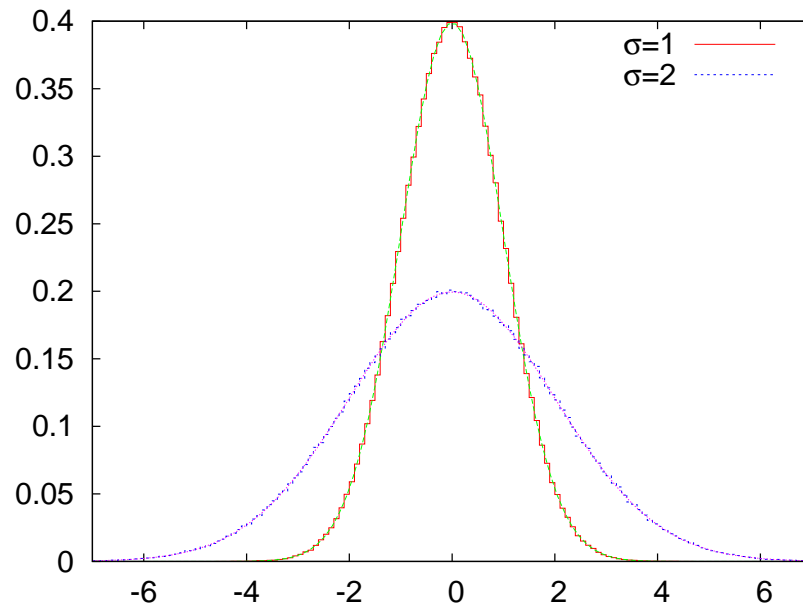
Σχήμα 11.6: Η εξάρτηση της διασποράς (11.3) από το  $n$  για τη συνάρτηση `drandom()`.

```

real(8),parameter :: sigma = 1.0D0
real(8)           :: r,phi
logical ,save     :: new    = .TRUE.
real(8),save      :: x
real(8),parameter :: PI2    = 6.28318530717958648D0
real(8)           :: drandom
if(new)then
  new      = .FALSE.
  r        = drandom()
  phi      = PI2*drandom()
  r        = sigma*sqrt(-2.0D0*log(r))
  x        = r*cos(phi)
  gaussran = r*sin(phi)
else
  new      = .TRUE.
  gaussran = x
endif
end function gaussran

```

Το αποτέλεσμα φαίνεται στο σχήμα 11.7. Προσέξτε το attribute `SAVE` στις μεταβλητές `new` και `x`. Αυτό σημαίνει πως η τιμή τους διατηρείται ανάμεσα στα διαφορετικά καλέσματα της συνάρτησης `drandom`. Αυτό γίνεται γιατί σε κάθε εφαρμογή της (11.17) παράγουμε 2 τυχαίους αριθμούς, ενώ η συνάρτηση επιστρέφει τον ένα. Για το λόγο αυτό η συνάρ-



Σχήμα 11.7: Η κατανομή των ψευδοτυχαίων αριθμών που παράγονται από την συνάρτηση `gaussran()` για  $\sigma = 1$  και  $\sigma = 2$ . Το ιστόγραμμα υπερτίθεται της γραφικής παράστασης της (11.12).

τηση διατηρεί μνήμη το αν θα πρέπει να παράγει καινούργιο ζευγάρι (η “σημαία” (flag) `new`) και αν όχι, να επιστρέψει την τιμή ενός από τους τυχαίους αριθμούς (μεταβλητή `x`). Η ανάλυση των αποτελεσμάτων αφήνεται για άσκηση.

## 11.2 Χρήση Γεννητριών Ψευδοτυχαίων Αριθμών

Η χρήση της `drandom()` μπορεί να εξυπηρετήσει σχεδόν όλες μας τις ανάγκες για την παραγωγή (ψευδο)τυχαίων αριθμών στο βιβλίο αυτό. Πολλές φορές είναι αναγκαίο να χρησιμοποιήσουμε γεννήτριες που δίνουν καλύτερης ποιότητας τυχαίους αριθμούς και στην παράγραφο αυτή θα δώσουμε οδηγίες για το πώς να χρησιμοποιήσετε δύο άλλες, καλής ποιότητας και διαθέσιμες σχεδόν παντού, γεννήτριες.

Η πρώτη είναι `intrinsic` στη γλώσσα Fortran και είναι η υπορουτίνα `RANDOM_NUMBER`. Στο περιβάλλον Fortran που δίνει η `gfortran`<sup>5</sup> η

<sup>5</sup>Άλλα περιβάλλοντα Fortran μπορεί να χρησιμοποιούν άλλους αλγόριθμους, διαβά-



RANDOM\_NUMBER αυτή χρησιμοποιεί τον αλγόριθμο “multiply-with-carry” του George Marsaglia σε συνδυασμό με γεννήτρια modulo και “shift-register” με περίοδο μεγαλύτερη από  $2^{123} \approx 10^{37}$ . Η κατάσταση της γεννήτριας είναι πιο πολύπλοκη από αυτή της drandom(), η οποία καθορίζεται από ένα μόνο αριθμό, οπότε προκειμένου να τη χρησιμοποιήσουμε πρέπει να μάθουμε

- πώς να την ξεκινάμε από μια νέα κατάσταση
- πώς να σώζουμε σε ένα αρχείο την τρέχουσα κατάσταση
- πώς να διαβάζουμε και να ξεκινάμε τη γεννήτρια από μια παλιά κατάσταση
- και φυσικά, πώς να παίρνουμε τους τυχαίους αριθμούς.

Ειδικά το πώς να σώζουμε την τρέχουσα κατάσταση μιας γεννήτριας είναι πολύ σημαντικό, όταν έχουμε μια εργασία την οποία εκτελούμε χωρίζοντάς την σε μέρη, συνήθως γιατί θέλουμε κάθε μέρος της εργασίας να εκτελείται μέσα σε συγκεκριμένο χρονικό διάστημα. Στην περίπτωση αυτή, θέλουμε η ακολουθία των τυχαίων αριθμών που θα πάρουμε όταν ξεκινάει ένα νέο μέρος της εργασίας να είναι ακριβώς από το σημείο που την είχαμε αφήσει στο προηγούμενο. Η διαδικασία αυτή λέγεται checkpointing.

Το ξεκίνημα από μια νέα κατάσταση λέγεται seeding. Το seeding για τη RANDOM\_NUMBER γίνεται από ένα απροσδιόριστο αριθμό από NSEEDS ακέραιους. Για να μάθουμε ποιος είναι αυτός ο αριθμός, πρέπει να καλέσουμε τη ρουτίνα RANDOM\_SEED(size = NSEEDS) η οποία με το συγκεκριμένο όρισμα size = NSEEDS θα μας δώσει τον αριθμό των seeds, NSEEDS. Στη συνέχεια, πρέπει να ορίσουμε τις integer τιμές σε ένα array μεγέθους NSEEDS, έστω το seeds(NSEEDS) και να καλέσουμε ξανά την RANDOM\_SEED(PUT = seeds) με το όρισμα PUT = seeds, η οποία θα αποθηκεύσει την κατάλληλη πληροφορία από το array seeds. Αυτό εμείς θα το κάνουμε χρησιμοποιώντας ένα μόνο integer αριθμό seed. Ο απαραίτητος κώδικας για τα παραπάνω είναι:

```
integer          :: NSEEDS
integer , allocatable :: seeds(:)
integer          :: seed
!
```

στη τεκμηρίωση του εκάστοτε μεταγλωττιστή. Για το λόγο αυτό ο αριθμός NSEEDS μπορεί να διαφέρει από περιβάλλον σε περιβάλλον.

```
seed = 47279823
call RANDOM_SEED(size = NSEEDS)
ALLOCATE(seeds(NSEEDS))
seeds = seed + 37 * (/ (i - 1, i = 1, NSEEDS) /)
call RANDOM_SEED(PUT = seeds)
```

Η τελευταία γραμμή<sup>6</sup> χρησιμοποιεί τις τιμές στο array seeds(1) ... seeds(NSEEDS) για να αρχικοποιήσει την κατάσταση της RANDOM\_NUMBER. Το σημαντικό είναι ότι για την ίδια τιμή του seed θα πάρουμε κάθε φορά την ίδια ακολουθία τυχαίων αριθμών.

Πολλές φορές θέλουμε να ξεκινήσουμε τη γεννήτρια των τυχαίων αριθμών από μια όσο το δυνατόν πιο τυχαία κατάσταση, έτσι ώστε κάθε φορά που τρέχουμε το πρόγραμμα να παίρνουμε μια διαφορετική ακολουθία τυχαίων αριθμών. Στα συστήματα τύπου Unix όπως το GNU/Linux μπορούμε να χρησιμοποιήσουμε δύο ειδικά αρχεία που δίνουν σχεδόν κρυπτογραφικής ποιότητας τυχαίους αριθμούς, τα /dev/random και /dev/urandom. Αυτά παράγουν τυχαία bits από την τρέχουσα κατάσταση του υπολογιστή και είναι πρακτικά αδύνατον να προβλέψει κανείς την ακολουθία bits που θα πάρει. Η /dev/urandom είναι προτιμότερη, γιατί η /dev/random σταματά να δουλεύει, όταν δεν έχει τελείως καινούργια τυχαία bits στη δεξαμενή της και μπορεί να χρειαστεί να περιμένουμε να μας δώσει τα bits που ζητάμε. Ο απαραίτητος κώδικας για seeding από την /dev/urandom είναι

```
open(unit=13, file='/dev/urandom', access='stream', &
  form='unformatted')
call RANDOM_SEED(size = NSEEDS)
ALLOCATE(seeds(NSEEDS))
read (13) seeds
close(13)
```

Το αρχείο /dev/urandom δίνει binary data και το ανοίγουμε ως unformatted, για τον ίδιο λόγο η εντολή read δεν έχει οδηγία format, αλλά μόνο τον αριθμό του unit. Η εντολή read διαβάζει όσα bits χρειάζονται για να γεμίσει όλο το array seeds. Μερικές φορές μπορεί να χρειαστεί να δουλέψουμε σε περιβάλλοντα που δε διαθέτουν το αρχείο /dev/urandom. Για να ξεκινήσουμε τότε τη γεννήτρια χρησιμοποιούμε το τρέχοντα χρόνο του λειτουργικού συστήματος και τον αριθμό της διεργασίας (process ID = pid) του προγράμματος, όταν τρέχει. Στο συνοδευτικό λογισμικό, στο αρχείο seed.f90, μπορείτε να δείτε πώς<sup>7</sup>.

<sup>6</sup>Η προτελευταία θέτει seeds(1)=seed, seeds(2)=seed+37, seeds(3)=seed+37\*2, ...

<sup>7</sup>Μπορείτε να χρησιμοποιήσετε και το λειτουργικό σύστημα για να περάσετε τυχαία

Στη συνέχεια, θα δούμε πώς να σώζουμε την τρέχουσα κατάσταση της γεννήτριας. Χρησιμοποιούμε πάλι την υπορουτίνα `RANDOM_SEED(get = seeds)` η οποία θα τοποθετήσει στο array `seeds` την απαραίτητη πληροφορία. Στη συνέχεια, γράφουμε τα `seeds` σε ένα αρχείο για να τα διαβάσουμε αργότερα. Ο απαραίτητος κώδικας είναι:

```
integer                :: NSEEDS
integer, allocatable  :: seeds(:)
!
call RANDOM_SEED(size = NSEEDS)
ALLOCATE(seeds(NSEEDS))
call RANDOM_SEED(GET = seeds)
open(unit=11, file='state')
write(11,*)seeds
```

Για να ξεκινήσουμε τη γεννήτρια από την κατάσταση που έχει σωθεί μέσα στο αρχείο `state`, απλά διαβάζουμε τα `seeds` από το αρχείο και ξεκινάμε τη γεννήτρια όπως πριν:

```
open(unit=11, file='state')
read(11,*)seeds
call RANDOM_SEED(PUT = seeds)
```

Για να παράγουμε τυχαίους αριθμούς, μπορούμε να χρησιμοποιήσουμε μεταβλητή `scalar` για να τους διαβάζουμε έναν-έναν ή ένα array το οποίο η `RANDOM_NUMBER` θα γεμίσει με τυχαίους αριθμούς. Η πρώτη μέθοδος έχει ένα ελαφρό overhead στην απόδοση και αν μπορούμε να χρησιμοποιήσουμε arrays το προτιμούμε. Ο απαραίτητος κώδικας για την παραγωγή τυχαίων αριθμών με την πρώτη μέθοδο είναι

```
real(8)                :: r
!
do icount = 1,10
  call RANDOM_NUMBER(r)
  print *,r
enddo
```

ενώ με τη δεύτερη:

`seeds` στο πρόγραμμά σας. Δοκιμάστε τις εντολές `set x = `< /dev/urandom tr -dc "[:digit:]" | head -c9 | awk 'printf "%d",$1'` ; echo $x` και `set x = `perl -e 'srand();print int(1000000000*rand());'` ; echo $x`. Χρησιμοποιήστε μετά την τιμή μεταβλητής `x` για `seed`.

```
integer ,parameter      :: NR=20
real(8),dimension(NR) :: randoms
call RANDOM_NUMBER(randoms)
print *,randoms
```

Τα παραπάνω συνοψίζονται στον κώδικα που βρίσκεται στο αρχείο test\_random\_number.f90 τον οποίο παραθέτουμε παρακάτω:

```
program use_random_number
  implicit none
  integer      :: NSEEDS
  integer ,allocatable :: seeds(:)
  integer      :: seed
  real(8)      :: r
  integer ,parameter      :: NR=20
  real(8),dimension(NR) :: randoms
  integer(8)      :: icount
  integer      :: i
  !-----
  !start from a new seed:
  seed = 47279823
  !get number of seeds for generator:
  call RANDOM_SEED(size = NSEEDS)
  ALLOCATE(seeds(NSEEDS))
  !fill in the rest of the seeds:
  seeds = seed + 37 * (/ (i - 1, i = 1, NSEEDS) /)
  !initialize the generator from the arrays seeds:
  call RANDOM_SEED(PUT = seeds)
  !-----
  !generate random numbers one by one:
  do icount = 1,10
    call random_number(r)
    print *,r
  enddo
  !generate random numbers in an array:
  call random_number(randoms)
  print '(1000G28.17)',randoms
  !-----
  !save state of random_number:
  open(unit=11,file='rannum.seed')
  call RANDOM_SEED(GET = seeds)
  write(11,'(5I20)')seeds
  close(11)
  !-----
  !generate some randoms:
  call random_number(randoms)
  print '(A,1000G28.17)', '#FIRST :',randoms
```

```

!-----
!read state of random_number:
open(unit=11,file='rannum.seed')
read(11,*)seeds
call RANDOM_SEED(PUT = seeds)
!-----
!generate same randoms:
call random_number(randoms)
print '(A,1000G28.17)', '#SECOND:',randoms
end program use_random_number

```

Μεταγλωττίστε και δείτε τα αποτελέσματα με τις εντολές

```

> gfortran test_random_number.f90 -o random_number
> ./random_number

```

Μια πολύ καλή γεννήτρια τυχαίων αριθμών προτάθηκε από τον Martin Lüscher [46] και το πρόγραμμα που τη χρησιμοποιεί λέγεται Ranlux. Εκτός από την άριστη ποιότητα τυχαίων αριθμών με περίοδο μεγαλύτερη από  $10^{171}$ , το μεγάλο πλεονέκτημα της RANLUX είναι ότι ο κώδικας θα τρέξει σε οποιαδήποτε πλατφόρμα διαθέτει Fortran, οπότε μπορείτε να τη χρησιμοποιήσετε σε όλα σας τα προγράμματα, παντού. Το πρόγραμμα, το οποίο θα βρείτε και στο συνοδευτικό λογισμικό με κάποιες μικρές μετατροπές ως προς τον αρχικό κώδικα, έχει γραφτεί από τον Fred James και μπορείτε να το κατεβάσετε στην αρχική του μορφή από τους συνδέσμους που δίνονται στη βιβλιογραφία [46]. Η γεννήτρια βασίζεται στον αλγόριθμο subtract-with-borrow των Marsaglia και Zaman [48], ο οποίος έχει πολύ μεγάλη περίοδο, αλλά αποτυγχάνει σε ορισμένα στατιστικά τεστ στα οποία υποβάλλεται. Βασιζόμενος στις χαοτικές ιδιότητες του προαναφερόμενου αλγόριθμου, ο Lüscher πρότεινε την εξαφάνιση των προβλημάτων αποδίδοντάς τα στους συσχετισμούς των τυχαίων αριθμών που βρίσκονται κοντά μεταξύ τους στην ακολουθία που παράγει ο αλγόριθμος.

Για να ξεκινήσετε τη γεννήτρια από ένα συγκεκριμένο σημείο που δίνεται από ένα seed που θα παρέχετε, χρησιμοποιείτε την υπορουτίνα RLUXGO και ο απαραίτητος κώδικας είναι

```

integer                                :: seed,ranlux_level
!-----
seed      = 58266273
ranlux_level = 2
call RLUXGO(ranlux_level,seed,0,0)

```

Προσέξτε την επιλογή της μεταβλητής `ranlux_level`: Αυτή καθορίζει την ποιότητα των τυχαίων αριθμών που παράγει η ρουτίνα και μπορεί να παίρνει τιμές 1, 2, 3 ή 4. Για τις ανάγκες στο βιβλίο αυτό, `ranlux_level=2` είναι αρκετό, `ranlux_level=3` είναι η προκαθορισμένη τιμή. Φυσικά, όσο μεγαλύτερη η τιμή της `ranlux_level`, τόσο μεγαλύτερος και ο χρόνος που χρειάζεται για την παραγωγή τυχαίων αριθμών (δείτε σχετική άσκηση).

Για να σώσουμε την κατάσταση που βρίσκεται η ρουτίνα, πρέπει να έχουμε ορίσει ένα integer array μεγέθους 25 και να καλέσουμε την υπορουτίνα `RLUXUT` για να αποθηκεύσει την κατάλληλη πληροφορία στο array αυτό. Στη συνέχεια, μπορούμε να σώσουμε το array σε ένα αρχείο για να το διαβάσουμε αργότερα. Ο απαραίτητος κώδικας για τη λειτουργία αυτή είναι:

```
integer , parameter      :: NSEEDS = 25
integer , dimension(NSEEDS) :: seeds
!
open(unit=11, file='ranlux.seed')
call RLUXUT(seeds)
write(11,*)seeds
close(11)
```

Για να ξεκινήσουμε τη RANLUX από την κατάσταση που σώσαμε στο προαναφερόμενο αρχείο, χρησιμοποιούμε την υπορουτίνα `RLUXIN` ως εξής:

```
integer , parameter      :: NSEEDS = 25
integer , dimension(NSEEDS) :: seeds
open(unit=11, file='ranlux.seed')
read(11,*)seeds
call RLUXIN(seeds)
```

Για να παράγουμε τυχαίους αριθμούς έναν-έναν χρησιμοποιώντας μια μεταβλητή scalar, καλούμε την RANLUX ως εξής

```
real(8)                  :: r
!
call ranlux(r,1)
print *,r
```

ενώ για να πάρουμε πολλούς τυχαίους αριθμούς σε ένα μονοδιάστατο array χρησιμοποιούμε τις εντολές

```

integer ,parameter      :: NR=20
real(8) ,dimension(NR)  :: randoms
!
call ranlux(randoms,NR)
print *,randoms

```

όπου η παράμετρος NR τίθεται κάθε φορά στην επιθυμητή τιμή. Βάζοντας όλα τα παραπάνω μαζί σε ένα πρόγραμμα που θα βρείτε στο αρχείο test\_ranlux.f90, μπορούμε να δοκιμάσουμε τη χρήση της RANLUX:

```

program use_ranlux
implicit none
integer ,parameter      :: NSEEDS = 25
integer ,dimension(NSEEDS) :: seeds
integer                 :: seed,ranlux_level
integer(8)              :: icount
real(8)                 :: r
integer ,parameter      :: NR=20
real(8) ,dimension(NR)  :: randoms
!
!start from a new seed:
seed      = 58266273
ranlux_level = 2
call RLUXGO(ranlux_level,seed,0,0)
!
!generate random numbers one by one:
do icount = 1,10
  call ranlux(r,1)
  print *,r
enddo
!generate random numbers in an array:
call ranlux(randoms,NR)
print '(1000G28.17)',randoms
!
!save state of ranlux:
open(unit=11,file='ranlux.seed')
call RLUXUT(seeds)
write(11,'(5I20)')seeds
close(11)
!
!generate some randoms:
call ranlux(randoms,NR)
print '(A,1000G28.17)', '#FIRST :',randoms
!
!read state of ranlux:
open(unit=11,file='ranlux.seed')
read(11,*)seeds
call RLUXIN(seeds)

```

```
!
!generate same randoms!
call ranlux(randoms,NR)
print '(A,1000G28.17)', '#SECOND:',randoms
end program use_ranlux
```

Μεταγλωττίστε, μαζί με το αρχείο ranlux.F που περιέχει τον κώδικα για τη Ranlux, και τρέξτε το παραπάνω πρόγραμμα με τις εντολές

```
> gfortran test_ranlux.f90 ranlux.F -o ranlux
> ./ranlux
```

### 11.3 Τυχαίες Διαδρομές

Υποθέτουμε ότι ένα σωματίο μπορεί να βρεθεί στις θέσεις ενός τετραγωνικού πλέγματος στις δύο διαστάσεις (στο επίπεδο). Αν σε κάποια χρονική στιγμή βρίσκεται σε μια από τις θέσεις του πλέγματος και αφού ισορροπήσει σε αυτή, τότε μπορεί να πηδήξει τυχαία σε μια γειτονική θέση όπου με τη σειρά του κάθεται για κάποιο χρόνο και ισορροπεί. Καθώς ισορροπεί, η ορμή που είχε και το βοήθησε να κάνει το πήδημα χάνεται, οπότε χάνει τη μνήμη της θέσης που βρισκόταν προηγουμένως. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς. Ο μηχανισμός του φαινομένου δεν θα μας απασχολήσει<sup>8</sup> και αναζητούμε μόνο ένα απλό φαινομενολογικό μοντέλο της διαδικασίας.

Υποθέτουμε ότι το σωματίο πηδάει με ίση πιθανότητα σε μια από τις πλησιέστερες θέσεις του πλέγματος κάθε φορά που περνάει σταθερός χρόνος  $\tau$ . Οι θέσεις του πλέγματος απέχουν μεταξύ τους απόσταση  $a$  (πλεγματική σταθερά). Το διάνυσμα που περιγράφει την μεταβολή της θέσης του σωματιδίου στο  $i$ -οστό πήδημα είναι μια τυχαία μεταβλητή  $\vec{\xi}_i$  μέτρου  $|\vec{\xi}_i| = a$ . Δηλαδή, δεδομένης της θέσης  $\vec{r}_k$  του σωματιδίου τη χρονική στιγμή  $t_k = k\tau$ , η θέση του  $\vec{r}_{k+1}$  τη χρονική στιγμή  $t_{k+1} = (k+1)\tau = t_k + \tau$  θα είναι

$$\vec{r}_{k+1} = \vec{r}_k + \vec{\xi}_k \quad (11.21)$$

<sup>8</sup>Μπορεί λ.χ. να είναι θερμικά διεγερμένα ηχητικά κύματα που δίνουν στο σωματίο την απαραίτητη ενέργεια για το πήδημα, το κβαντικό φαινόμενο σήραγγας κλπ



όπου

$$\vec{\xi}_k = \begin{cases} a\hat{x} & \text{με πιθανότητα } \frac{1}{4} \\ -a\hat{x} & \text{με πιθανότητα } \frac{1}{4} \\ a\hat{y} & \text{με πιθανότητα } \frac{1}{4} \\ -a\hat{y} & \text{με πιθανότητα } \frac{1}{4} \end{cases}. \quad (11.22)$$

Σύμφωνα με τα παραπάνω, οι τιμές των  $\vec{\xi}_i$  είναι ανεξάρτητες από την τιμή της θέσης που είχε προηγουμένως το σωματίο, οπότε οι τιμές  $\vec{\xi}_i$  και  $\vec{\xi}_j$  είναι ασυσχέτιστες για  $i \neq j$  και ισχύει ότι

$$\langle \vec{\xi}_i \cdot \vec{\xi}_j \rangle = \langle \vec{\xi}_i \rangle \cdot \langle \vec{\xi}_j \rangle. \quad (11.23)$$

Οι τιμές που παίρνουν τα  $\vec{\xi}_i$  είναι ισοπίθανες, οπότε, επειδή οι θετικές και αρνητικές τιμές συμβαίνουν το ίδιο συχνά οι θετικοί και αρνητικοί όροι στον υπολογισμό της  $\langle \vec{\xi}_i \rangle$  αλληλοαναιρούνται και έχουμε

$$\langle \vec{\xi}_i \rangle = \vec{0}, \quad (11.24)$$

και έτσι  $\langle \vec{\xi}_i \cdot \vec{\xi}_j \rangle = 0$  για  $i \neq j$ . Επειδή το μήκος των διανυσμάτων είναι σταθερό  $|\vec{\xi}_i| = a$ , παίρνουμε επομένως τη σχέση

$$\langle \vec{\xi}_i \cdot \vec{\xi}_j \rangle = a^2 \delta_{i,j}. \quad (11.25)$$

Η πιθανότητα να εμφανιστεί μια διαδρομή  $C_N$  μήκους  $N$  είναι<sup>9</sup>

$$p(C_N) = \frac{1}{z^N} \quad (11.26)$$

όπου  $z = 4$  ο αριθμός των πλησιέστερων γειτόνων μιας πλεγματικής θέσης. Η πιθανότητα αυτή εξαρτάται από το μήκος της διαδρομής και όχι από τη γεωμετρία της. Αυτό προκύπτει από την προφανή σχέση  $p(C_{N+1}) = \frac{1}{z} p(C_N)$ , αφού υπάρχουν ακριβώς  $z$  ισοπίθανες περιπτώσεις. Οπότε η συνάρτησή επιμερισμού είναι

$$Z_N = z^N, \quad (11.27)$$

και είναι ίση με τον αριθμό των διαφορετικών διαδρομών μήκους  $N$ .

Μέσα σε χρόνο  $t = N\tau$  το σωματίο μετατοπίζεται κατά

$$\vec{R} = \sum_{i=1}^N \vec{\xi}_i. \quad (11.28)$$

<sup>9</sup>Δηλ. μετά από χρόνο  $t = N\tau$ , όχι το φυσικό μήκος της καμπύλης. Μετράμε και τα πηδήματα πάνω στους συνδέσμους του πλέγματος που έχει ήδη επισκεφτεί το σωματίο.

Η μέση τιμή της μετατόπισης είναι 0

$$\langle \vec{R} \rangle = \sum_{i=1}^N \langle \vec{\xi}_i \rangle = \vec{0}. \quad (11.29)$$

Η μέση τιμή της μετατόπισης στο τετράγωνο είναι

$$\langle R^2 \rangle = \langle \vec{R} \cdot \vec{R} \rangle = \sum_{i,j=1}^N \langle \vec{\xi}_i \cdot \vec{\xi}_j \rangle = a^2 \sum_{i,j=1}^N \delta_{i,j} = a^2 N. \quad (11.30)$$

Βγάζουμε λοιπόν το πολύ σημαντικό αποτέλεσμα ότι ο τυχαίος περιπατητής απομακρύνεται πολύ αργά από το αρχικό του σημείο

$$R_{rms} = \sqrt{\langle R^2 \rangle} = a\sqrt{N} \propto \sqrt{t}. \quad (11.31)$$

Για σωμάτιο με μη μηδενική μέση ταχύτητα (δες πρόβλημα) αναμένεται  $R_{rms} \propto t$ .

Η παραπάνω σχέση ορίζει έναν κρίσιμο εκθέτη  $\nu$

$$\langle R^2 \rangle \sim N^{2\nu}, \quad (11.32)$$

όπου το σύμβολο  $\sim$  σημαίνει ασυμπτωτική συμπεριφορά για  $N \rightarrow \infty$ . Για ένα κλασικό περιπατητή  $\nu = 1$ , ενώ για τον τυχαίο περιπατητή  $\nu = \frac{1}{2}$ .

Παραλλαγές του Τυχαιού Περιπατητή (Random Walker – RW για συντομία) αποτελούν ο Μη Επιστρέφων Τυχαίος Περιπατητής (Non Reversal Random Walk – NRRW για συντομία) και ο Αυτοαποφεύγων Τυχαίος Περιπατητής (Self Avoiding Random Walk – SAW για συντομία). Ο NRRW ορίζεται όταν τα διανύσματα  $\vec{\xi}_i$  επιλέγονται ισοπίθανα αποκλείοντας την επιστροφή στο αρχικό σημείο. Ο SAW είναι ένας NRRW όπου κάθε φορά που ο περιπατητής επισκέπτεται μία πλεγματική θέση που έχει ήδη επισκεφτεί, το ... περπάτημα σταματάει. Επί πλέον είναι δυνατόν να προσθέσουμε, εκτός από την άπειρη απωστική ενέργεια για σημεία που συμπίπτουν, και μια ελκτική ενέργεια  $-\epsilon$  για κάθε ζευγάρι σημείων που ανήκουν στη διαδρομή και είναι πλησιέστεροι γείτονες. Κάθε επιτρεπόμενη διαδρομή θα ζυγίζεται τότε με πιθανότητα κατά Boltzmann σύμφωνα με την (12.4).

Για τον NRRW η σχέση (11.32) είναι ίδια με τον RW, δηλαδή  $\nu = \frac{1}{2}$ . Παρόλο που οι λεπτομέρειες των διαδρομών σε μικρές αποστάσεις είναι διαφορετικές, οι ιδιότητες τους σε μακροσκοπικές κλίμακες είναι παρόμοιες. Είναι περίπτωση συστημάτων που ανήκουν στην ίδια κλάση

παγκοσμιότητας (universality class) σύμφωνα με τη συζήτηση που θα βρείτε στην ενότητα 13.1.

Δεν συμβαίνει το ίδιο και για τον SAW. Για το σύστημα αυτό προβλέπεται ότι [49]

$$\langle R^2 \rangle^{\text{SAW}} \sim N^{2\nu} \quad \nu = \frac{3}{4}, \quad (11.33)$$

οπότε οι τυπικές διαδρομές στο πρότυπο αυτό είναι μεγαλύτερες από τον RW. Αν εισάγουμε την έλξη μεταξύ πλησιέστερων γειτόνων σύμφωνα με τα παραπάνω, τότε υπάρχει κρίσιμη θερμοκρασία  $\beta_c$  τέτοια ώστε για θερμοκρασία  $\beta < \beta_c$  να έχουμε παρόμοια συμπεριφορά με την (11.33), ενώ για  $\beta > \beta_c$  οι διαδρομές να “συνθλίβονται” από την αλληλεπίδραση και να παίρνουμε  $\nu = 1/3 < \nu_{\text{RW}}$ . Για  $\beta = \beta_c$  έχουμε  $\nu = \frac{1}{2}$ . Για περισσότερες λεπτομέρειες παραπέμπουμε στο βιβλίο των Binder-Heermann [7].

Μπορούμε τώρα να προγραμματίσουμε τον τυχαίο περιπατητή. Ο αλγόριθμος είναι πολύ απλός και οι μετρήσεις γίνονται με απλή δειγματοληψία:

1. Επιλέγουμε αριθμό τυχαίων διαδρομών που θα παράγουμε.
2. Επιλέγουμε αριθμό βημάτων κάθε διαδρομής.
3. Επιλέγουμε αρχική θέση των διαδρομών
4. Σε κάθε βήμα μιας διαδρομής επιλέγουμε με ίση πιθανότητα κίνηση προς τα δεξιά, αριστερά, πάνω και κάτω.
5. Στο τέλος κάθε διαδρομής, μετράμε τις ιδιότητες που μας ενδιαφέρουν ( $\bar{R}$ ,  $R^2$ , κλπ).
6. Στο τέλος υπολογίζουμε τις μέσες τιμές και σφάλματα στις ποσότητες που μετρήσαμε.

Η μόνη έννοια που χρειάζεται να εξηγήσουμε πώς θα προγραμματίσουμε είναι αυτή της επιλογής της “τυχαίας διεύθυνσης”. Το πρόγραμμα που δείχνεται παρακάτω θα το βρείτε στο αρχείο `rw.f90`

```
program random_walker
  implicit none
  integer,parameter :: Nwalk = 1000
  integer,parameter :: Nstep = 100000
  integer           :: iwalk,istep,ir
  real(8)           :: x,y
  real(8)           :: drandom
```

```

integer                :: seed
common /randoms/      seed

seed = 374676287
open(unit=20,file='dataR')
do iwalk = 1,Nwalk
  x = 0.0D0 ; y = 0.0D0
  open(unit=21,file='data')
  do istep=1,Nstep
    ir = INT(drandom()*4)
    select case(ir)
      case(0)
        x = x + 1.0D0
      case(1)
        x = x - 1.0D0
      case(2)
        y = y + 1.0D0
      case(3)
        y = y - 1.0D0
    end select
    write(21,*) x,y
  enddo !do istep=1,Nstep
  close(21)
  call sleep(2)
  write(20,*) x*x+y*y
  call flush(20)
enddo !do iwalk = 1,Nwalk
end program random_walker

```

Στο πρόγραμμα αυτό το μήκος των διαδρομών  $N_{\text{step}}$  και ο αριθμός των διαδρομών  $N_{\text{walk}}$  είναι παράμετροι που έχουν προκαθορισμένες τιμές. Οπότε για την αλλαγή των τιμών τους απαιτείται επαναμεταγλωττισμός του προγράμματος. Τα αποτελέσματά μας τα αποθηκεύουμε στα αρχεία `dataR` και `data` στα οποία αποθηκεύουμε το τετράγωνο της τελικής μετατόπισης  $R^2$  του RW και τις συντεταγμένες των σημείων  $(x, y)$  που επισκέπτεται ο περιπατητής, αντίστοιχα, σε κάθε διαδρομή. Για να κάνουμε τα περιεχόμενα των αρχείων άμεσα διαθέσιμα, αδειάζουμε τα I/O buffers με την υπορουτίνα `flush(unit)`. Παρατηρούμε ότι το αρχείο `data` ανοίγει και μηδενίζεται στην αρχή κάθε διαδρομής, οπότε περιέχει της συντεταγμένες μιας μόνο διαδρομής.

Κάθε διαδρομή εκτελείται σε  $N_{\text{step}}$  βήματα, όπου αφού επιλεγεί το τυχαίο διάνυσμα  $\vec{\xi}_{\text{istep}}$ , προστίθεται στην εκάστοτε θέση  $\vec{r}_{\text{istep}} = (x, y)$ . Το  $\vec{\xi}_{\text{istep}}$  επιλέγεται στη γραμμή

```
ir = INT(drandom()*4)
```

όπου ο  $ir = 0, 1, 2, 3$  λόγω της συνάρτησης INT που κόβει το δεκαδικό μέρος ενός real. Οι τιμές του  $ir$  αντιστοιχούν στις τέσσερις δυνατές τιμές του  $\xi$ . Η ενημέρωση της θέσης του περιπατητή γίνεται με το select case(ir) που ανάλογα με την τιμή του  $ir$  ενημερώνει τις συντεταγμένες που μεταβάλλονται στην αντίστοιχη διεύθυνση.

Για τη μεταγλώττιση του κώδικα και την εκτέλεση του προγράμματος δίνουμε τις εντολές

```
> gfortran rw.f90 drandom.f90 -o rw
> ./rw
```

που μας δίνει το εκτελέσιμο αρχείο rw που τρέχει το πρόγραμμα. Λόγω της εντολής call sleep(2), το πρόγραμμα σταματάει για 2 δευτερόλεπτα κάθε φορά που ολοκληρώνεται μια διαδρομή (αφαιρέστε την, όταν θα θέλετε να παράγετε πολλές διαδρομές). Αυτό μας επιτρέπει να παρακολουθήσουμε γραφικά τις διαδρομές που παράγονται. Κατά τη διάρκεια που το πρόγραμμα τρέχει, δώστε από το gnuplot την εντολή

```
gnuplot> plot "data" with lines
```

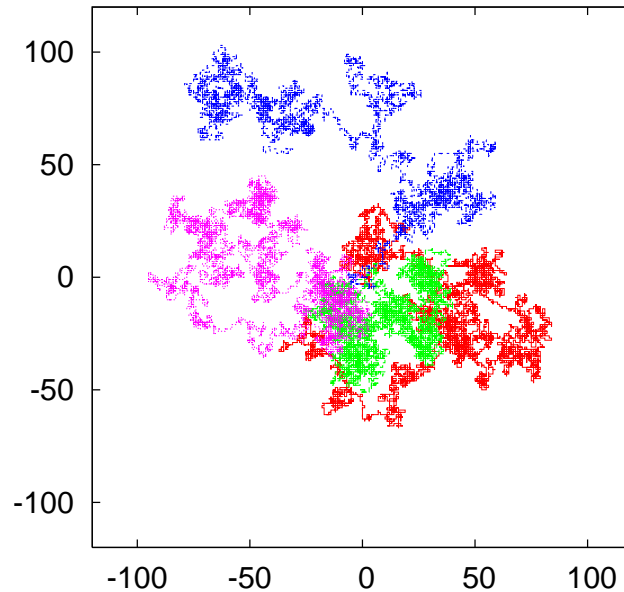
Επαναλάβετε την όσες φορές θέλετε για να δείτε άλλες τυχαίες διαδρομές. Για να γίνεται αυτή η διαδικασία αυτόματα χρησιμοποιήστε το script eternal-rw από το συνοδευτικό λογισμικό:

```
> ./rw &
> ./eternal-rw &
> killall rw eternal-rw gnuplot
```

όπου την τελευταία εντολή τη δίνετε όταν θέλετε να τερματίσετε την εκτέλεση των προγραμμάτων.

Στο σχήμα 11.8 παρουσιάζονται τυπικές διαδρομές από το τρέξιμο του παραπάνω προγράμματος. Στο σχήμα 11.9 φαίνονται τα αποτελέσματα για την ποσότητα  $\langle R^2 \rangle$  για  $N = 10, \dots, 100000$  τα οποία επιβεβαιώνουν τη σχέση (11.30)  $\langle R^2 \rangle = N$ . Το σχήμα αυτό μπορεί να αναπαραχθεί ως εξής:

1. Θέτουμε τις Nwalk και Nstep στις επιθυμητές τιμές στο αρχείο rw.f90. Αφαιρούμε τις εντολές call sleep(2) και write(21,\*) x,y και μεταγλωττίζουμε τον κώδικα.
2. Τρέχουμε το πρόγραμμα και αναλύουμε τα δεδομένα από το αρχείο dataR:



Σχήμα 11.8: Τέσσερις τυπικές διαδρομές του τυχαίου περιπατητή για  $N = 10000$ .

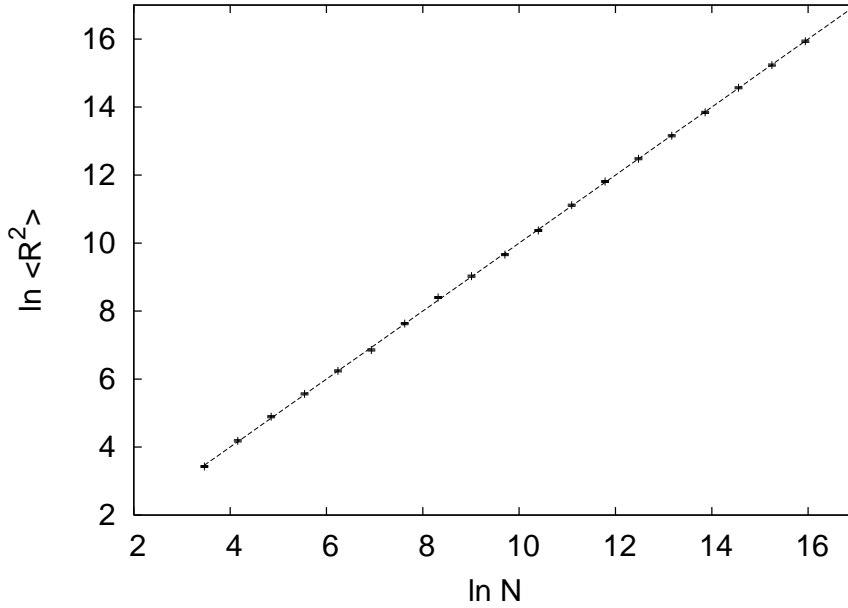
```
> ./rw
> awk '{av += $1}END{ print av/NR}' dataR
```

και σημειώνουμε τα αποτελέσματα σε ένα αρχείο `r2.dat` σε δύο στήλες με το μήκος των διαδρομών  $N$  στην πρώτη και το υπολογιζόμενο  $\langle R^2 \rangle$  στη δεύτερη. Το πρόγραμμα `awk` με την εντολή<sup>10</sup> `{av+=$1}` προσθέτει την πρώτη στήλη κάθε γραμμής του αρχείου `dataR` στη μεταβλητή `av`. Αφού διαβάσει το αρχείο, με την εντολή `END{print av/NR}`, τυπώνει τη μεταβλητή `av` διαιρεμένη με τον αριθμό των γραμμών στο αρχείο (`NR` = “Number of Records”). Παίρνουμε έτσι το μέσο όρο της πρώτης στήλης του `dataR`.

3. Με τη μέθοδο των ελαχίστων τετραγώνων βρίσκουμε τη βέλτιστη ευθεία  $y = ax + b$  που περνάει από τα σημεία  $(\ln N, \ln \langle R^2 \rangle)$ . Αυτό μπορεί να γίνει προσαρμόζοντας τα δεδομένα του αρχείου `r2.dat` με την εντολή `fit` του `gnuplot`:

```
gnuplot> fit a*x+b "r2.dat" u (log($1)):(log($2)) via a,b
```

<sup>10</sup>Η εντολή `av+=$1` είναι ισοδύναμη με την `av=av+$1`.



Σχήμα 11.9: Αριθμητική επιβεβαίωση της σχέσης  $\langle R^2 \rangle = N$  για  $N = 10, \dots, 100000$ . Η ευθεία γραμμή είναι προσαρμογή της συνάρτησης  $y = ax$  με  $a = 0.9994(13)$ .

4. Φτιάχνουμε τη γραφική παράσταση με την εντολή:

```
gnuplot> plot "r2.dat" u (log($1)):(log($2)) w e,a*x+b
```

Φυσικά τα παραπάνω αποτελέσματα δεν έχουν νόημα αν δεν γνωρίζουμε τα στατιστικά σφάλματα που υπεισέρχονται. Αφού πρόκειται για στατιστικές ποσότητες ανεξάρτητες μεταξύ τους, η αναμενόμενη τιμή προσεγγίζεται στο όριο άπειρων μετρήσεων με μια ταχύτητα  $\sim 1/\sqrt{M}$ , όπου  $M$  είναι ο αριθμός των μετρήσεων. Επειδή παράγουμε στοχαστικά ανεξάρτητες μεταξύ τους τυχαίες διαδρομές<sup>11</sup>, το σφάλμα στις μετρήσεις μας θα δίνεται από τη σχέση (11.3), λ.χ.

$$\delta \langle R^2 \rangle = \sqrt{\frac{1}{M-1} \left( \frac{1}{M} \sum_{i=1}^M (R_i^2)^2 - \left( \frac{1}{M} \sum_{i=1}^M R_i^2 \right)^2 \right)} \quad (11.34)$$

Το σφάλμα υπολογίζεται εύκολα, είτε προσθέτοντας τον υπολογισμό στο πρόγραμμα `rw.f90`, είτε κρατώντας το βασικό πρόγραμμα στη δυ-

<sup>11</sup>Αν υπάρχει στοχαστική εξάρτηση, τότε αυτή πρέπει να ληφθεί υπόψη όπως θα δούμε στα επόμενα κεφάλαια.

νατόν απλούστερη μορφή και αφήνοντας την επαναλαμβανόμενη εργασία σε εξωτερικά βοηθητικά προγράμματα (utilities). Στο αρχείο `average` γράφουμε τις εντολές `awk`:

```
#!/usr/bin/awk -f
{
    av += $1;      # the sum of data
    er += $1*$1;   # the sum of squares of data
}
END{
    av /= NR;      # NR = "Number of Records" = number of lines
    er /= NR;
    # formula for error of uncorrelated measurements
    er = sqrt( (er - av*av)/(NR-1) );
    print av, "+/-", er;
}
```

Το παραπάνω αρχείο είναι παράδειγμα από script το οποίο ερμηνεύεται από την `awk`. Αυτό γίνεται αντιληπτό στο λειτουργικό σύστημα από την πρώτη γραμμή `#!/bin/awk -f` που καλεί το πρόγραμμα όταν εκτελεστεί το αρχείο `average` ως μια οποιαδήποτε εντολή. Για να γίνει αυτό, το αρχείο πρέπει να γίνει εκτελέσιμο με την εντολή `chmod a+x average`. Στη συνέχεια, μπορούμε να εκτελέσουμε τις εντολές που περιέχει πάνω στο αρχείο `dataR` από τη γραμμή εντολών:

```
> ./average dataR
```

Θυμίζουμε πως στην `awk` οι εντολές που έχουμε μεταξύ `{ ... }` εκτελούνται για κάθε γραμμή του αρχείου `dataR`. Αυτές που είναι μεταξύ `END{ ... }` εκτελούνται, αφού διαβαστεί όλο το αρχείο<sup>12</sup>. Οπότε οι γραμμές

```
av += $1;      # the sum of data
er += $1*$1;   # the sum of squares of data
```

προσθέτουν στις μεταβλητές `av` και `er` την τιμή της πρώτης στήλης κάθε γραμμής του αρχείου `dataR` και του τετραγώνου αυτής αντίστοιχα. Οι γραμμές

```
av /= NR;      # NR = "Number of Records" = number of lines
```

<sup>12</sup>Υπάρχει και η δυνατότητα εκτέλεσης εντολών πριν διαβαστεί το αρχείο (λ.χ. για αρχικοποίηση μεταβλητών) όταν αυτές είναι μεταξύ `BEGIN{ ... }`



```
er /= NR;
```

εκτελούνται αφού διαβαστεί όλο το αρχείο dataR και διαιρούν τις παραπάνω μεταβλητές με την προκαθορισμένη μεταβλητή της awk NR. Αυτή έχει σε κάθε στιγμή τιμή ίση με τον αριθμό των γραμμών του αρχείου που έχει επεξεργαστεί το πρόγραμμα, άρα όταν εκτελείται το END{ ... } έχει τιμή ίση με το συνολικό αριθμό γραμμών του αρχείου. Οι τελευταίες γραμμές του script κάνουν τον τελικό υπολογισμό του σφάλματος σύμφωνα με τη σχέση (11.34) και τυπώνουν το αποτέλεσμα. Στο συνοδευτικό λογισμικό, θα βρείτε το αρχείο rw1-anal.csh όπου όλες οι παραπάνω εντολές κωδικοποιούνται σε ένα σενάριο φλοιού. Διαβάστε τις οδηγίες χρήσεις στα αρχικά σχόλια που υπάρχουν εκεί.

## 11.4 Ασκήσεις

1. Να αναπαράγετε τα αποτελέσματα του σχήματος 11.6 και να επιβεβαιώσετε τη σχέση (11.5)
2. Να παράγετε ακολουθία ψευδοτυχαίων αριθμών που έχουν Gaussian κατανομή με τυπική απόκλιση  $\sigma = 1/\sqrt{2}$ . Να φτιάξετε το αντίστοιχο σχήμα του 11.7 μαζί με τη συνάρτηση (11.12).
3. Να παράγετε ακολουθία ψευδοτυχαίων αριθμών που έχουν κατανομή κατά Cauchy με  $c = 1$ . Να κάνετε το διάγραμμα σχετικών συχνοτήτων μαζί με τη συνάρτηση κατανομής.
4. Να γράψετε πρόγραμμα που να υπολογίζει την περίοδο της συνάρτησης `drandom()`. Να ελέγξετε αν κατά τη διάρκεια της περιόδου παράγονται οι αριθμοί 0 και 1.
5. Να ελέγξετε πόσο “κοστίζει” σε CPU χρόνο η παραγωγή τυχαίων αριθμών. Αν έχετε ένα εκτελέσιμο πρόγραμμα, λ.χ. `random`, τρέξτε το μαζί με την εντολή `/usr/bin/time` ως εξής:

```
> /usr/bin/time ./random
```

Όταν τελειώσει η εκτέλεση της εντολής, το πρόγραμμα `/usr/bin/time` θα τυπώσει στο `stderr` το χρόνο CPU που κατανάλωσε το πρόγραμμα σε δευτερόλεπτα. Μετρήστε το χρόνο που χρειάζεται για την παραγωγή  $10^9$  τυχαίων αριθμών από την `drandom()`, την `random_number` και την `ranlux`. Στην τελευταία θα μεταβάλετε τον `ranlux_level`

από 1 έως 4. Πώς μεταβάλλεται ο χρόνος αυτός αν παράγετε τυχαίους αριθμούς με τις `random_number` και `ranlux` έναν-έναν και πώς αν τους ζητάτε σε ένα array μεγέθους 1000; Αν αυξήσετε το μέγεθος του array σε 10000, μεταβάλλεται ο χρόνος αυτός; (Υπόδειξη: Δείτε το αρχείο `performance_ran.f90` στο συνοδευτικό λογισμικό)

6. Για κάθε μία από τις γεννήτριες `drandom()`, `random_number` και `ranlux` να παράγετε 10 τυχαίους αριθμούς. Στη συνέχεια, να σώσετε την κατάστασή της σε ένα αρχείο. Μετά να παράγετε άλλους 10 τυχαίους αριθμούς. Διαβάστε, στη συνέχεια, από το αρχείο την κατάσταση της γεννήτριας από το αρχείο και να παράγετε άλλους 10 τυχαίους αριθμούς. Βεβαιωθείτε πως οι δύο τελευταίες ακολουθίες είναι ίδιες.
7. Μετατρέψτε το πρόγραμμα στο αρχείο `seed.f90` που θα βρείτε στο συνοδευτικό λογισμικό, ώστε να κάνετε seeding τη `ranlux`. Να το κάνετε με δύο τρόπους: (a) Παράγοντας έναν `seed` και ξεκινώντας με την `RLUXGO`. (b) Παράγοντας 25 `seeds` και ξεκινώντας με την `RLUXIN`.
8. Αποδείξτε ότι αν η μέση τιμή των διανυσμάτων  $\langle \vec{\xi}_i \rangle = \vec{v}\tau$ , τότε  $\langle \vec{R} \rangle = \vec{v}\tau N$  και έχουμε γραμμική σχέση μεταξύ μετατόπισης-μήκους τροχιάς. Η ποσότητα  $\vec{v}$  είναι η μέση ταχύτητα του σωματιδίου. Υπολογίστε το  $\langle R^2 \rangle$  για μεγάλες τιμές του  $N$ .
9. Εξετάστε αριθμητικά τις παραπάνω σχέσεις θέτοντας στον κώδικά σας την πιθανότητα στην πρώτη γραμμή της (11.22) ίση με  $1/2$  και τις υπόλοιπες  $1/6$ . Υπολογίστε τις ποσότητες  $\langle (\xi_i)_x \rangle$ ,  $\langle (\xi_i)_y \rangle$  και από αυτές τη μέση ταχύτητα του σωματιδίου. Εξετάστε τις σχέσεις  $\langle R^2 \rangle \sim N^a$  και  $\langle R_x \rangle \sim N^{2a_x}$   $\langle R_y \rangle \sim N^{2a_y}$ . Ποια η σχέση μεταξύ  $a$ ,  $a_x$  και  $a_y$ ;
10. Μεταβάλετε το πρόγραμμα `rw.f90`, ώστε ο χρήστης να παρέχει στο πρόγραμμα τις παραμέτρους `Nwalk` και `Nstep` από τα ορίσματα της εντολής που δίνει για να τρέξει το πρόγραμμα (δηλ. `./rw 100 2000` αν θέλει να παράξει 100 τυχαίες διαδρομές με  $N = 2000$ ). Χρησιμοποιήστε για το λόγο αυτό τις συναρτήσεις της Fortran `GETARG` και `IARGC`. (Υποδ.: δείτε το αρχείο `rw1.f90` στο συνοδευτικό λογισμικό)
11. Για τον τυχαίο περιπατητή γνωρίζουμε ότι  $\langle \vec{R} \rangle = \vec{0}$ . Υπολογίστε τις μέσες τιμές  $\langle x \rangle$  και  $\langle y \rangle$  για  $N = 100, 100000$ . Είναι πραγματικά

μηδέν; Γιατί; Πώς εξαρτάται το αποτέλεσμα σας από τον αριθμό των μετρήσεων;

12. Υπολογίστε τη μέση τιμή του αριθμού των φορών που ο τυχαίος περιπατητής επιστρέφει στο αρχικό σημείο σαν συνάρτηση του  $N$ . Τι συμβαίνει για  $N \rightarrow \infty$  και γιατί;
13. Να αναπαράγετε το σχήμα 11.9 για τον τυχαίο περιπατητή RW.
14. Να γράψετε πρόγραμμα που να υλοποιεί τον αλγόριθμο για τον NRRW και να αναπαράγετε το αντίστοιχο του σχήματος 11.9.
15. Στο πρόγραμμα `rw.f90` η θέση του περιπατητή καθορίζεται από δύο REAL(8) μεταβλητές  $x, y$ . Η επόμενη θέση υπολογίζεται από τη σχέση  $x=x+1.0D0$ ,  $y=y+1.0D0$ . Συζητήστε πόσο μεγάλες διαδρομές μπορείτε να μελετήσετε με αυτή την επιλογή. Να λάβετε υπόψη σας τη σχέση  $\langle R^2 \rangle = N$ .
16. Να επαναλάβετε την προηγούμενη άσκηση ορίζοντας INTEGER(8) μεταβλητές  $x, y$ . Η επόμενη θέση υπολογίζεται τώρα από τη σχέση  $x=x+1$ ,  $y=y+1$ . Συζητήστε τα πλεονεκτήματα και μειονεκτήματα κάθε επιλογής.
17. Να επαναλάβετε την προηγούμενη άσκηση ορίζοντας INTEGER(4) μεταβλητές  $x, y$ . Συζητήστε τα πλεονεκτήματα και μειονεκτήματα κάθε επιλογής μετρώντας τώρα και πόσο γρήγορα τρέχει το πρόγραμμά σας με την εντολή `/usr/bin/time`.
18. Να γράψετε κώδικα για τον SAW. Μέχρι πόσο μεγάλο  $N$  μπορείτε να προσομοιώσετε; Να δείξετε πως ο υπολογιστικός χρόνος για τον υπολογισμό δεδομένου αριθμού τυχαίων διαδρομών αυξάνει τουλάχιστον εκθετικά με το μέγεθος της διαδρομής  $N$ . Να αναζητήσετε στο ίντερνετ με πιο αλγόριθμο είναι δυνατόν να γίνει η προσομοίωση του SAW για αρκετά μεγάλα  $N$ .



## ΚΕΦΑΛΑΙΟ 12

### Προσομοιώσεις Μόντε Κάρλο

Στο κεφάλαιο αυτό γίνεται επισκόπηση των βασικών αρχών της μεθόδου υπολογισμού Μόντε Κάρλο σε συστήματα της στατιστικής φυσικής. Στην αρχή γίνεται μια επισκόπηση μερικών βασικών εννοιών της στατιστικής φυσικής. Στόχος είναι οι έννοιες αυτές να “φρεσκαριστούν” σε σύντομο χώρο στο κείμενο, οπότε δεν γίνεται προσπάθεια να είμαστε απόλυτα ακριβείς. Ειδικά, αναφερόμαστε στη συνάρτηση επιμερισμού της κανονικής συλλογής, στην εντροπία, στην πυκνότητα καταστάσεων και στη μελέτη των διακυμάνσεων των θερμοδυναμικών ποσοτήτων. Προτρέπουμε τον αναγνώστη να ανατρέξει στη βιβλιογραφία για πιο πλήρη κατανόηση των εννοιών αυτών όπως, ενδεικτικά, στα συγγράμματα [4, 42, 50, 51, 52, 53].

Δεν υπάρχει ενδιαφέρον φυσικό σύστημα του οποίου ο υπολογισμός της συνάρτησης επιμερισμού να γίνεται με απ’ ευθείας υπολογισμό. Στις περισσότερες περιπτώσεις καταφεύγουμε σε στατιστική δειγματοληψία και η πιο διαδεδομένη μέθοδος, λόγω της αποτελεσματικότητας της και της γενικής της εφαρμογής, είναι η μέθοδος Μόντε Κάρλο. Είναι αξιοσημείωτο ότι σε συστήματα όπως το απλό πρότυπο Ising χρήσιμα αποτελέσματα μπορούν να εξαχθούν σε μία τυπική προσομοίωση που γίνεται σε ένα λάπτοπ με τη μελέτη δειγμάτων της τάξης του  $\approx 10^{-3000}$  συνολικού χώρου των καταστάσεων<sup>1</sup>. Για πιο πολύπλοκα συστήματα, ο λόγος αυτός γίνεται ακόμα πιο εντυπωσιακός. Αυτό γίνεται διαισθητικά αποδεκτό λαμβάνοντας υπόψη ότι και στο εργαστήριο ένα πραγματικό στατιστικό σύστημα μας δίνει πειραματική πληροφορία, όταν μέσα στο χρόνο του πειράματος το σύστημα επισκέπτεται ακόμα μικρότερα πο-

---

<sup>1</sup>Λ.χ. για το  $d = 2$ ,  $L = 100$  πρότυπο Ising έχουμε  $2^{100 \times 100} = 2^{10000} \approx 10^{3010}$  καταστάσεις. Σε μια τυπική προσομοίωση παίρνουμε δείγμα από  $\approx 10^7$  καταστάσεις δηλ. ποσοστό  $\approx 10^{-3003}$ !

σοστά του φασικού χώρου<sup>2</sup>.

## 12.1 Στατιστική Φυσική

Η Στατιστική Φυσική έχει σκοπό να περιγράψει συστήματα με πολύ μεγάλο αριθμό βαθμών ελευθερίας  $N$ . Απλά συστήματα έχουν τυπικά  $N \approx 10^{23} - 10^{44}$ . Για τα συστήματα αυτά οι εξισώσεις που περιγράφουν μικροσκοπικά το σύστημα είναι πρακτικά αδύνατον και τελικά μάλλον άχρηστο να λυθούν. Αρκούν μερικά σωστά ορισμένες “ιδιότητες όγκου” (bulk properties) του συστήματος για να μας δώσουν τις χρήσιμες φυσικές πληροφορίες για το σύστημα. Λ.χ. σε έναν μαγνήτη πολλές φορές μας αρκεί να γνωρίζουμε την εσωτερική ενέργεια και μαγνήτιση του υλικού, σε ένα ρευστό την ενέργεια και πυκνότητά του κ.ο.κ. και όχι αναλυτικά την θέση, ενέργεια, ορμή και στροφορμή κάθε σωματιδίου που τα αποτελούν. Αυτά είναι γνωστά από τη θερμοδυναμική, στη στατιστική φυσική όμως γίνεται η απόπειρα να παραχθούν οι παραπάνω ιδιότητες των συστημάτων από τις μικροσκοπικές τους ιδιότητες, δηλ. κινηματική – βαθμοί ελευθερίας και δυναμική – Hamiltonian του συστήματος.

Στην περίπτωση μας θα κάνουμε τις, όχι ιδιαίτερα περιοριστικές, υποθέσεις ότι το σύστημά μας περιγράφεται από διακριτές καταστάσεις που μπορούν να απαριθμηθούν μέσα σε ένα σύνολο  $\{\mu\}$  με αντίστοιχες ενέργειες  $E_0 < E_1 < \dots < E_n < \dots$ . Το σύστημα αυτό είναι σε επαφή με μεγάλη δεξαμενή θερμότητας θερμοκρασίας  $\beta = 1/kT$  με το οποίο μπορεί να αλληλεπιδρά. Η επαφή με τη δεξαμενή και η δυναμική των βαθμών ελευθερίας έχει σαν αποτέλεσμα να συμβαίνουν τυχαίες μεταβάσεις μεταξύ καταστάσεων οι οποίες μπορούν να αλλάζουν την ενέργεια του συστήματος<sup>3</sup>. Η θεμελιώδεις ποσότητες που μας ενδιαφέρουν είναι τα βάρη (weights)  $w_\mu(t)$  που μας δίνουν την πιθανότητα να είναι το σύστημα στην κατάσταση  $\mu$  τη χρονική στιγμή  $t$ . Αυτές κωδικοποιούν την μικροσκοπική φυσική στη στατιστική φυσική.

<sup>2</sup>Ένα τυπικό αέριο με  $10^{22}$  μόρια σε δοχείο 1 λίτρου σε θερμοκρασία δωματίου και ατμοσφαιρική πίεση έχει μόρια που κινούνται με τυπικές ταχύτητες  $\approx 100\text{ms}^{-1}$  δηλ. τυπικό μήκος κύματος κατά de Broglie  $\approx 10^{-10}\text{m}$  δίνοντας  $\approx 10^{27}$  διαφορετικές καταστάσεις ανά μόριο. Συνολικά το σύστημα έχει  $(10^{27})^{10^{22}}$  καταστάσεις. Με ένα τυπικό ρυθμό από  $10^9$  χρούσεις ανά δευτερόλεπτο έχουμε  $\approx 10^{31}$  αλλαγές καταστάσεων ανά δευτερόλεπτο. Άρα θα χρειαστεί χρόνο περίπου  $10^{10^{23}}$  την ηλικία του σύμπαντος, ώστε το σύστημα να επισκεφτεί όλες τις καταστάσεις [4].

<sup>3</sup>Σε ένα απομονωμένο σύστημα η ενέργεια διατηρείται. Τέτοια συστήματα στη στατιστική φυσική μελετώνται στη μικροκανονική συλλογή.

Έστω ότι  $R(\mu \rightarrow \nu)$  δίνουν το ρυθμό μετάβασης από την κατάσταση  $\mu \rightarrow \nu$ , δηλ.

$$R(\mu \rightarrow \nu)dt = \text{Πιθανότητα μετάβασης } \mu \rightarrow \nu \text{ στο χρόνο } dt \quad (12.1)$$

Τότε μπορούμε να γράψουμε την πολύ γενική “δεσπόζουσα εξίσωση” (master equation):

$$\begin{aligned} \frac{dw_\mu(t)}{dt} &= \sum_\nu \{w_\nu(t)R(\nu \rightarrow \mu) - w_\mu(t)R(\mu \rightarrow \nu)\} \\ \sum_\mu w_\mu(t) &= 1. \end{aligned} \quad (12.2)$$

Η πρώτη από τις παραπάνω εξισώσεις μας λέει απλά ότι η μεταβολή του βάρους  $w_\mu(t)$  είναι ίση με το ρυθμό που το σύστημα εισέρχεται στην κατάσταση  $\mu$  από οποιαδήποτε άλλη  $\nu$  μείον το ρυθμό με τον οποίο φεύγει από την κατάσταση  $\mu$ . Η δεύτερη εκφράζει ότι τα βάρη  $w_\mu(t)$  ερμηνεύονται ως πιθανότητες και φυσικά η πιθανότητα να είναι το σύστημα σε κάποια κατάσταση είναι ίση με 1.

Οι ρυθμοί μετάβασης  $R(\mu \rightarrow \nu)$  προκύπτουν από τη θερμική φύση της αλληλεπίδρασης του συστήματος με τη θερμική δεξαμενή. Στην πράξη αυτοί προσομοιώνονται με κατάλληλες επιλογές κατά τη διάρκεια των υπολογισμών Μόντε Κάρλο. Τα  $R(\mu \rightarrow \nu)$  θεωρούνται ανεξάρτητα του χρόνου, οπότε το παραπάνω σύστημα εξισώσεων για τα  $w_\mu(t)$  είναι γραμμικό, και ο περιορισμός  $0 \leq w_\mu(t) \leq 1$  οδηγεί στο (μη τετριμμένο) συμπέρασμα ότι σε άπειρο χρόνο τα  $w_\mu(t)$  συγκλίνουν γρήγορα (για μεγάλα συστήματα) σε αριθμούς  $p_\mu$ , τις πιθανότητες κατάληψης ισορροπίας. Δηλαδή μετά από κάποιο χρόνο

$$p_\mu = \lim_{t \rightarrow \infty} w_\mu(t), \quad \sum_\mu p_\mu = 1. \quad (12.3)$$

Οι πιθανότητες  $p_\mu$  για σύστημα σε ισορροπία με θερμική ισορροπία με δεξαμενή θερμοκρασίας  $\beta = 1/kT$ ,  $k = 1.38 \times 10^{-23} \text{ JK}^{-1}$  μπορεί να δειχθεί (Gibbs 1902) ότι ακολουθούν την κατανομή Boltzmann

$$p_\mu = \frac{1}{Z} e^{-\beta E_\mu}. \quad (12.4)$$

Η παράμετρος  $\beta$  θα αναφέρεται απλά ως η θερμοκρασία του συστήματος και βλέπουμε ότι μέσω του εκθετικού στην εξίσωση (12.4) καθορίζει μία χαρακτηριστική ενέργεια για το σύστημα. Η μέτρησή της σε βαθμούς Kelvin κλπ οφείλεται σε ιστορικό ατύχημα εξαιτίας της

άγνοιας της μικροσκοπικής της προέλευσης κατά την αρχική θεμελίωση της θερμοδυναμικής.

Η σταθερά  $Z$  στην (12.4) είναι η συνάρτηση επιμερισμού του συστήματος και είναι η σταθερά κανονικοποίησης της κατανομής  $p_\mu$ . Η σχέση  $\sum_\mu p_\mu = 1$  μας δίνει

$$Z(\beta) = \sum_\mu e^{-\beta E_\mu} \quad (12.5)$$

Η τιμή μιας φυσικής ποσότητας που μετριέται στο εργαστήριο έχει στοχαστικό χαρακτήρα. Για συστήματα με πολύ μεγάλο αριθμό βαθμών ελευθερίας  $N$  πρακτικά κάποιος ενδιαφέρεται για τη μέση τιμή μιας ποσότητας. Αυτό γίνεται γιατί η πιθανότητα να μετρηθεί μια τιμή που διαφέρει σημαντικά από τη μέση τιμή είναι αμελητέα. Σύμφωνα με τα παραπάνω η μέση τιμή  $\langle \mathcal{O} \rangle$  μιας φυσικής ποσότητας  $\mathcal{O}$  η οποία παίρνει την τιμή  $\mathcal{O}_\mu$  στην κατάσταση  $\mu$  θα είναι

$$\langle \mathcal{O} \rangle = \sum_\mu p_\mu \mathcal{O}_\mu = \frac{1}{Z} \sum_\mu \mathcal{O}_\mu e^{-\beta E_\mu}. \quad (12.6)$$

Όπως θα δούμε παρακάτω, η τυπική απόκλιση  $\Delta \mathcal{O}$  για ένα τυπικό θερμοδυναμικό σύστημα είναι τέτοια, ώστε

$$\frac{\Delta \mathcal{O}}{\mathcal{O}} \sim \frac{1}{\sqrt{N}}, \quad (12.7)$$

ποσοστό το οποίο είναι αμελητέο για συνήθη μακροσκοπικά συστήματα (π.χ. για  $N \sim 10^{23}$  έχουμε  $\Delta \mathcal{O}/\mathcal{O} \sim 10^{-11}$ ). Για το λόγο αυτό, όταν το σύστημα είναι μεγάλο, οι διακυμάνσεις μπορούν να αγνοηθούν. Το όριο  $N \rightarrow \infty$  ονομάζεται **θερμοδυναμικό όριο** και το ενδιαφέρον μας εστιάζεται στην συμπεριφορά του συστήματος στο όριο αυτό. Στην πράξη, ενώ τα συστήματα στο εργαστήριο είναι τις περισσότερες φορές πολύ κοντά στο όριο αυτό, στις προσομοιώσεις μας, πολύ συχνά, δεν είναι δυνατόν να μελετήσουμε αρκετά μεγάλα συστήματα. Η όλη τέχνη επικεντρώνεται στο σχεδιασμό αλγορίθμων προσομοίωσης και μεθόδων ανάλυσης, έτσι ώστε να έχουμε εμπιστοσύνη ότι τα αποτελέσματά μας αντανακλούν τη συμπεριφορά του συστήματος στο θερμοδυναμικό όριο.

Η συνάρτηση επιμερισμού κωδικοποιεί λόγω του ορισμού (12.5) όλη τη στατιστική πληροφορία για το σύστημα. Δεν είναι μια απλή συνάρτηση μίας μεταβλητής (της  $\beta$ ), αλλά απαριθμεί με σχετικό βάρος τις δυνατές καταστάσεις του συστήματος. Απλό παράδειγμα αποτελεί ο υπολογισμός μέσω αυτής της μέσης ενέργειας  $\langle E \rangle$  (εσωτερικής ενέρ-



γεια  $U$  στη θερμοδυναμική) του συστήματος:

$$\begin{aligned} U &\equiv \langle E \rangle = \frac{1}{Z} \sum_{\mu} E_{\mu} e^{-\beta E_{\mu}} = -\frac{1}{Z} \sum_{\mu} \frac{\partial}{\partial \beta} e^{-\beta E_{\mu}} = -\frac{1}{Z} \frac{\partial}{\partial \beta} \sum_{\mu} e^{-\beta E_{\mu}} \\ &= -\frac{1}{Z} \frac{\partial Z}{\partial \beta} = -\frac{\partial \ln Z}{\partial \beta}. \end{aligned} \quad (12.8)$$

Με τον ίδιο τρόπο μπορεί κανείς να υπολογίσει την ειδική θερμότητα

$$C = \frac{\partial U}{\partial T} = \frac{\partial \beta}{\partial T} \frac{\partial U}{\partial \beta} = (-k\beta^2) \left( -\frac{\partial^2 \ln Z}{\partial \beta^2} \right) = k\beta^2 \frac{\partial^2 \ln Z}{\partial \beta^2}. \quad (12.9)$$

## 12.2 Εντροπία

Από τη θερμοδυναμική γνωρίζουμε ότι η εντροπία  $S$  ενός θερμοδυναμικού συστήματος ορίζεται από τη σχέση

$$S = -\frac{\partial F}{\partial T}, \quad F = U - TS, \quad (12.10)$$

όπου  $F$  είναι η ελεύθερη ενέργεια του συστήματος. Θα επιχειρήσουμε τώρα να δώσουμε ορισμούς που να είναι συνεπείς με τους παραπάνω και να προκύπτουν από τις μικροσκοπικές καταστάσεις του συστήματος.

Ορίζουμε την ελεύθερη ενέργεια από τη σχέση

$$e^{-\beta F} = Z \equiv \sum_{\mu} e^{-\beta E_{\mu}}, \quad (12.11)$$

ή ισοδύναμα

$$F = -\frac{1}{\beta} \ln Z. \quad (12.12)$$

Παρατηρήστε ότι ο παραπάνω ορισμός της  $F$  ταυτίζεται με την ενέργεια θεμελιώδους κατάστασης για  $T \rightarrow 0^+$ . Πράγματι τότε  $\beta \rightarrow \infty$  και ο μόνος όρος που επιζεί στο άθροισμα (12.11) είναι ο πρώτος. Για τον λόγο αυτό από την (12.10) θα έχουμε  $\lim_{T \rightarrow 0} S = 0$  που είναι ο τρίτος νόμος της θερμοδυναμικής.

---

<sup>4</sup>Όταν δεν υπάρχει αυθόρμητο σπάσιμο συμμετρίας που οδηγεί σε εκφυλισμό της ενέργειας της θεμελιώδους κατάστασης.

Ο ορισμός (12.11) είναι συνεπής με την (12.10), γιατί

$$U = -\frac{\partial \ln Z}{\partial \beta} = -\frac{\partial}{\partial \beta}(-\beta F) = F + \beta \frac{\partial F}{\partial \beta} = F - T \frac{\partial F}{\partial T} = F + TS. \quad (12.13)$$

Η σύνδεση της εντροπίας  $S$  με τη μικροφυσική γίνεται με τις σχέσεις (12.11) και (12.10):

$$\frac{S}{k} = \frac{U - F}{kT} = \beta(U - F) = \beta \left( \sum_{\mu} p_{\mu} E_{\mu} + \frac{1}{\beta} \ln Z \right). \quad (12.14)$$

Αλλά

$$p_{\mu} = \frac{e^{-\beta E_{\mu}}}{Z} \Rightarrow E_{\mu} = -\frac{1}{\beta} (\ln p_{\mu} + \ln Z), \quad (12.15)$$

οπότε

$$\begin{aligned} \frac{S}{k} &= \beta \sum_{\mu} \left( -\frac{1}{\beta} (\ln p_{\mu} + \ln Z) p_{\mu} + \frac{1}{\beta} \ln Z \right) \\ &= -\sum_{\mu} p_{\mu} \ln p_{\mu} - \ln Z \sum_{\mu} p_{\mu} + \ln Z \\ &= -\sum_{\mu} p_{\mu} \ln p_{\mu}. \end{aligned} \quad (12.16)$$

Τελικά

$$S = -k \sum_{\mu} p_{\mu} \ln p_{\mu} \quad (12.17)$$

Ας κάνουμε μία διερεύνηση του παραπάνω τύπου. Ας υποθέσουμε ότι σε ένα (ομολογουμένως ασυνήθιστο <sup>5)</sup> σύστημα όλες οι καταστάσεις έχουν την ίδια ενέργεια. Στην περίπτωση αυτή, με απλή αντικατάσταση στην εξίσωση (12.17) παίρνουμε ότι

$$p_{\mu} = \frac{1}{g} = \text{σταθ.} \Rightarrow S = k \ln g. \quad (12.18)$$

Δηλαδή η εντροπία μετράει τον αριθμό των καταστάσεων του συστήματος, όπως ακριβώς και στην περίπτωση της μικροκανονικής συλλογής. Πράγματι, η τελευταία σχέση προκύπτει και για την κατανομή

$$p_{\mu} = \begin{cases} \frac{1}{g(E)} & E_{\mu} = E \\ 0 & E_{\mu} \neq E \end{cases}, \quad (12.19)$$

<sup>5</sup>π.χ. η δισδιάστατη κβαντική βαρύτητα απουσία ύλης

που μπορεί να θεωρηθεί πως δίνει την μικροκανονική συλλογή αφού βάζει τον περιορισμό  $E_\mu = E = \text{σταθ.}$  Η συνάρτηση  $g(E)$  [σε πολλά βιβλία συμβολίζεται με  $\Omega(E)$ ] μετράει τον αριθμό των καταστάσεων με ενέργεια ίση με  $E$ . Από αυτή προκύπτει η πιθανότητα  $p(E)$  το σύστημα να βρεθεί να έχει ενέργεια  $E$

$$p(E) = \langle \delta_{E,E_\mu} \rangle = \sum_\mu p_\mu \delta_{E,E_\mu} = \frac{1}{Z} \sum e^{-\beta E_\mu} \delta_{E,E_\mu} = \frac{1}{Z} e^{-\beta E} \sum \delta_{E,E_\mu}. \quad (12.20)$$

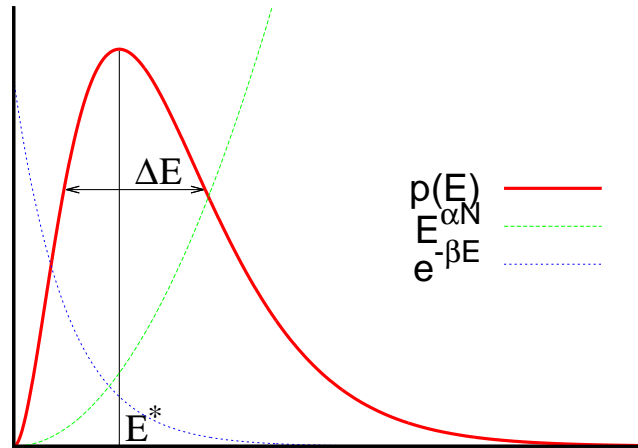
Επειδή προφανώς  $\sum_\mu \delta_{E,E_\mu} = g(E)$ , παίρνουμε

$$p(E) = \langle \delta_{E,E_\mu} \rangle = \frac{g(E) e^{-\beta E}}{Z}. \quad (12.21)$$

Για ένα τυπικό σύστημα σαν αυτά που θα μελετήσουμε ισχύει

$$g(E) \sim E^{\alpha N} \quad (12.22)$$

όπου  $N$  ο αριθμός των βαθμών ελευθερίας του συστήματος και  $\alpha$  μια σταθερά. Η ποιοτική συμπεριφορά της κατανομής (12.21) φαίνεται στο



Σχήμα 12.1: Η πιθανότητα  $p(E)$  όπως προκύπτει από τον ανταγωνισμό του παράγοντα Boltzmann  $e^{-\beta E}$  και της πυκνότητας καταστάσεων  $g(E) \sim E^{\alpha N}$  για μία τυπική περίπτωση.

σχήμα 12.1. Γενικά παρατηρούμε ότι οι πιθανές τιμές της ενέργειας επικεντρώνονται γύρω από μια τιμή  $E^*$  και η απόκλιση  $\Delta E$  είναι ένα

μέτρο της διασποράς των τιμών που, όπως θα δείξουμε παρακάτω, ο λόγος  $\Delta E/E$  μειώνεται με το  $N$  με χαρακτηριστική συμπεριφορά  $1/\sqrt{N}$ . Πράγματι, η συνάρτηση (ανάλογη της  $p(E)$ )

$$\tilde{p}(E) = E^{\alpha N} e^{-\beta E} = e^{-\beta E - \alpha N \ln E} \quad (12.23)$$

έχει μέγιστο, όταν

$$\left. \frac{\partial \ln \tilde{p}(E)}{\partial E} \right|_{E=E^*} = 0 \Rightarrow \left. \frac{\partial}{\partial E} (-\beta E + \alpha N \ln E) \right|_{E=E^*} = -\beta + \frac{\alpha N}{E^*} = 0 \quad (12.24)$$

ή

$$E^* = \frac{\alpha}{\beta} N. \quad (12.25)$$

Το  $E^*$  μετατοπίζεται προς μεγαλύτερες τιμές με την αύξηση της θερμοκρασίας (μείωση του  $\beta$ ) και είναι, όπως αναμένεται, ανάλογο του μεγέθους του συστήματος. Αναπτύσσοντας κατά Taylor

$$\begin{aligned} \ln \tilde{p}(E) &= \ln \tilde{p}(E^*) + (E - E^*) \left. \frac{\partial \ln \tilde{p}(E)}{\partial E} \right|_{E=E^*} \\ &\quad + \frac{1}{2} (E - E^*)^2 \left. \frac{\partial^2 \ln \tilde{p}(E)}{\partial E^2} \right|_{E=E^*} + \dots \\ &= \ln \tilde{p}(E^*) + \frac{1}{2} (E - E^*)^2 \left( -\frac{\alpha N}{(E^*)^2} \right) + \dots, \end{aligned} \quad (12.26)$$

όπου χρησιμοποιήσαμε την συνθήκη (12.24) και υπολογίσαμε την  $\left. \frac{\partial^2 \ln \tilde{p}(E)}{\partial E^2} \right|_{E=E^*}$ . Οπότε, προκύπτει ότι

$$p(E) \approx p(E^*) e^{-\alpha N \frac{(E - E^*)^2}{2(E^*)^2}}. \quad (12.27)$$

Η παραπάνω κατανομή είναι Gaussian με τυπική απόκλιση

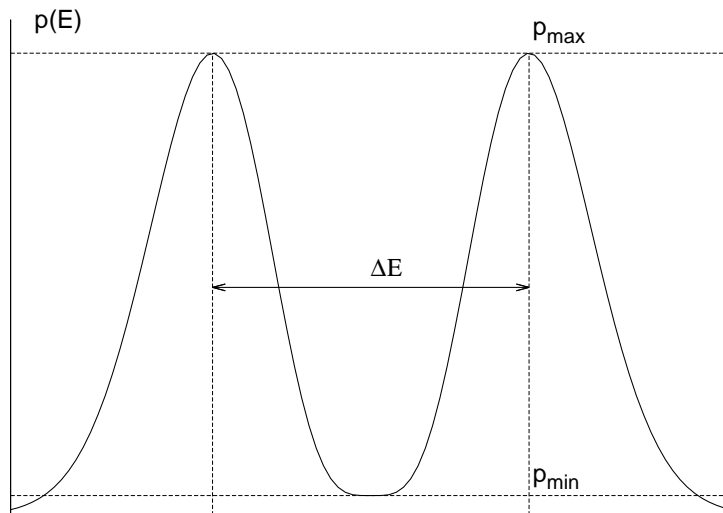
$$\Delta E \sim \sqrt{\frac{(E^*)^2}{\alpha N}} = \sqrt{\frac{(\frac{\alpha N}{\beta})^2}{\alpha N}} \sim \frac{\sqrt{N}}{\beta}, \quad (12.28)$$

όπου κρατήσαμε μόνο την εξάρτηση από το μέγεθος του συστήματος  $N$  και τη θερμοκρασία  $\beta$ . Οπότε πράγματι επιβεβαιώνουμε τη σχέση (12.7)

$$\frac{\Delta E}{E^*} \sim \frac{\frac{\sqrt{N}}{\beta}}{\frac{\alpha N}{\beta}} = \frac{1}{\sqrt{N}}. \quad (12.29)$$

Η παραπάνω ανάλυση υπέθεσε αναλυτική συμπεριφορά [ανάπτυγμα Taylor, σχέση (12.26)] η οποία δεν ισχύει όταν είμαστε σε ένα κρίσιμο σημείο μιας μετάβασης φάσης.

Μια άλλη σημαντική περίπτωση που η παραπάνω ανάλυση δεν ισχύει είναι όταν η κατανομή  $p(E)$  έχει παραπάνω από ένα μέγιστα <sup>6</sup>. Αυτό συμβαίνει όταν το σύστημα υπόκειται σε μετάβαση φάσης πρώτης τάξης, π.χ. όταν ο πάγος γίνεται νερό ή όταν ένα μαγνητικό υλικό που βρίσκεται σε ένα μαγνητικό πεδίο χάνει τη μαγνήτισή του λόγω αύξησης της θερμοκρασίας του. Στην περίπτωση αυτή, οι δύο καταστάσεις πάγος – νερό/μαγνήτης – παραμαγνήτης έχουν την ίδια πιθανότητα εμφάνισης (“συνυπάρχουν”) και μία τυπική κατανομή με δομή δύο κορυφών φαίνεται στο σχήμα 12.2.



Σχήμα 12.2: Η πιθανότητα  $p(E)$  με δομή δύο κορυφών σε σύστημα με μετάβαση φάσης 1ης τάξης. Τα δύο μέγιστα αντιστοιχούν στις δύο συνυπάρχουσες καταστάσεις (“πάγος”–“νερό”) και  $\Delta E/N$  αντιστοιχεί στη λανθάνουσα θερμότητα (latent heat). Στο θερμοδυναμικό όριο  $N \rightarrow \infty$  το  $R = p_{\min}/p_{\max}$  μειώνεται σαν  $R \sim e^{-fA}$ , όπου  $A$  η ελάχιστη επιφάνεια που χωρίζει τις δύο φάσεις και  $f$  η διεπαφική τάση (interface tension).

<sup>6</sup>Όταν έχει περισσότερα από ένα τοπικά μέγιστα, το ολικό μέγιστο επικρατεί των υπολοίπων στο θερμοδυναμικό όριο  $N \rightarrow \infty$ .

### 12.3 Διακυμάνσεις

Κάθε παρατηρήσιμη ποσότητα  $\mathcal{O}$  έχει στοχαστική συμπεριφορά σύμφωνα με μια κατανομή πιθανότητας  $p(\mathcal{O})$  που προκύπτει από την κατανομή Boltzmann (12.4). Μια τέτοια κατανομή χαρακτηρίζεται πλήρως από τη μέση τιμή  $\langle \mathcal{O} \rangle$  και τις ροπές ανώτερης τάξης, δηλ. τις μέσες τιμές  $\langle (\mathcal{O} - \langle \mathcal{O} \rangle)^n \rangle$ ,  $n = 1, 2, 3, \dots$ . Η πιο χρήσιμη ποσότητα από αυτές δίνεται από τη διακύμανση γύρω από τη μέση τιμή για  $n = 2$

$$(\Delta \mathcal{O})^2 \equiv \langle (\mathcal{O} - \langle \mathcal{O} \rangle)^2 \rangle = \langle \mathcal{O}^2 \rangle - \langle \mathcal{O} \rangle^2. \quad (12.30)$$

Η ποσότητα αυτή δίνει τις περισσότερες φορές ένα μέτρο της απόκλισης της  $\mathcal{O}$  από τη μέση τιμή της  $\langle \mathcal{O} \rangle$ . Για την περίπτωση της ενέργειας  $\mathcal{O} = E$  παίρνουμε

$$(\Delta E)^2 \equiv \langle (E - \langle E \rangle)^2 \rangle = \langle E^2 \rangle - \langle E \rangle^2, \quad (12.31)$$

και από τις σχέσεις

$$\langle E^2 \rangle = \frac{1}{Z} \sum_{\mu} E_{\mu}^2 e^{-\beta E_{\mu}} = \frac{1}{Z} \frac{\partial^2}{\partial \beta^2} \sum_{\mu} e^{-\beta E_{\mu}} = \frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2}, \quad (12.32)$$

και

$$\langle E \rangle = \frac{1}{Z} \sum_{\mu} E_{\mu} e^{-\beta E_{\mu}} = -\frac{1}{Z} \frac{\partial}{\partial \beta} \sum_{\mu} e^{-\beta E_{\mu}} = -\frac{1}{Z} \frac{\partial Z}{\partial \beta}, \quad (12.33)$$

προκύπτει ότι

$$(\Delta E)^2 = \langle E^2 \rangle - \langle E \rangle^2 = \frac{1}{Z} \frac{\partial^2 Z}{\partial \beta^2} - \left( -\frac{1}{Z} \frac{\partial Z}{\partial \beta} \right)^2 = \frac{\partial^2 \ln Z}{\partial \beta^2}, \quad (12.34)$$

που σύμφωνα με τη σχέση (12.9) είναι η ειδική θερμότητα

$$C = \frac{\partial \langle E \rangle}{\partial T} = k\beta^2 (\Delta E)^2. \quad (12.35)$$

Άρα, καταλήγουμε στο ενδιαφέρον συμπέρασμα ότι η ειδική θερμότητα ενός συστήματος (θερμοδυναμική ποσότητα) συνδέεται άμεσα με τις μικροσκοπικές στατιστικές διακυμάνσεις της ενέργειας του συστήματος.

Αυτό ισχύει γενικά για οποιαδήποτε φυσική ποσότητα του συστήματος που έχει γραμμική σύζευξη με ένα εξωτερικό πεδίο. Μία τέτοια ποσότητα για ένα μαγνητικό σύστημα που βρίσκεται μέσα σε ένα ομογενές μαγνητικό πεδίο  $B$  είναι η μαγνήτιση  $M$ . Αν  $M_{\mu}$  είναι η μαγνήτιση

του συστήματος στην κατάσταση  $\mu$  και αν υποθέσουμε ότι είναι στην διεύθυνση του  $\vec{B}$ , τότε η Hamiltonian του συστήματος γίνεται

$$H = E - BM \quad (12.36)$$

και η συνάρτηση επιμερισμού

$$Z = \sum_{\mu} e^{-\beta E_{\mu} + \beta BM_{\mu}}. \quad (12.37)$$

Ο όρος “γραμμική σύζευξη” οφείλεται στο γραμμικό όρο  $BM$  στη Hamiltonian. Οι ποσότητες  $B$  και  $M$  ονομάζονται συζυγείς. Παρόμοια σχέση έχουν η πίεση/όγκος ( $P/V$ ) σε ένα αέριο ή το χημικό δυναμικό/αριθμός σωματιδίων ( $\mu/N$ ) στη μεγαλοκανονική συλλογή.

Εξαιτίας αυτής της γραμμικής σύζευξης παίρνουμε

$$\langle M \rangle = \frac{1}{Z} \sum_{\mu} M_{\mu} e^{-\beta E_{\mu} + \beta BM_{\mu}} = \frac{1}{\beta Z} \frac{\partial Z}{\partial B} = -\frac{\partial F}{\partial B}, \quad (12.38)$$

σχέση που είναι ανάλογη με την (12.8). Η αντίστοιχη της (12.34) προκύπτει από (12.30) για  $\mathcal{O} = M$

$$(\Delta M)^2 \equiv \langle (M - \langle M \rangle)^2 \rangle = \langle M^2 \rangle - \langle M \rangle^2, \quad (12.39)$$

και από

$$\langle M^2 \rangle = \frac{1}{Z} \sum_{\mu} M_{\mu}^2 e^{-\beta E_{\mu} + \beta BM_{\mu}} = \frac{1}{\beta^2 Z} \frac{\partial^2 Z}{\partial B^2}, \quad (12.40)$$

οπότε

$$(\Delta M)^2 = \frac{1}{\beta^2} \left\{ \frac{1}{Z} \frac{\partial^2 Z}{\partial B^2} - \frac{1}{Z^2} \left( \frac{\partial Z}{\partial B} \right)^2 \right\} = \frac{1}{\beta^2} \frac{\partial^2 \ln Z}{\partial B^2} = \frac{1}{\beta} \frac{\partial \langle M \rangle}{\partial B}. \quad (12.41)$$

Η μαγνητική επιδεκτικότητα  $\chi$  ορίζεται από τη σχέση

$$\chi = \frac{1}{N} \frac{\partial \langle M \rangle}{\partial B} = \frac{\beta}{N} \langle (M - \langle M \rangle)^2 \rangle, \quad (12.42)$$

από όπου φαίνεται ότι σχετίζεται άμεσα με τις διακυμάνσεις της μαγνήτισης. Η παραπάνω ανάλυση μπορεί να γίνει πανομοιότυπα για οποιοδήποτε ζεύγος συζυγών ποσοτήτων.

## 12.4 Συναρτήσεις Συσχετισμού

Οι συναρτήσεις συσχετισμού προκύπτουν από τη συζήτηση της προηγούμενης παραγράφου, αν θεωρήσουμε μαγνητικά πεδία τα οποία έχουν τιμή που εξαρτάται από τη θέση στο χώρο. Για λόγους απλότητας (αλλά και επειδή πρόκειται να μελετήσουμε μόνο τέτοια συστήματα) το σύστημά μας βρίσκεται μέσα σε ένα “χώρο” στον οποίο οι δυνατές θέσεις είναι οι διακριτές θέσεις ενός πλέγματος τις οποίες αντιστοιχούμε σε φυσικούς αριθμούς<sup>7</sup>  $i = 1, \dots, N$ . Τότε το μαγνητικό πεδίο  $B_i$  θα είναι συνάρτηση της θέσης στο πλέγμα και αλληλεπιδρά με το σπιν  $s_i$

$$H = E - \sum_i B_i s_i, \quad (12.43)$$

και η μαγνήτιση ανά πλεγματική θέση  $m_i \equiv s_i$ <sup>8</sup> στη θέση πλέγματος  $i$  είναι

$$\langle s_i \rangle = \frac{1}{\beta} \frac{\partial \ln Z}{\partial B_i}. \quad (12.44)$$

Η συνάρτηση συσχετισμού δύο σημείων (connected two point correlation function) ορίζεται ως

$$G_c^{(2)}(i, j) = \langle (s_i - \langle s_i \rangle)(s_j - \langle s_j \rangle) \rangle = \langle s_i s_j \rangle - \langle s_i \rangle \langle s_j \rangle = \frac{1}{\beta^2} \frac{\partial^2 \ln Z}{\partial B_i \partial B_j}. \quad (12.45)$$

Η παραπάνω συνάρτηση έχει μεγάλη θετική τιμή, όταν οι τιμές  $s_i, s_j$  είναι ισχυρά συσχετισμένες, δηλ. “μεταβάλλονται μαζί” στα τυχαία δείγματα που παίρνουμε από το σύστημα, ενώ αντίθετα είναι σχεδόν μηδέν όταν η τιμή της  $s_i$  εξαρτάται ελάχιστα από την  $s_j$  (ασυσχέτιστες τυχαίες μεταβλητές). Υπάρχει, φυσικά, και οι περίπτωση οι  $s_i, s_j$  να είναι ισχυρά αντι-συσχετισμένες και η συνάρτηση συσχετισμού να είναι αρνητική.

Η συνάρτηση συσχετισμού  $G_c^{(2)}(i, j)$  παίρνει τη μέγιστη τιμή της  $\langle (s_i - \langle s_i \rangle)^2 \rangle$  για  $i = j$ . Στη συνέχεια, πέφτει γρήγορα κατά απόλυτη τιμή. Για ένα σύννηθες σύστημα

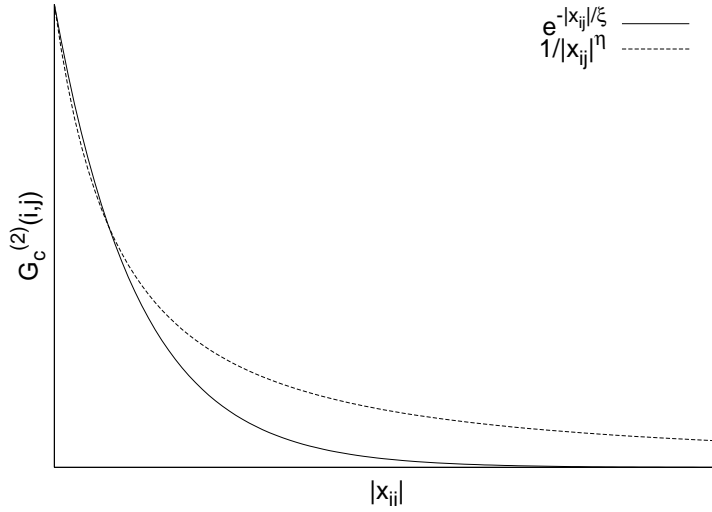
$$G_c^{(2)}(i, j) \sim e^{-|x_{ij}|/\xi}, \quad (12.46)$$

όπου  $|x_{ij}|$  η απόσταση των σημείων  $i, j$ . Το μήκος συσχετισμού  $\xi$ , είναι ένα χαρακτηριστικό μήκος για το σύστημα που δίνει ένα μέτρο της

<sup>7</sup>Οι βαθμοί ελευθερίας που αναφέραμε προηγουμένως μπορεί να είναι περισσότεροι από  $N$ .

<sup>8</sup>Οι δύο φυσικές ποσότητες είναι διαφορετικές, αλλά ανάλογες. Εδώ απλά αγνοούμε τη σταθερά αναλογίας.





Σχήμα 12.3: Η συνάρτηση συσχετισμού  $G_c^{(2)}(i, j)$  για  $\xi < \infty$  και  $\xi \rightarrow \infty$ .

απόστασης όπου υπάρχει ουσιαστικός συσχετισμός μεταξύ των τιμών της μαγνήτισης σε δύο πλεγματικές θέσεις. Εξαρτάται από τις παραμέτρους που ορίζουν το σύστημα  $\xi = \xi(\beta, B, N, \dots)$ . Είναι σημαντικό να κατανοηθεί ότι το μήκος συσχετισμού ορίζει μια κλίμακα μήκους που προκύπτει δυναμικά. Αντίθετα κλίμακες μήκους όπως το μέγεθος  $L$  του συστήματος ή η απόσταση  $a$  μεταξύ δύο πλεγματικών σημείων (πλεγματική σταθερά) είναι κλίμακες μήκους που δίνονται από τον ορισμό του συστήματος και δεν εξαρτώνται από τις δυναμικές παραμέτρους. Συνήθως το  $\xi$  είναι της τάξης μεγέθους της πλεγματικής σταθεράς  $a$  και το σύστημα δεν παρουσιάζει συσχετισμούς σε μακροσκοπικές κλίμακες (δηλ. της τάξης του  $L$ ).

Πολύ ενδιαφέρουσα φυσική προκύπτει όταν σε κάποια συστήματα μπορούμε να ρυθμίσουμε με λεπτότητα τις παραμέτρους από τις οποίες εξαρτάται το  $\xi$ , έτσι ώστε στο θερμοδυναμικό όριο να πάρουμε  $\xi \rightarrow \infty$ . Αυτό γίνεται στην περιοχή μίας συνεχούς (όχι πρώτης τάξης) μετάβασης φάσης. Στην περίπτωση, αυτή η εκθετική συμπεριφορά χάνεται και έχουμε πολύ βραδύτερη πτώση της  $G_c^{(2)}(i, j)$  (βλ. σχήμα 12.3), που σε  $d$  χωρικές διαστάσεις δίνεται από

$$G_c^{(2)}(i, j) \sim \frac{1}{|x_{ij}|^{d-2+\eta}}. \quad (12.47)$$

Καθώς πλησιάζουμε το κρίσιμο σημείο, οι συσχετισμοί εκτείνονται σε αποστάσεις  $|x_{ij}| \gg a$ . Τότε το σύστημα παύει να “βλέπει” τις λεπτο-

μέρειες του πλέγματος και συμπεριφέρεται με πολύ καλή προσέγγιση σαν το πλέγμα να ήταν ένας συνεχής χώρος. Το όριο αυτό αναφέρεται ως το “συνεχές όριο” (continuum limit) μιας θεωρίας που ορίζεται σε ένα πλέγμα. Εξαιτίας του ότι οι λεπτομέρειες του πλέγματος γίνονται ασήμαντες στο όριο αυτό, θεωρίες που ορίζονται μικροσκοπικά με διαφορετικό τρόπο (λ.χ. μια σε τετραγωνικό πλέγμα και μία σε εξαγωνικό) έχουν το ίδιο συνεχές όριο. Το φαινόμενο αυτό ονομάζεται παγκοσμιότητα (universality) και παίζει κεντρικό ρόλο στη μελέτη στατιστικών συστημάτων, καθώς και στην κβαντική θεωρία πεδίου.

## 12.5 Δειγματοληψία

Ο κύριος στόχος μας είναι ο προσδιορισμός της μέσης τιμής  $\langle \mathcal{O} \rangle$  μιας φυσικής ποσότητας  $\mathcal{O}$  (λ.χ. ενέργειας, μαγνήτισης, συνάρτησης συσχετισμού) σε ένα στατιστικό σύστημα στην κανονική συλλογή

$$\langle \mathcal{O} \rangle = \sum_{\mu} p_{\mu} \mathcal{O}_{\mu} = \frac{\sum_{\mu} \mathcal{O}_{\mu} e^{-\beta E_{\mu}}}{\sum_{\mu} e^{-\beta E_{\mu}}} . \quad (12.48)$$

Για το λόγο αυτό επιλέγουμε δείγμα από  $M$  καταστάσεις  $\{\mu_1, \mu_2, \dots, \mu_M\}$  οι οποίες κατανέμονται σύμφωνα με την κατανομή πιθανότητας  $P_{\mu}$  και ορίζουμε τον εκτιμητή (estimator)  $\mathcal{O}_M$  της  $\langle \mathcal{O} \rangle$

$$\mathcal{O}_M = \frac{\sum_{i=1}^M \mathcal{O}_{\mu_i} P_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^M P_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}} . \quad (12.49)$$

Ο παραπάνω τύπος γίνεται εύκολα κατανοητός αφού, για μεγάλο δείγμα,  $P_{\mu_i} \approx$  “Συχνότητα εύρεσης κατάστασης  $\mu_i$  στο δείγμα”, και περιμένουμε ότι

$$\langle \mathcal{O} \rangle = \lim_{M \rightarrow \infty} \mathcal{O}_M . \quad (12.50)$$

Ο στόχος μας είναι η κατάλληλη επιλογή της κατανομής  $P_{\mu}$ , έτσι ώστε η σύγκλιση (12.50) να γίνεται γρήγορα. Διακρίνουμε τις εξής περιπτώσεις:

### 12.5.1 Απλή Δειγματοληψία

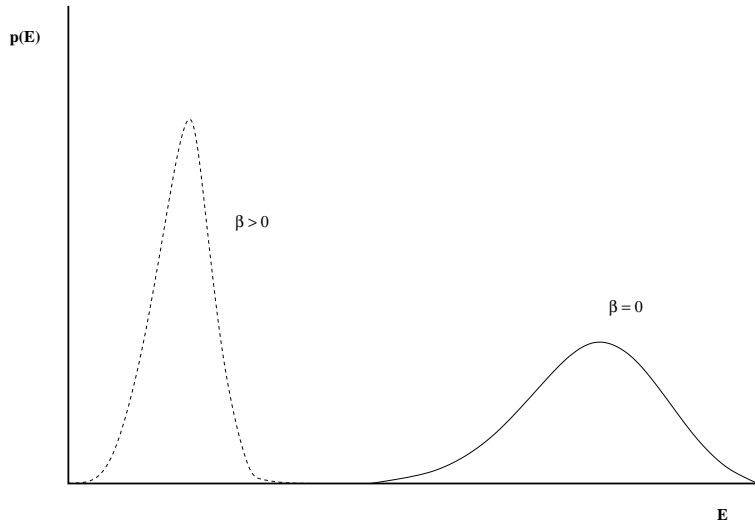
Διαλέγουμε  $P_{\mu} = \text{σταθ.}$ , οπότε η (12.49) γίνεται

$$\mathcal{O}_M = \frac{\sum_{i=1}^M \mathcal{O}_{\mu_i} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^M e^{-\beta E_{\mu_i}}} . \quad (12.51)$$

Το πρόβλημα με την επιλογή αυτή είναι ότι έχουμε πρόβλημα επικάλυψης του δείγματος με τις καταστάσεις που πραγματικά συνεισφέρουν στο άθροισμα (12.48). Όπως αναφέραμε και στην εισαγωγή, ο αριθμός των καταστάσεων που επιλέγεται σε μια προσομοίωση Μόντε Κάρλο στο δείγμα είναι ένα ελάχιστο ποσοστό του χώρου των καταστάσεων του συστήματος. Άρα, η πιθανότητα να πάρουμε εκείνες τις καταστάσεις που δίνουν σημαντική συνεισφορά στο άθροισμα (12.48) είναι εν γένει πολύ μικρή. Ας πάρουμε για παράδειγμα την περίπτωση  $O = E$  σε ένα τυπικό μοντέλο. Σύμφωνα με τη σχέση (12.21) έχουμε

$$\langle E \rangle = \sum_E E p(E) \quad (12.52)$$

όπου  $p(E)$  είναι η πιθανότητα εμφάνισης ενέργειας  $E$  στο σύστημα. Η ποιοτική μορφή της φαίνεται στο σχήμα 12.1. Από τις σχέσεις (12.25) και (12.28) έχουμε ότι  $E^* \sim 1/\beta$  και  $\Delta E \sim 1/\beta$ , οπότε για τις περιπτώσεις  $\beta = 0$  και  $\beta > 0$  παίρνουμε ποιοτικά τη συμπεριφορά που φαίνεται στο σχήμα 12.4. Η κατανομή της απλής δειγματοληψίας αντι-



Σχήμα 12.4: Η πιθανότητα  $p(E)$  για θερμοκρασίες  $\beta = 0$  και  $\beta > 0$ . Οι δύο κατανομές έχουν ελάχιστη επικάλυψη.

στοιχεί στην περίπτωση  $\beta = 0$  (βλ. σχέση (12.4)). Για να υπολογίσουμε με ακρίβεια το άθροισμα (12.52) για  $\beta > 0$  πρέπει να πάρουμε καλό δείγμα στην περιοχή όπου το γινόμενο  $E p_{\beta > 0}(E)$  είναι σχετικά σημαντικό. Όπως βλέπουμε και στο σχήμα 12.4, η πιθανότητα να πάρουμε στο δείγμα κατάσταση με ενέργεια τέτοια ώστε το  $E p_{\beta > 0}(E)$  να είναι

σχετικά σημαντικό είναι πολύ μικρή, όταν η δειγματοληψία γίνεται με την κατανομή  $p_{\beta=0}(E)$ .

Παρόλο που η συγκεκριμένη μέθοδος έχει το μειονέκτημα αυτό, μπορεί να φανεί χρήσιμη σε ορισμένες περιπτώσεις. Την εφαρμόσαμε ήδη στην περίπτωση του τυχαίου περιπατητή. Παρατηρήστε επίσης ότι το δείγμα που παίρνουμε είναι ανεξάρτητο του  $\beta$  και χρησιμοποιώντας τη σχέση (12.51) υπολογίζουμε τις μέσες τιμές για κάθε  $\beta$ .

### 12.5.2 Importance Sampling

Από ότι είδαμε παραπάνω, ένα πολύ μικρό μέρος του χώρου των καταστάσεων δίνει σημαντική συνεισφορά στον υπολογισμό του  $\langle \mathcal{O} \rangle$ . Αν επιλέξουμε το δείγμα με πιθανότητα

$$P_\mu = p_\mu = \frac{e^{-\beta E_\mu}}{Z}, \quad (12.53)$$

περιμένουμε να δειγματοληψήσουμε ακριβώς μέσα στον υπόχωρο αυτό. Πράγματι, ο υπολογισμός του εκτιμητή (12.49) γίνεται με τη σχέση

$$\mathcal{O}_M = \frac{\sum_{i=1}^M \mathcal{O}_{\mu_i} (e^{-\beta E_{\mu_i}})^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^M (e^{-\beta E_{\mu_i}})^{-1} e^{-\beta E_{\mu_i}}} = \frac{1}{M} \sum_{i=1}^M \mathcal{O}_{\mu_i}. \quad (12.54)$$

Η παραπάνω δειγματοληψία λέγεται δειγματοληψία με κριτήριο σημαντικότητας (importance sampling) και είναι ο τρόπος ο οποίος χρησιμοποιείται για προσομοιώσεις στατιστικών συστημάτων στην κανονική συλλογή. Το δείγμα εξαρτάται από τη θερμοκρασία  $\beta$  και ο υπολογισμός των μέσων τιμών (12.54) απαιτεί καινούργια δειγματοληψία κάθε φορά σε αντίθεση με τη (12.51). Αυτή η δυσκολία όμως, στις περισσότερες περιπτώσεις, είναι πολύ μικρότερη από το πρόβλημα της επικάλυψης που συζητήθηκε στην παράγραφο 12.5.1.

## 12.6 Διαδικασίες Markov

Για να πάρουμε ένα δείγμα το οποίο κατανέμεται σύμφωνα με την κατανομή  $P_\mu$  δεν αρκεί να το κάνουμε απευθείας. Λ.χ. αν επιχειρήσουμε να δημιουργήσουμε δείγμα με  $P_\mu = \frac{e^{-\beta E_\mu}}{Z}$  επιλέγοντας τυχαία κατάσταση  $\mu$  και δεχόμαστε ή απορρίπτουμε την εισαγωγή στο δείγμα με πιθανότητα  $P_\mu$  θα έχουμε ελάχιστη πιθανότητα η κατάσταση να γίνει αποδεκτή στο δείγμα. Οπότε θα βρεθούμε στην ίδια δυσκολία που βρήκαμε στην περίπτωση της απλής δειγματοληψίας. Για το λόγο αυτό θα

χρησιμοποιήσουμε μια διαδικασία Markov για τη δημιουργία του δείγματος. Αυτή είναι μια διαδικασία όπου δεδομένου του συστήματος σε μια κατάσταση  $\mu$  παράγει με στοχαστικό τρόπο μια νέα κατάσταση  $\nu$ . Έτσι δημιουργείται μια αλυσίδα καταστάσεων

$$\mu_1 \rightarrow \mu_2 \rightarrow \dots \rightarrow \mu_M, \quad (12.55)$$

η οποία θα αποτελέσει το ζητούμενο δείγμα  $\{\mu_i\} \equiv \{\mu_1, \mu_2, \dots, \mu_M\}$ . Φανταζόμαστε ότι η επιλογή της κατάστασης  $\mu_i$  γίνεται στο “χρόνο”  $i$ . Η πιθανότητα μετάβασης  $P(\mu \rightarrow \nu)$  (transition probability) στην κατάσταση  $\nu$ , όταν το σύστημα είναι στην κατάσταση  $\mu$  πρέπει να ικανοποιεί τις παρακάτω συνθήκες:

1. Είναι ανεξάρτητη του “χρόνου”.
2. Εξαρτάται μόνο από τις καταστάσεις  $\mu, \nu$  και όχι από τη διαδρομή που κάναμε μέχρι την κατάσταση  $\mu$ .
3. Ικανοποιείται η σχέση

$$\sum_{\nu} P(\mu \rightarrow \nu) = 1. \quad (12.56)$$

Προσοχή, συνήθως  $P(\mu \rightarrow \mu) > 0$  και το σύστημα έχει πιθανότητα να παραμείνει στην ίδια κατάσταση.

4. Για  $M \rightarrow \infty$  το δείγμα  $\{\mu_i\}$  ακολουθεί την κατανομή  $P_{\mu}$ .

Η προσομοίωση Μόντε Κάρλο με τον τρόπο αυτό γίνεται επιλέγοντας κατάλληλα μια αρχική κατάσταση  $\mu_1$  για το σύστημα και εφαρμόζοντας τον παραπάνω αλγόριθμο. Η μεγαλύτερη προσπάθεια επικεντρώνεται στον προσδιορισμό των πιθανοτήτων μετάβασης  $P(\mu \rightarrow \nu)$ , έτσι ώστε η σύγκλιση 4 να επιτυγχάνεται γρήγορα.

Σημαντική είναι και η επιλογή της αρχικής κατάστασης  $\mu_1$ . Αν αυτή δεν είναι μια τυπική κατάσταση του τελικού δείγματος θα πρέπει να περάσει κάποιος χρόνος μέχρι το σύστημα να βρεθεί σε “κατάσταση ισορροπίας” όπου πια η διαδικασία Markov δειγματοληπτεί μέσα στη σωστή κατανομή. Ο χρόνος που απαιτείται (thermalization time) μπορεί να γίνει σημαντικό μέρος της προσπάθειάς μας, αν γίνει λάθος επιλογή της  $\mu_1$  ή/και των  $P(\mu \rightarrow \nu)$ .

Απαραίτητη προϋπόθεση για να πετύχουμε το δείγμα να ακολουθεί την ζητούμενη κατανομή σε μια τέτοια διαδικασία είναι να ικανοποιεί το κριτήριο της **εργοδικότητας**. Αυτό σημαίνει ότι από κάθε κατάσταση

$\mu$  που επιλέγουμε, κάθε άλλη δυνατή κατάσταση  $\nu$  είναι προσβάσιμη μέσω της διαδικασίας με ένα πεπερασμένο αριθμό από βήματα. Αν αυτό το κριτήριο δεν ικανοποιείται και υπάρχουν σημαντικές περιοχές του χώρου των καταστάσεων στις οποίες δεν μπορούμε να δειγματοληπτήσουμε, δεν θα είναι δυνατόν να πετύχουμε τη ζητούμενη κατανομή. Στην πράξη, επειδή δεδομένης της  $\mu$  οι καταστάσεις  $\nu$  για τις οποίες  $P(\mu \rightarrow \nu) > 0$  είναι ελάχιστες, πρέπει να είμαστε ιδιαίτερα προσεκτικοί, ώστε ο αλγόριθμος που επιλέγουμε να μην παραβιάζει τη συνθήκη της εργοδικότητας<sup>9</sup>.

## 12.7 Συνθήκη Λεπτομερούς Ισορροπίας

Από την εξίσωση (12.2) μπορούμε εύκολα να καταλάβουμε ότι για να βρεθεί το σύστημα σε κατάσταση ισορροπίας στην κατανομή  $p_\mu$ , οι πιθανότητες μετάβασης πρέπει να ικανοποιούν τη σχέση

$$\sum_{\nu} p_{\mu} P(\mu \rightarrow \nu) = \sum_{\mu} p_{\nu} P(\nu \rightarrow \mu). \quad (12.57)$$

Αυτό σημαίνει ότι ο ρυθμός με τον οποίο το σύστημα μεταβαίνει από την κατάσταση  $\mu$  σε κάποια άλλη, είναι ίσος με το ρυθμό με τον οποίο το σύστημα μεταβαίνει στην κατάσταση  $\mu$  από κάποια άλλη. Προφανώς, η σχέση (12.56) μας δίνει

$$p_{\mu} = \sum_{\mu} p_{\nu} P(\nu \rightarrow \mu). \quad (12.58)$$

Η παραπάνω συνθήκη είναι αναγκαία αλλά δεν είναι ικανή (δες κεφάλαιο 2.2.3 του [4]). Μια ικανή, αλλά όχι αναγκαία, συνθήκη είναι η συνθήκη λεπτομερούς ισορροπίας (detailed balance condition) η οποία όταν ικανοποιείται από τις πιθανότητες μετάβασης, τότε το σύστημα αργά ή γρήγορα θα φτάσει σε κατάσταση θερμικής ισορροπίας

$$p_{\mu} P(\mu \rightarrow \nu) = p_{\nu} P(\nu \rightarrow \mu). \quad (12.59)$$

<sup>9</sup>Στην πράξη υπάρχουν αλγόριθμοι για του οποίους έχουμε παραβίαση της εργοδικότητας, αλλά επειδή οι καταστάσεις που δεν είναι προσβάσιμες αποτελούν σύνολο “μέτρου μηδέν” στο χώρο των καταστάσεων, η παραβίαση δεν επηρεάζει τα αποτελέσματά μας. Αντίθετα, υπάρχουν περιπτώσεις όπου η συνθήκη δεν παραβιάζεται αλλά η πιθανότητα να φτάσει κανείς σε κάποιες περιοχές του χώρου των καταστάσεων είναι στην πράξη απαγορευτικά μικρή. Αυτό για παράδειγμα μπορεί να συμβεί κοντά σε μια μετάβαση φάσης πρώτης τάξης όπου το σύστημα δυσκολεύεται να περάσει από καταστάσεις της μιας φάσης στην άλλη.

Αθροίζοντας και τα δύο μέλη της (12.59) προκύπτει η συνθήκη ισορροπίας (12.57). Για την κατανομή της κανονικής συλλογής (12.4) έχουμε

$$\frac{P(\mu \rightarrow \nu)}{P(\nu \rightarrow \mu)} = \frac{p_\nu}{p_\mu} = e^{-\beta(E_\nu - E_\mu)}. \quad (12.60)$$

Μπορεί να δειχτεί ότι, αν οι πιθανότητες μετάβασης ικανοποιούν τις παραπάνω συνθήκες, η κατανομή ισορροπίας του συστήματος θα είναι η κατανομή Boltzmann (12.4). Το πρόγραμμα της προσομοίωσης Μόντε Κάρλο μπορεί να συνοψιστεί στα επόμενα βήματα:

1. Γράφουμε λογισμικό το οποίο κωδικοποιεί κατάλληλα επιλεγμένες πιθανότητες μετάβασης  $P(\mu \rightarrow \nu)$  που ικανοποιούν την (12.60).
2. Επιλέγουμε κατάλληλη αρχική κατάσταση  $\mu_1$ .
3. Αφήνουμε το σύστημα να εξελιχθεί μέχρι να προσεγγίσουμε την κατανομή Boltzmann (12.4) (thermalization).
4. Συλλέγουμε δεδομένα για τις παρατηρήσιμες ποσότητες  $\mathcal{O}$  και υπολογίζουμε τον εκτιμητή  $\mathcal{O}_M$  με την (12.54).
5. Σταματάμε μόλις πετύχουμε την επιθυμητή ακρίβεια.

Η εξίσωση (12.60) έχει πολλές λύσεις. Το ποια θα επιλέξουμε εξαρτάται από την αποδοτικότητά τους σε ένα συγκεκριμένο πρόβλημα και πρέπει να εξεταστεί προσεκτικά κατά περίπτωση. Παραδείγματα τέτοιων επιλογών είναι:

$$P(\mu \rightarrow \nu) = A \cdot e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}, \quad (12.61)$$

$$P(\mu \rightarrow \nu) = A \cdot \frac{e^{-\beta(E_\nu - E_\mu)}}{1 + e^{-\beta(E_\nu - E_\mu)}}, \quad (12.62)$$

$$P(\mu \rightarrow \nu) = A \cdot \begin{cases} e^{-\beta(E_\nu - E_\mu)} & E_\nu - E_\mu > 0 \\ 1 & E_\nu - E_\mu \leq 0 \end{cases}, \quad (12.63)$$

για κατάλληλα επιλεγμένες καταστάσεις  $\nu \neq \mu$ ,

$$P(\mu \rightarrow \mu) = 1 - \sum_{\nu} P(\mu \rightarrow \nu), \quad (12.64)$$

ενώ  $P(\mu \rightarrow \nu') = 0$  για οποιαδήποτε άλλη κατάσταση  $\nu'$ . Οι σταθερές  $A$  πρέπει να επιλεγούν κατάλληλα, έτσι ώστε

$$\sum_{\nu \neq \mu} P(\mu \rightarrow \nu) < 1 \quad (12.65)$$

για να έχει νόημα η (12.64).

Η σχέση (12.65) μας δίνει μεγάλη ελευθερία στην επιλογή των πιθανοτήτων μετάβασης. Στην πράξη οι  $P(\mu \rightarrow \nu)$  σπάνε σε δύο κομμάτια

$$P(\mu \rightarrow \nu) = g(\mu \rightarrow \nu) A(\mu \rightarrow \nu), \quad (12.66)$$

τα οποία αντιστοιχούν σε διακριτά βήματα στον αλγόριθμο.

Η πιθανότητα  $g(\mu \rightarrow \nu)$  είναι η **πιθανότητα επιλογής** (selection probability) της κατάστασης  $\nu$ , όταν το σύστημα βρίσκεται στην κατάσταση  $\mu$ . Το πρώτο βήμα δηλαδή είναι να επιλέξουμε μια κατάσταση  $\nu \neq \mu$  με πιθανότητα  $g(\mu \rightarrow \nu)$ .

Το δεύτερο βήμα είναι να επιλέξουμε με πιθανότητα  $A(\mu \rightarrow \nu)$ , αν το σύστημα θα μεταβεί στην κατάσταση  $\nu$ . Αν η απάντηση είναι όχι, τότε παραμένουμε στην κατάσταση  $\mu$ . Με τον τρόπο αυτό ικανοποιείται η σχέση (12.64). Οι πιθανότητες  $A(\mu \rightarrow \nu)$  ονομάζονται **λόγοι αποδοχής**.

Ο στόχος μας επικεντρώνεται στην εύρεση αλγόριθμου τέτοιου, ώστε οι πιθανότητες επιλογής να δίνουν τους μέγιστους δυνατούς λόγους αποδοχής για καταστάσεις  $\nu$  ασυσχέτιστες κατά το μέγιστο δυνατόν από την κατάσταση  $\mu$ . Ιδανική περίπτωση είναι να έχω  $A(\mu \rightarrow \nu) = 1$  για όλα τα  $\nu$  για τα οποία  $g(\mu \rightarrow \nu) > 0$ . Αυτό συμβαίνει για παράδειγμα στους cluster αλγόριθμους συστημάτων σπιν (λ.χ. πρότυπα Ising, Potts) όπως ο αλγόριθμος του Wolff που θα μελετήσουμε αργότερα.

## 12.8 Ασκήσεις

1. Αποδείξτε τη σχέση (12.18).
2. Αποδείξτε τη σχέση (12.19).
3. Αποδείξτε τη σχέση (12.45).
4. Αποδείξτε ότι οι σχέσεις (12.61)–(12.63) ικανοποιούν την (12.60).



## ΚΕΦΑΛΑΙΟ 13

### Το Πρότυπο Ising

Στο Κεφάλαιο αυτό θα γίνει εισαγωγή των βασικών μεθόδων Μόντε Κάρλο για την προσομοίωση του πρότυπου Ising στο δισδιάστατο τετραγωνικό πλέγμα. Αρχικά, η μελέτη θα γίνει με τον αλγόριθμο Metropolis όπου θα μελετηθεί η διαδικασία που φέρνουμε το σύστημα σε κατάσταση θερμικής ισορροπίας (thermalization), καθώς και η διαδικασία με την οποία παίρνουμε στατιστικά ανεξάρτητες μετρήσεις μελετώντας τους χρόνους αυτοσυσχετισμού (autocorrelation times) του συστήματος. Η κατανόηση των βασικών εννοιών είναι θεμελιώδης, αφού τα παραπάνω είναι οι κύριες τεχνικές δυσκολίες που αντιμετωπίζουμε όταν προσπαθούμε να μελετήσουμε συστήματα με μεγάλο αριθμό βαθμών ελευθερίας.

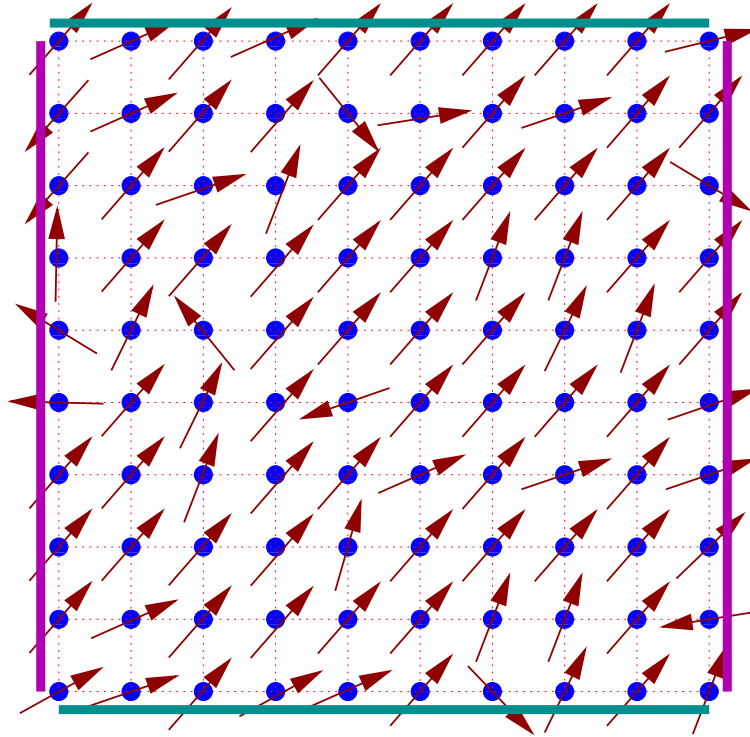
#### 13.1 Εισαγωγή

Το πρότυπο Ising (1925) [54] έχει παίξει ιστορικά πολύ σημαντικό ρόλο στις εξελίξεις των ιδεών της στατιστικής φυσικής και της κβαντικής θεωρίας πεδίου. Ειδικότερα στις δύο διαστάσεις, το πρότυπο είναι αρκετά σύνθετο, ώστε να έχει ενδιαφέρουσες μη τετριμμένες ιδιότητες και αρκετά απλό, ώστε να μπορούμε να αντλήσουμε αναλυτικά πολύτιμες πληροφορίες για τη φυσική των μεταβάσεων φάσης. Το πρότυπο εκδηλώνει μετάβαση φάσης 2ης τάξης με αποτέλεσμα να μπορούν να μελετηθούν οι ιδιότητες των συνεχών μεταβάσεων φάσης (κρίσιμοι εκθέτες, παγκοσμιότητα, ομάδα επανακανονικοποίησης, συνεχές όριο). Με την αναλυτική λύση<sup>1</sup> του Onsager (1948) [55] και άλλων παίρνουμε αποτελέσματα με τα οποία μπορούμε να ελέγξουμε προσεγγιστικές μεθόδους,

---

<sup>1</sup>Για μια πολύ καλή συζήτηση της λύσης του Onsager, δείτε το βιβλίο του T. Huang [56] και την εργασία του C.N. Yang [57].

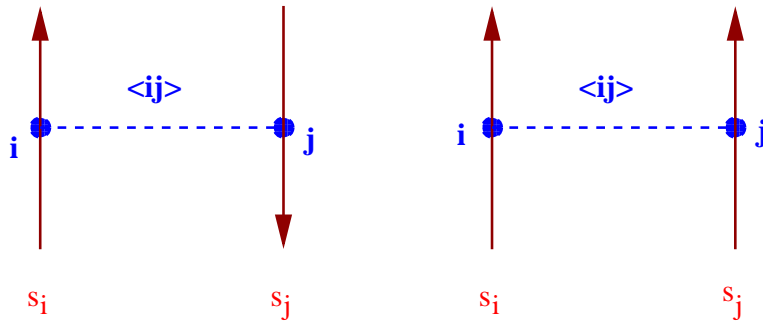
όπως οι προσομοιώσεις Μόντε Κάρλο, αναπτύγματα μεγάλης και μικρής θερμοκρασίας, mean field theory κλπ. Επιπλέον έχει και φυσικό ενδιαφέρον, επειδή είναι ένα πρότυπο ενός μαγνητικού υλικού που, παρόλη την απλότητά του, περιγράφει ποιοτικά πολλές από τις μη τετριμμένες ιδιότητες τους. Επίσης, λόγω της παγκοσμιότητας μπορεί να θεωρηθεί πρότυπο και για τη μετάβαση φάσης σε υγρό-αέριο (τριπλό σημείο). Εξαιρετο βιβλίο για πρότυπα της στατιστικής φυσικής που λύνονται επακριβώς με αναλυτικές μεθόδους αποτελεί το βιβλίο του Baxter [53]. Για τον ορισμό του προτύπου θεωρήστε ένα δισδιάστατο τετραγωνικό



Σχήμα 13.1: Δισδιάστατο τετραγωνικό πλέγμα του οποίου οι θέσεις  $i = 1, \dots, N$  καταλαμβάνονται από “άτομα” ή “μαγνητάκια” με σπιν  $s_i$  τα οποία έχουν τυχαίο προσανατολισμό στο επίπεδο (πρότυπο XY). Συνήθως, λαμβάνονται υπόψη μόνο οι αλληλεπιδράσεις πλησιέστερων γειτόνων  $-J\vec{s}_i \cdot \vec{s}_j$  για κάθε σύνδεσμο  $\langle ij \rangle$ . Η τοπολογία είναι τοροϊδής (toroidal), όταν ταυτίσουμε τις οριζόντιες πλευρές μεταξύ τους και τις κάθετες μεταξύ τους, δηλ. τα ίδια χρώματα στο σχήμα.

πλέγμα, όπως αυτό του σχήματος 13.1 στο οποίο τοποθετείτε σε κάθε πλεγματική θέση (node) “άτομα” ή “μαγνητάκια” με σπιν  $s_i$ . Η γεωμετρία καθορίζεται από την ελάχιστη απόσταση μεταξύ των γειτόνων, την πλεγματική σταθερά  $a$ , και τον αριθμό των πλεγματικών θέσεων  $N$ . Σε

κάθε πλευρά έχουμε  $L$  πλεγματικές θέσεις, έτσι ώστε  $N = L \times L = L^d$ , με  $d = 2$  τη διάσταση του χώρου. Η τοπολογία καθορίζεται από τις σχέσεις γειτονίας, και ιδιαίτερα από τις σχέσεις γειτονίας των πλεγματικών θέσεων που βρίσκονται στο σύνορο του τετραγώνου. Η τοροϊδής τοπολογία λαμβάνεται, αν δημιουργήσουμε δεσμό μεταξύ των γειτόνων που ανήκουν στις κάθετες και οριζόντιες πλευρές του τετραγώνου του σχήματος (13.1). Η δυναμική του συστήματος καθορίζεται από τη μαγνητική αλληλεπίδραση (spin-spin interaction) την οποία απλοποιούμε, ώστε να είναι κοντινής εμβέλειας και ιδιαίτερα μόνο μεταξύ των πλησιέστερων γειτόνων. Στο σιδηρομαγνητικό πρότυπο Ising θεωρούμε ότι



Σχήμα 13.2: Τα σπιν στο πρότυπο Ising παίρνουν δύο μόνο τιμές “πάνω” και “κάτω” και η ενέργεια του συστήματος προκύπτει από τη συνεισφορά κάθε συνδέσμου (link)  $\langle ij \rangle$ . Αυτή για το σιδηρομαγνητικό πρότυπο παίρνει δύο δυνατές τιμές  $+J$  και  $-J$  με  $J > 0$  για αντίρροπα και ομόρροπα σπιν αντίστοιχα. Το σύστημα έχει τη συμμετρία της διακριτής ομάδας  $Z_2$ .

οι δυνατές τιμές των σπιν είναι δύο,  $+1$  ή  $-1$ . Στην πιο απλή περίπτωση, η αλληλεπίδραση είναι αλληλεπίδραση σπιν-σπιν μόνο μεταξύ πλησιέστερων γειτόνων με την κατάσταση που έχει τα σπιν ομόρροπα να βρίσκεται στην κατάσταση χαμηλότερης ενέργειας<sup>2</sup>. Αυτό απεικονίζεται στο σχήμα 13.2. Το σύστημα μπορεί να βρίσκεται υπό την επίδραση ομογενούς μαγνητικού πεδίου  $B$  του οποίου η διεύθυνση θεωρείται ότι είναι παράλληλη ή αντιπαράλληλη με αυτή των σπιν. Είμαστε τώρα έτοιμοι να γράψουμε την Hamiltonian και τη συνάρτηση επιμερισμού του συστήματος.

Θεωρούμε τετραγωνικό πλέγμα από  $N$  πλεγματικές θέσεις (sites ή vertices) διατεταγμένες σε τετράγωνο με πλευρές από  $L$  πλεγματικές θέσεις τις οποίες απαριθμούμε με αριθμούς  $i = 1, 2, \dots, N$ . Το πλέγμα έχει  $N_l$  δεσμούς (bonds ή links) μεταξύ των πλησιέστερων γειτόνων.

<sup>2</sup>Το αντίστροφο ισχύει για το αντισιδηρομαγνητικό πρότυπο Ising.

Αυτοί χαρακτηρίζονται από τα ζεύγη των πλεγματικών θέσεων  $i, j$  που ενώνουν και θα τους συμβολίζουμε με  $\langle ij \rangle$ . Ταυτίζουμε τις πλευρές του τετραγώνου, όπως στο σχήμα 13.1. Επειδή τότε κάθε δεσμός ενώνει ακριβώς δύο πλεγματικές θέσεις και από κάθε πλεγματική θέση ξεκινούν ακριβώς τέσσερις δεσμοί, θα ισχύει

$$2N_l = 4N \Rightarrow N_l = 2N. \quad (13.1)$$

Σε κάθε πλεγματική θέση τοποθετούμε σπιν  $s_i = \pm 1$ .

Η Hamiltonian του συστήματος θα δίνεται από τη σχέση

$$H = -J \sum_{\langle ij \rangle} s_i s_j - B \sum_i s_i. \quad (13.2)$$

Ο πρώτος όρος δίνει την αλληλεπίδραση μεταξύ των σπιν και για  $J > 0$  – το οποίο υποθέτουμε στο βιβλίο αυτό – το σύστημα είναι σιδηρομαγνητικό. Κάθε δεσμός που ενώνει ομόροπα σπιν έχει ενέργεια  $-J$  που είναι κατά  $2J$  μικρότερη από έναν δεσμό με αντίροπα σπιν. Το σύστημα ενεργειακά προτιμά καταστάσεις με δεσμούς ομόροπους, δηλ. τα σπιν να είναι όλα προσανατολισμένα προς την ίδια κατεύθυνση. Η ελάχιστη ενέργεια αντιστοιχεί στη μοναδική<sup>3</sup> κατάσταση με όλα τα σπιν να κοιτούν προς την κατεύθυνση του  $B$ , την θεμελιώδη κατάσταση<sup>4</sup>. Η ενέργειά της είναι

$$E_0 = -JN_l - BN = -(2J + B)N. \quad (13.3)$$

Η συνάρτηση επιμερισμού είναι

$$Z = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \dots \sum_{s_N=\pm 1} e^{-\beta H[\{s_i\}]} \equiv \sum_{\{s_i\}} e^{\beta J \sum_{\langle ij \rangle} s_i s_j + \beta B \sum_i s_i}, \quad (13.4)$$

όπου  $\{s_i\} \equiv \{s_1, s_2, \dots, s_N\}$  είναι μια διάταξη (configuration) των σπιν στο πλέγμα. Ο αριθμός των όρων του αθροίσματος είναι ίσος με τον αριθμό των δυνατών διατάξεων  $\{s_i\}$  των σπιν που είναι  $2^N$ , δηλ. αυξάνουν εκθετικά με το  $N$ . Για ένα  $5 \times 5$  πλέγμα, ο αριθμός των όρων είναι  $2^{25} \approx 3.4 \times 10^6$ .

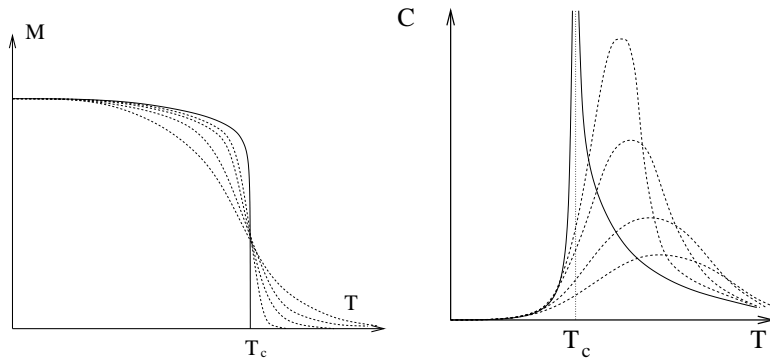
<sup>3</sup>Στην περίπτωση του αντισιδηρομαγνητικού συστήματος  $J < 0$ , η θεμελιώδης ενέργεια είναι εκφυλισμένη και οι αντίστοιχες καταστάσεις είναι πιο πολύπλοκο να περιγραφούν.

<sup>4</sup>Για  $B = 0$ , το σύστημα έχει “πάνω–κάτω” συμμετρία (συμμετρία  $Z_2$ ). Δύο καταστάσεις που προκύπτουν με την εφαρμογή της συμμετρίας αυτής (αντιστροφή όλων των σπιν) ταυτίζονται.

Το δισδιάστατο πρότυπο Ising με  $B = 0$  έχει την εξαιρετικά ενδιαφέρουσα ιδιότητα ότι για  $\beta = \beta_c$ , όπου

$$\beta_c = \frac{1}{2} \ln(1 + \sqrt{2}) \approx 0.4406867935 \dots \quad (13.5)$$

το σύστημα παρουσιάζει μετάβαση φάσης από τη διατεταγμένη φάση ή φάση χαμηλής θερμοκρασίας, όπου το σύστημα είναι μαγνητισμένο ( $\langle |M| \rangle > 0$ ) στην άτακτη φάση ή φάση υψηλής θερμοκρασίας, όπου η μαγνήτιση χάνεται ( $\langle |M| \rangle = 0$ ). Η θερμοκρασία  $\beta_c$  είναι η κρίσιμη θερμοκρασία, η θερμοκρασία Curie. Η μετάβαση φάσης είναι συνεχής και δευτέρας τάξης, γιατί η συνάρτηση  $\langle |M| \rangle(\beta)$  είναι συνεχής για  $\beta = \beta_c$ , αλλά όχι η παράγωγός της. Αυτό φαίνεται ποιοτικά στο σχήμα 13.3. Επειδή η τιμή της  $\langle |M| \rangle$  ξεχωρίζει τις δύο φάσεις του συστήματος, η  $\langle |M| \rangle$  λέγεται παράμετρος τάξης. Για  $\beta \neq \beta_c$ , η συνάρτηση συσχετι-



Σχήμα 13.3: Ποιοτική συμπεριφορά της μαγνήτισης (αριστερά) και της ειδικής θερμότητας (δεξιά) στη μετάβαση φάσης στο πρότυπο Ising. Με συνεχή γραμμή φαίνεται η (μη αναλυτική) συμπεριφορά στο θερμοδυναμικό όριο, ενώ με διακεκομμένες η συμπεριφορά για αυξανόμενο, αλλά πεπερασμένο μέγεθος  $N$  του συστήματος. Οι τελευταίες συγκλίνουν ομοιόμορφα προς τις μη αναλυτικές συναρτήσεις.

σμού (12.45) συμπεριφέρεται όπως στη σχέση (12.46) με πεπερασμένο μήκος συσχετισμού  $\xi(\beta)$ . Καθώς πλησιάζουμε την κρίσιμη θερμοκρασία το μήκος συσχετισμού τείνει στο άπειρο και μάλιστα συμπεριφέρεται ασυμπτωτικά σαν

$$\xi(\beta) \equiv \xi(t) \sim |t|^{-\nu} \quad t = \frac{\beta_c - \beta}{\beta_c}. \quad (13.6)$$

Η συνάρτηση συσχετισμού συμπεριφέρεται σύμφωνα με την εξίσωση (12.47)

$$G_c^{(2)}(i, j) \sim \frac{1}{|x_{ij}|^\eta}. \quad (13.7)$$

ενώ παρόμοια συμπεριφορά βάθμισης (scaling) παρουσιάζουν και η ειδική θερμότητα  $C$ , μαγνήτιση  $M \equiv \langle M \rangle$  και μαγνητική επιδεκτικότητα  $\chi$  σύμφωνα με τις σχέσεις

$$C \sim |t|^{-\alpha} \quad (13.8)$$

$$M \sim |t|^\beta \quad (13.9)$$

$$\chi \sim |t|^{-\gamma}. \quad (13.10)$$

ενώ η μαγνήτιση για  $t = 0$  και το μαγνητικό πεδίο  $B \neq 0$ , συμπεριφέρεται σύμφωνα με τη σχέση

$$M \sim B^{-1/\delta}. \quad (13.11)$$

Οι εκθέτες που παρουσιάζονται στις παραπάνω σχέσεις λέγονται κρίσιμοι εκθέτες ή εκθέτες βάθμισης (critical ή scaling exponents) και οι τιμές τους παρουσιάζουν την ιδιότητα της παγκοσμιότητας. Δηλ. οι τιμές τους δεν εξαρτώνται από τις λεπτομέρειες του πλέγματος (τετραγωνικό, τριγωνικό κλπ) ή της αλληλεπίδρασης (πλησιέστερων ή μη γειτόνων, μεγαλύτερες δυνάμεις του σπιν κλπ) και μία ολόκληρη κλάση από πρότυπα έχουν την ίδια συμπεριφορά! Απαραίτητη προϋπόθεση είναι τα πρότυπα αυτά να έχουν τις ίδιες ιδιότητες συμμετρίας, να ορίζονται σε χώρο ίδιας διάστασης και οι αλληλεπιδράσεις να είναι κοντινής εμβέλειας. Στο συγκεκριμένο πρότυπο οι εκθέτες παίρνουν τις λεγόμενες τιμές Onsager

$$\alpha = 0, \quad \beta = \frac{1}{8}, \quad \gamma = \frac{7}{4} \quad (13.12)$$

$$\delta = 15, \quad \nu = 1, \quad \eta = \frac{1}{4}$$

Η συμπεριφορά (13.6–13.11) χαρακτηρίζεται από τη μη αναλυτικότητα των αντίστοιχων συναρτήσεων. Αυτή δεν είναι δυνατόν να προκύψει από τη συνάρτηση επιμερισμού (13.4) για πλέγμα με  $N$  θέσεις, αφού σε ένα πεπερασμένο άθροισμα από εκθετικά είναι αναγκαστικά αναλυτική συνάρτηση (άρα και οι παράγωγοί της). Η μη αναλυτική συμπεριφορά φανερώνεται στο όριο απείρου μεγέθους (θερμοδυναμικό όριο) όπου οι παραπάνω συναρτήσεις τείνουν προς μια μη-αναλυτική συνάρτηση όπως στο σχήμα 13.3. Η απώλεια της αναλυτικότητας οφείλεται στους συσχετισμούς των σπιν σε μακροσκοπικές αποστάσεις.

Σε πολλά συστήματα που μελετάμε τα παραπάνω μη αναλυτικά σημεία, αναζητούμε μια παράμετρο τάξης (order parameter) η οποία χαρακτηρίζει τη συμμετρία του συστήματος. Στη συγκεκριμένη περίπτωση του πρότυπου Ising, η παράμετρος τάξης είναι η μαγνήτιση με τη συμμετρία  $s_i \rightarrow -s_i$ . Συνήθως, στη μία φάση η παράμετρος τάξης είναι μη

μηδενική, ενώ στην άλλη μηδενίζεται. Αυτό συνεπάγεται μη αναλυτική συμπεριφορά, αφού μια αναλυτική συνάρτηση που είναι μηδενική σε ένα διάστημα, είναι παντού μηδέν.

Η συμπεριφορά παγκοσμιότητας ή ανεξαρτησίας κλίμακας (scale invariance) παρουσιάζεται κάθε φορά που έχουμε απόκλιση του μήκους συσχετισμού  $\xi \rightarrow \infty$ . Στην περίπτωση μας, για να φτάσουμε το κρίσιμο σημείο, έχουμε να ρυθμίσουμε μόνο μια παράμετρο, τη θερμοκρασία, οπότε αναμένει κανείς να εμφανιστεί μόνο μια καινούργια κλίμακα στο πρότυπο. Ανεξαρτησία κλίμακας εμφανίζεται, όταν το μήκος συσχετισμού γίνει πολύ μεγαλύτερο από τη μικροσκοπική κλίμακα  $a$ , οπότε οποιαδήποτε ποσότητα που είναι συνάρτηση της απόστασης  $r$  μπορεί να εξαρτάται μόνο από το λόγο  $r/\xi$ . Η παγκοσμιότητα προκύπτει από το γεγονός ότι, στην περίπτωση αυτή, φαίνεται ότι τα πάντα εξαρτώνται από εκείνες τις διακυμάνσεις με μεγάλο μήκος κύματος που απαιτούνται από τη συμμετρία της παραμέτρου τάξης. Η σημαντική απλούστευση που προκύπτει είναι ότι για να μελετήσουμε ένα πραγματικό φυσικό σύστημα σε μια συνεχή μετάβαση φάσης  $\xi \rightarrow \infty$ , αρκεί να μελετήσουμε το απλούστερο πρότυπο με τη δεδομένη συμμετρία και χωρικές διαστάσεις.

## 13.2 Ο Αλγόριθμος Metropolis

Θεωρούμε την περίπτωση του τετραγωνικού πλέγματος με  $L$  πλεγματικές θέσεις σε κάθε πλευρά, έτσι ώστε  $N = L \times L = L^2$  να είναι ο συνολικός αριθμός πλεγματικών θέσεων και  $N_l = 2N$  να είναι ο συνολικός αριθμός των “δεσμών” μεταξύ των πλησιέστερων γειτόνων. Η τελευταία σχέση ισχύει, επειδή επιλέγουμε ελικοειδείς συνοριακές συνθήκες στο πλέγμα, όπως στο σχήμα 13.6, όπως θα αναλύσουμε στην επόμενη παράγραφο. Σε κάθε πλεγματική θέση  $i$  έχουμε ένα βαθμό ελευθερίας, το “σπιν”  $s_i$  το οποίο παίρνει δύο τιμές, έστω  $\pm 1$ . Θεωρούμε το σύστημα για την περίπτωση του μηδενικού μαγνητικού πεδίου  $B = 0$ , οπότε η Hamiltonian του συστήματος θα δίνεται από τη σχέση<sup>5</sup>

$$H = - \sum_{\langle ij \rangle} s_i s_j. \quad (13.13)$$

Θυμίζουμε πως  $\sum_{\langle ij \rangle}$  είναι άθροισμα πάνω στους δεσμούς  $\langle i, j \rangle$ , δηλ. πάνω στα ζεύγη γειτονικών πλεγματικών θέσεων.  $\sum_{\langle i, j \rangle} = (1/2) \sum_{i=1}^N \sum_{j=1}^N$  αφού στη δεύτερη περίπτωση κάθε δεσμός αθροίζεται δύο φορές, μία

<sup>5</sup>Η σταθερά  $J = 1$  με κατάλληλη επιλογή μονάδων για τα  $s_i$ .

από κάθε πλεγματική θέση που βρίσκεται στα άκρα του. Η συνάρτηση επιμερισμού είναι η

$$Z = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \dots \sum_{s_N=\pm 1} e^{-\beta H[\{s_i\}]} \equiv \sum_{\{s_i\}} e^{\beta \sum_{\langle ij \rangle} s_i s_j}. \quad (13.14)$$

Στόχος μας είναι η δημιουργία ενός δείγματος του συστήματος που κατανέμεται σύμφωνα με την κατανομή Boltzmann (12.4) και θα το δημιουργήσουμε φτιάχνοντας μια αλυσίδα Markov σύμφωνα με αυτά που ειπώθηκαν στην παράγραφο 12.6. Η δειγματοληψία θα γίνει σύμφωνα με την (12.53) και οι μέσες τιμές θα υπολογίζονται από το δείγμα από την (12.54). Επιλέγοντας σε κάθε βήμα της διαδικασίας την επόμενη κατάσταση του συστήματος σύμφωνα με την (12.60), το δείγμα μας θα βρίσκεται προσεγγιστικά στην επιθυμητή κατανομή για αρκετά μεγάλο πλήθος στοιχείων. Αυτό επιτυγχάνεται κάνοντας έναν αρκετά μεγάλο αριθμό βημάτων (“χρόνο”) στην διαδικασία Markov.

Έστω ότι το σύστημά μας βρίσκεται αρχικά στην κατάσταση  $\mu^6$ . Η πιθανότητα να βρεθεί στο επόμενο βήμα της διαδικασίας στην κατάσταση  $\nu$ , σύμφωνα με την (12.66) είναι

$$P(\mu \rightarrow \nu) = g(\mu \rightarrow \nu) A(\mu \rightarrow \nu), \quad (13.15)$$

όπου  $g(\mu \rightarrow \nu)$  είναι η **πιθανότητα επιλογής** της κατάστασης  $\nu$ , όταν το σύστημα βρίσκεται στην κατάσταση  $\mu$  και  $A(\mu \rightarrow \nu)$  ο **λόγος αποδοχής**, δηλ. η πιθανότητα τότε το σύστημα να μεταβεί στη νέα κατάσταση. Αν ικανοποιείται η συνθήκη λεπτομερούς ισορροπίας (12.60)

$$\frac{P(\mu \rightarrow \nu)}{P(\nu \rightarrow \mu)} = \frac{g(\mu \rightarrow \nu) A(\mu \rightarrow \nu)}{g(\nu \rightarrow \mu) A(\nu \rightarrow \mu)} = e^{-\beta(E_\nu - E_\mu)}, \quad (13.16)$$

τότε η κατανομή στο δείγμα μας θα συγκλίνει στην (12.4)  $p_\mu = e^{-\beta E_\mu} / Z$ . Για να είναι οι πιθανότητες  $P(\mu \rightarrow \nu)$  αρκετά μεγάλες, ώστε το σύστημα να αλλάζει συχνά κατάσταση, οι μεταβολές στην ενέργεια  $E_\nu - E_\mu$  δε θα πρέπει να είναι πολύ μεγάλες. Θα πρέπει το γινόμενο με τη θερμοκρασία να είναι αριθμός της τάξης της μονάδας. Ένας τρόπος να το πετύχουμε αυτό είναι να εφαρμόσουμε έναν αλγόριθμο που σε κάθε βήμα θα μεταβάλλει την τιμή ενός από τα σπιν του πλέγματος  $s_i = \pm 1 \rightarrow s'_i \mp 1$ . Επειδή η ενέργεια (13.13) είναι τοπική ποσότητα, η μεταβολή της θα είναι μικρή. Πιο συγκεκριμένα, αν κάθε πλεγματική θέση έχει  $z = 4$  πλησιέστερους γείτονες, με μια τέτοια μεταβολή το πρόσημο του όρου

<sup>6</sup>Η κατάσταση  $\mu$  εδώ ορίζεται από μία συγκεκριμένη διάταξη των  $N$  σπινς  $\{s_i\}_{i=1\dots N}$ .



$s_i s_j$  αλλάζει για  $z$  όρους της (13.13), αυτούς που έχουν στο ένα άκρο τους τη θέση  $i$ . Η μεταβολή της ενέργειας είναι  $\pm 2$  για κάθε δεσμό. Αν η κατάσταση  $\mu$  ορίζεται να είναι η  $\{s_1, \dots, s_i, \dots, s_N\}$  και η κατάσταση  $\nu$  είναι η  $\{s_1, \dots, s'_i, \dots, s_N\}$  (δηλ. με όλα τα σπιν ίδια, εκτός από το  $s_i$  το οποίο αλλάζει πρόσημο), τότε η μεταβολή της ενέργειας θα είναι

$$|\Delta E| \leq 2z \Leftrightarrow E_\mu - 2z \leq E_\nu \leq E_\mu + 2z. \quad (13.17)$$

Αν επιλέξουμε τη θέση  $i$  τυχαία, τότε

$$g(\mu \rightarrow \nu) = g(\nu \rightarrow \mu) = \begin{cases} \frac{1}{N} & (\mu, \nu) \text{ διαφέρουν κατά ένα σπιν} \\ 0 & \text{αλλιώς} \end{cases}, \quad (13.18)$$

και ο αλγόριθμος είναι εργοδικός. Θα πρέπει, τότε, να ισχύει

$$\frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = e^{-\beta(E_\nu - E_\mu)}. \quad (13.19)$$

Ένας απλός τρόπος να ικανοποιήσουμε τη συνθήκη αυτή είναι να πάρουμε τη σχέση (12.61)

$$A(\mu \rightarrow \nu) = A_0 \cdot e^{-\frac{1}{2}\beta(E_\nu - E_\mu)}. \quad (13.20)$$

Για να μεγιστοποιήσουμε τους λόγους αποδοχής, αφού θα πρέπει να ισχύει  $A(\mu \rightarrow \nu) \leq 1$ ,  $|\Delta E| \leq 2z$ , παίρνουμε  $A_0 = e^{-\beta z}$ . Άρα,

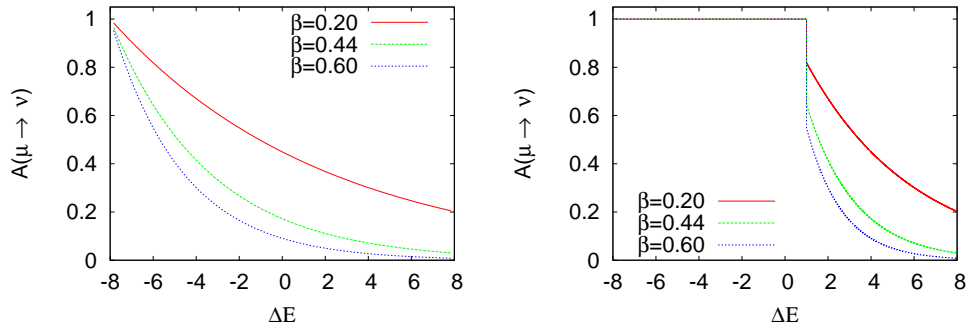
$$A(\mu \rightarrow \nu) = e^{-\frac{1}{2}\beta(E_\nu - E_\mu + 2z)}. \quad (13.21)$$

Στο σχήμα 13.4 φαίνεται η εξάρτηση του  $A(\mu \rightarrow \nu)$  από τη μεταβολή της ενέργειας για διάφορα  $\beta$ . Παρατηρούμε ότι η πιθανότητα αυτή γίνεται μικρή, ακόμα και για μηδενική μεταβολή της ενέργειας, με αποτέλεσμα η αποδοτικότητα της επιλογής αυτής να είναι μικρή.

Είναι πολύ αποδοτικότερο να χρησιμοποιήσουμε τον αλγόριθμο των Nicolas Metropolis *et. al.* 1953 [58] ο οποίος δίνεται από την (12.63)

$$A(\mu \rightarrow \nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & E_\nu - E_\mu > 0 \\ 1 & E_\nu - E_\mu \leq 0 \end{cases}. \quad (13.22)$$

Σύμφωνα με την παραπάνω σχέση, το σύστημα πάντα δέχεται την αλλαγή μιας κατάστασης, όταν αυτή ελαττώνει την ενέργεια, ενώ όταν η αλλαγή οδηγεί σε αύξηση της ενέργειας, αυτή γίνεται δεκτή με πιθανότητα μικρότερη της μονάδας. Όπως φαίνεται και από το σχήμα 13.4, το σύστημα δέχεται αλλαγές καταστάσεων πολύ συχνότερα από



Σχήμα 13.4: Λόγοι αποδοχής  $A(\mu \rightarrow \nu)$  για το δισδιάστατο πρότυπο Ising στο τετραγωνικό πλέγμα για τη σχέση (13.21) (αριστερά) και για τον αλγόριθμο Metropolis (δεξιά) σαν συνάρτηση της μεταβολής της ενέργειας  $\Delta E = E_\nu - E_\mu$ . Φαίνεται η υπερχρή του τελευταίου, αφού οι λόγοι αποδοχής είναι πολύ μεγαλύτεροι.

την προηγούμενη επιλογή μας. Ο αλγόριθμος αυτός έχει πολύ γενικότερη εφαρμογή από την περίπτωση που μελετάμε και έχει ευρεία χρήση σε ένα μεγάλο φάσμα προβλημάτων λόγω της γενικότητας, απλότητας και αποδοτικότητάς του. Να σημειώσουμε πως η επιλογή μας να αλλάζουμε την κατάσταση  $\mu \rightarrow \nu$ , αλλάζοντας την τιμή ενός μόνο σπιν δεν είναι περιορισμός της μεθόδου αυτής και θα μπορούσε να εφαρμοστεί για οποιαδήποτε αλλαγή καταστάσεων.

### 13.3 Σχεδιασμός Κώδικα

Για τον σχεδιασμό του κώδικα, το πιο σημαντικό βήμα είναι να ορίσουμε τη δομή των δεδομένων. Οι βαθμοί ελευθερίας είναι τα σπιν  $s_i = \pm 1$  ορισμένα στις  $N$  πλεγματικές θέσεις. Το πιο σημαντικό για την καλή απόδοση του προγράμματος είναι να ορίσουμε τις σχέσεις γειτονίας των πλεγματικών θέσεων στη μνήμη του υπολογιστή και τις συνοριακές συνθήκες. Οι τελευταίες είναι πολύ σημαντικές και μια λάθος επιλογή θα μπορούσε να κάνει την επίδραση του ορίου του πλέγματος σημαντική και έτσι, να έχουμε σημαντικά σφάλματα πεπερασμένου μεγέθους του πλέγματος (finite size effects). Οι πιο δημοφιλείς επιλογές είναι οι περιοδικές ή τοροειδείς συνοριακές συνθήκες, αλλά και οι ελικοειδείς συνοριακές συνθήκες. Ο σημαντικός παράγοντας στις επιλογές αυτές είναι ότι κάθε πλεγματική θέση έχει τον ίδιο αριθμό πλησιέστερων γειτόνων και έτσι, έχουμε την ίδια τοπική γεωμετρία. Για παράδειγμα, αυτό δε συμβαίνει αν επιλέξουμε το πλέγμα να έχει σύνορο πάνω στο οποίο τα σπιν να παίρνουν καθορισμένες ή ελεύθερες τιμές.

1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
6	7	8	9	10	6	7	8	9	10	6	7	8	9	10
11	12	13	14	15	11	12	13	14	15	11	12	13	14	15
16	17	18	19	20	16	17	18	19	20	16	17	18	19	20
21	22	23	24	25	21	22	23	24	25	21	22	23	24	25
1	2	3	4	5	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	1	2	3	4	5
6	7	8	9	10	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	6	7	8	9	10
11	12	13	14	15	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	11	12	13	14	15
16	17	18	19	20	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	16	17	18	19	20
21	22	23	24	25	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	21	22	23	24	25
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
6	7	8	9	10	6	7	8	9	10	6	7	8	9	10
11	12	13	14	15	11	12	13	14	15	11	12	13	14	15
16	17	18	19	20	16	17	18	19	20	16	17	18	19	20
21	22	23	24	25	21	22	23	24	25	21	22	23	24	25

Σχήμα 13.5: Τετραγωνικό πλέγμα με περιοδικές συνοριακές συνθήκες και  $L = 5$ . Η τοπολογία είναι τοροϊδής.

Για την τοποθέτηση των βαθμών ελευθερίας στη μνήμη, μια πρώτη επιλογή είναι να δώσουμε συντεταγμένες  $(i, j)$ ,  $i, j = 1, \dots, L$  σε κάθε πλεγματική θέση. Τα σπιν τοποθετούνται σε ένα array  $s(L, L)$  και αν επιλέξουμε μια θέση στο πλέγμα  $s(i, j)$ , οι τέσσερις πλησιέστεροι γείτονες θα είναι οι  $s(i \pm 1, j)$ ,  $s(i, j \pm 1)$ . Οι τοροειδείς συνοριακές συνθήκες εφαρμόζονται εύκολα προσθέτοντας/αφαιρώντας  $L$  στα  $i, j$ , όποτε αυτά γίνονται μικρότερα του 1/μεγαλύτερα του  $L$  αντίστοιχα. Αυτό φαίνεται στα σχήματα 13.5 και 13.34.

Για να υπολογιστεί όμως μια θέση στο πλέγμα πρέπει να υπολογιστεί η θέση της στη μνήμη  $(j-1)*L+i$  η οποία εμπεριέχει έναν (αθέατο) πολλαπλασιασμό. Μπορούμε να αποφύγουμε τον πολλαπλασιασμό αυτό εύκολα με εφαρμογή των ελικοειδών συνοριακών συνθηκών. Οι συντεταγμένες των πλεγματικών θέσεων καθορίζονται από έναν μόνο αριθμό  $i = 1, \dots, L^2 = N$  όπως φαίνεται στα σχήματα 13.6 και 13.35. Τα σπιν αποθηκεύονται στη μνήμη σε ένα μονοδιάστατο array  $s(N)$  και για να προσδιορίσουμε τους γείτονες μιας θέσης  $s(i)$  αρκεί να πάρουμε τα σπιν  $s(i \pm 1)$  και  $s(i \pm L)$ . Η απλότητα των ελικοειδών συνοριακών συνθηκών οφείλεται στο ότι για τους πλησιέστερους γείτονες των σπιν που βρίσκονται στο όριο του τετραγώνου, αρκεί να επαναφέρουμε τον δείκτη του array στα αποδεκτά όρια  $1 \leq i \leq N$  προσθέτοντας ή αφαι-



είναι ιδιαίτερα σημαντικό πρόβλημα, αλλά θα το μελετήσουμε λόγω της σημασίας του σε άλλα προβλήματα. Εδώ η αρχική κατάσταση μπορεί να επιλεγεί να είναι “παγωμένη” ( $\beta = \infty$  - όλα τα σπιν να έχουν την ίδια τιμή) ή “θερμή” ( $\beta = 0$  - τα σπιν να έχουν τιμή  $\pm 1$  με ίση πιθανότητα  $1/2$ ). Για μεγάλα πλέγματα όπου η εύρεση της κατάστασης θερμικής ισορροπίας μπορεί να απαιτεί αρκετά βήματα, επιλέγουμε τη σταδιακή μεταβολή της θερμοκρασίας από  $\beta = 0$  ή  $\beta = \infty$  στις επιθυμητές τιμές. Σε κάθε βήμα μεταβολής της θερμοκρασίας χρησιμοποιούμε τη διάταξη σπιν από την προηγούμενη προσομοίωση, έτσι ώστε να βρισκόμαστε γρήγορα στη νέα κατάσταση θερμικής ισορροπίας.

Ξεκινώντας από διαφορετικές αρχικές συνθήκες και συγκρίνοντας τα αποτελέσματά μας ελέγχουμε την ορθότητα τους, δηλ. αν το σύστημά μας έχει πετύχει τη θερμική ισορροπία και αν η μέθοδός μας πάσχει από προβλήματα εργοδικότητας.

Εξετάζουμε τώρα τα βήματα της διαδικασίας Markov που παράγεται από τον αλγόριθμο Metropolis. Έστω ότι το σύστημά μας είναι στην κατάσταση  $\mu = \{s_1^\mu, \dots, s_k^\mu, \dots, s_N^\mu\}$  και εξετάζουμε τη μετάβαση στη νέα κατάσταση  $\nu = \{s_1^\nu, \dots, s_k^\nu, \dots, s_N^\nu\}$  που προκύπτει από την αλλαγή της τιμής του σπιν  $s_k^\nu = -s_k^\mu$  (spin flip), ενώ τα υπόλοιπα σπιν παραμένουν τα ίδια  $s_j^\nu = s_j^\mu \ \forall j \neq k$ .

Η μεταβολή της ενέργειας θα είναι

$$E_\nu - E_\mu = \left( - \sum_{\langle ij \rangle} s_i^\nu s_j^\nu \right) - \left( - \sum_{\langle ij \rangle} s_i^\mu s_j^\mu \right) \quad (13.23)$$

$$= - \sum_{\langle ik \rangle} s_i^\mu (s_k^\nu - s_k^\mu) \quad (13.24)$$

$$= 2 \sum_{\langle ik \rangle} s_i^\mu s_k^\mu \quad (13.25)$$

$$= 2s_k^\mu \left( \sum_{\langle ik \rangle} s_i^\mu \right), \quad (13.26)$$

όπου στη δεύτερη σειρά απαλείψαμε τους κοινούς όρους των δύο αθροισμάτων και στην τρίτη χρησιμοποιήσαμε τη σχέση  $s_k^\nu - s_k^\mu = -2s_k^\mu$  που μπορείτε εύκολα να αποδείξετε εξετάζοντας τις δύο περιπτώσεις  $s_k^\mu = \pm 1$ . Το μεγάλο πλεονέκτημα της παραπάνω σχέσης είναι ότι είναι τοπική, εξαρτάται δηλαδή μόνο από τα γειτονικά σπιν και έτσι είναι γρήγορο να υπολογιστεί και η μεταβολή  $E_\nu - E_\mu$ , η οποία είναι μικρός αριθμός που δεν αυξάνει με το μέγεθος του πλέγματος.

Ο αλγόριθμος Metropolis εφαρμόζεται τώρα εύκολα. Υπολογίζοντας το άθροισμα στην παρένθεση της τελευταίας σειράς της (13.23) παίρ-

νουμε τη μεταβολή της ενέργειας  $E_\nu - E_\mu$ . Αν  $E_\nu - E_\mu \leq 0$ , δεχόμαστε ότι η κατάσταση  $\nu$  είναι η νέα κατάσταση του συστήματος. Αν  $E_\nu - E_\mu > 0$ , τότε  $A(\mu \rightarrow \nu) = e^{-\beta(E_\nu - E_\mu)} < 1$ . Επιλέγουμε έναν ομοιόμορφα κατανεμημένο τυχαίο αριθμό  $0 \leq x < 1$ . Η πιθανότητα να έχουμε  $x < A(\mu \rightarrow \nu)$  είναι  $A(\mu \rightarrow \nu)$ . Άρα αν  $x < A(\mu \rightarrow \nu)$ , δεχόμαστε τη μεταβολή. Αν  $A(\mu \rightarrow \nu) < x$ , η μεταβολή απορρίπτεται και παραμένουμε στην κατάσταση  $\mu$ .

Μια μικρή τεχνική παρατήρηση στον σχεδιασμό του προγράμματος είναι ότι οι δυνατές τιμές του αθροίσματος  $(\sum_{\langle ik \rangle} s_i^\mu) = -4, -2, 0, 2, 4$ . Οι τιμές που υπεισέρχονται στον υπολογισμό της  $A(\mu \rightarrow \nu)$  είναι μόνο αυτές που αυξάνουν την ενέργεια, δηλ. οι δύο τιμές 2, 4. Οι δύο αυτές τιμές για τα  $A(\mu \rightarrow \nu)$  μπορούν να αποθηκευτούν στη μνήμη του υπολογιστή και να αποφύγουμε την ακριβή σε χρόνο επανάληψη υπολογισμού των εκθετικών  $e^{-\beta(E_\nu - E_\mu)}$ .

Τέλος, το πρόγραμμά μας θα πρέπει να υπολογίζει τις φυσικές ποσότητες που θα μετρήσουμε κατά τη μελέτη του συστήματός μας. Αυτές είναι η ενέργεια (13.13)

$$E = - \sum_{\langle ij \rangle} s_i s_j, \quad (13.27)$$

και η μαγνήτιση

$$M = \left| \sum_i s_i \right|. \quad (13.28)$$

Η απόλυτη τιμή στην τελευταία σχέση μπαίνει λόγω της συμμετρίας ανάκλασης των σπιν του συστήματος (ομάδα  $Z_2$ ). Επειδή είναι ισοπίθανο να έχουμε δύο διατάξεις με όλα τα σπιν τους να έχουν αντίθετες τις τιμές τους σε κάθε πλεγματική θέση, η μέση τιμή  $\langle \sum_i s_i \rangle$  είναι πάντα 0. Για αρκετά μεγάλο αριθμό μετρήσεων, όσες φορές το σύστημά μας θα είναι σε μία διάταξη με  $\sum_i s_i = M_1$ , θα είναι και σε μία αντίστοιχη διάταξη με  $\sum_i s_i = -M_1$ . Στην πράξη, όπως θα δούμε, η μετάβαση από την κατάσταση με μαγνήτιση  $M_1$  στην κατάσταση με μαγνήτιση  $-M_1$  μπορεί να χρειάζεται μεγάλο αριθμό βημάτων Μόντε Κάρλο, ιδιαίτερα όταν  $\beta \gg \beta_c$ , και αυτό να μην παρατηρείται στην προσομοίωσή μας. Στο όριο  $N \rightarrow \infty$  (θερμοδυναμικό όριο) το σύστημα επιλέγει τη μία από τις δύο καταστάσεις (αυθόρμητο σπάσιμο συμμετρίας).

Οι μετρήσεις της ενέργειας και της μαγνήτισης μπορούν να γίνονται με δύο τρόπους. Στον πρώτο ενημερώνουμε την τιμή της ενέργειας και της μαγνήτισης σε κάθε βήμα Μόντε Κάρλο. Αυτό γίνεται εύκολα, γιατί η μεταβολή των τιμών των αθροισμάτων στις (13.27) και (13.28) εξαρτάται μόνο από την τιμή του σπιν  $s_k^\mu$  που αλλάζει και των τιμών του

σπιν των πλησιέστερων γειτόνων. Τη μεταβολή της ενέργειας την υπολογίζουμε έτσι κι αλλιώς από τη σχέση (13.23), ενώ η μεταβολή του αθροίσματος στην (13.28) δίνεται απλά από την

$$\sum_i s_i^\nu - \sum_i s_i^\mu = s_k^\nu - s_k^\mu = -2s_k^\mu \quad (13.29)$$

Ο δεύτερος τρόπος είναι να υπολογίζουμε τα αθροίσματα (13.27) και (13.28) κάθε φορά που θέλουμε να πάρουμε μια μέτρηση. Ποια μέθοδος είναι πιο συμφέρουσα εξαρτάται από το πόσο συχνά παίρνουμε μια στατιστικά ανεξάρτητη μέτρηση<sup>7</sup> και από τον λόγο αποδοχής των βημάτων Μόντε Κάρλο. Λ.χ. για τον υπολογισμό της μαγνήτισης, αν ο μέσος λόγος αποδοχής είναι  $\bar{A}$ , τότε με τον πρώτο τρόπο έχω  $\bar{A}N$  προσθέσεις ανά Μόντε Κάρλο βήματα (1 sweep), ενώ με τον δεύτερο έχω  $N$  προσθέσεις ανά μέτρηση.

Είναι συνηθισμένο να κανονικοποιούμε την ενέργεια ανά δεσμό

$$\langle e \rangle = \frac{1}{N_l} \langle E \rangle = \frac{1}{2N} \langle E \rangle, \quad (13.30)$$

η οποία παίρνει τιμή από -1 (θεμελιώδης κατάσταση, όλοι οι  $2N$  δεσμοί έχουν τιμή -1) μέχρι +1, ενώ τη μαγνήτιση ανά πλεγματική θέση

$$\langle m \rangle = \frac{1}{N} \langle M \rangle, \quad (13.31)$$

η οποία παίρνει τιμή από 0 (πλήρης αταξία -  $\beta = 0$ ) έως 1 (θεμελιώδης κατάσταση -  $\beta = \infty$ ). Για τον λόγο αυτό, η μαγνήτιση είναι μια παράμετρος τάξης (order parameter), δηλ. μια φυσική ποσότητα της οποίας η μέση τιμή μας υποδεικνύει τότε το σύστημα βρίσκεται σε φάση τάξης (ordered phase) ή σε φάση αταξίας (disordered phase).

Από τις ποσότητες αυτές υπολογίζουμε τις διακυμάνσεις της ενέργειας από την ειδική θερμότητα

$$c = \beta^2 N \langle (e - \langle e \rangle)^2 \rangle = \beta^2 N (\langle e^2 \rangle - \langle e \rangle^2), \quad (13.32)$$

και της μαγνήτισης από τη μαγνητική επιδεκτικότητα

$$\chi = \beta N \langle (m - \langle m \rangle)^2 \rangle = \beta N (\langle m^2 \rangle - \langle m \rangle^2). \quad (13.33)$$

Για να εκτιμήσουμε πόσα δεδομένα πρέπει να συλλέξουμε για μια καλή μέτρηση των παραπάνω ποσοτήτων, ανατρέχουμε στο αποτέλεσμα της στατιστικής που μας λέει ότι αν έχουμε  $n$  ανεξάρτητες μετρήσεις, τότε το στατιστικό σφάλμα ελαττώνεται σαν  $\sim 1/\sqrt{n}$ . Το πρόβλημα είναι τότε έχουμε ανεξάρτητες μετρήσεις, το οποίο θα το αντιμετωπίσουμε παρακάτω.

---

<sup>7</sup>Εξαρτάται, όπως θα δούμε παρακάτω, από τον χρόνο αυτοσυσχετισμού.

### 13.3.1 Ο Κώδικας

Στην παράγραφο αυτή θα παρουσιάσουμε και αναλύσουμε τον κώδικα<sup>8</sup> που θα χρησιμοποιήσουμε στην προσομοίωση του πρότυπου Ising. Ο κώδικας αυτός βρίσκεται στο συνοδευτικό λογισμικό, στον κατάλογο Ising\_Introduction του κεφαλαίου αυτού.

Θα ακολουθήσουμε τη φιλοσοφία του modular programming όπου θα τοποθετήσουμε σε διαφορετικά αρχεία ξεχωριστές ενότητες του κώδικα. Αυτό κάνει ευκολότερη την ανάπτυξη, έλεγχο και συντήρηση του κώδικα από έναν ή και περισσότερους προγραμματιστές. Αρχικά, σε ένα αρχείο–επικεφαλίδα (header file) τοποθετούμε τους ορισμούς που είναι κοινοί για ένα ή περισσότερα αρχεία. Το πλεονέκτημα μιας τέτοιας επιλογής είναι ότι έχουμε συγκεντρωμένες σε ένα αρχείο, όλες τις βασικές παραμέτρους και κοινές μεταβλητές (common blocks), ώστε κάθε αλλαγή τους να γίνεται αυτόματα και χωρίς λάθη παντού στον κώδικα. Το αρχείο αυτό, εδώ το ονομάζουμε include.inc, θα συμπεριλαμβάνεται σε κάθε αρχείο που θα το χρησιμοποιεί με μια εντολή του include:

```
!===== include.inc =====
implicit none
integer ,parameter      :: L = 12
integer ,parameter      :: N = L*L
integer ,parameter      :: XNN = 1, YNN = L
integer ,dimension(N)   :: s
real(8) ,dimension(0:4) :: prob
real(8)                  :: beta
common /lattice/         s
common /parameters/     beta,prob
!function definitions:
real(8)                  :: drandom
integer                  :: seed
common /randoms/         seed
```

Το μέγεθος του πλέγματος εισάγεται ως σταθερή παράμετρος, ενώ οι μεταβλητές σπιν και παράμετροι της προσομοίωσης τοποθετούνται σε common blocks. Το array s(N) είναι τα σπιν στις πλεγματικές θέσεις και παίρνουν τιμές  $\pm 1$ . Η μεταβλητή beta είναι η θερμοκρασία  $\beta$  του συστήματος και στην prob(0:4) τοποθετούμε τις δυνατές τιμές των πιθανοτήτων  $A(\mu \rightarrow \nu)$  όπως εξηγήσαμε στη σελίδα 534. Η drandom() είναι αυτή που θα χρησιμοποιήσουμε για γεννήτρια ψευδοτυχαίων αριθμών και ο σπόρος seed είναι ορατός από όλο το πρόγραμμα, έτσι ώστε να

<sup>8</sup>Οι βασικές ιδέες του προγράμματος έχουν παρθεί από το βιβλίο των Newmann και Barkema [4].



μπορούμε να τον θέσουμε από διαφορετικές συναρτήσεις, όποτε το επιθυμούμε. Τέλος, οι παράμετροι XNN και YNN είναι αυτές που μας οδηγούν στους πλησιέστερους γείτονες μιας πλεγματικής θέσης σύμφωνα με όσα είπαμε στην αρχή της ενότητας 13.3. Για παράδειγμα  $i + \text{XNN}$  είναι ο πλησιέστερος γείτονας της (εσωτερικής) θέσης  $i$  κατά την  $+x$  διεύθυνση και  $i - \text{YNN}$  είναι ο πλησιέστερος γείτονας κατά την  $-y$  διεύθυνση.

Το κυρίως πρόγραμμα βρίσκεται στο αρχείο `main.f90` και οδηγεί την προσομοίωση:

```
!===== main.f90 =====
program Ising2D
  include 'include.inc'
  integer :: start      ! start= 0 (cold)/1 (hot)
  integer :: isweep, nsweep

  nsweep = 1000
  beta=0.21D0; seed=9873; start=1;
  call init(start)
  do isweep = 1, nsweep
    call met
    call measure
  end do
end program Ising2D
```

Εδώ θέτουμε τις παραμέτρους της προσομοίωσης. Η αρχική κατάσταση καθορίζεται από την τιμή της `start` και είναι ψυχρή για `start=0` και θερμή για `start=1`. Η θερμοκρασία τίθεται από την τιμή της `beta` και ο αριθμός των sweeps του πλέγματος από την `nsweep`. Από το μικρό αυτό αρχείο βλέπουμε τη βασική δομή της προσομοίωσης: Στην αρχή τίθεται η αρχική κατάσταση του συστήματος (`init`), ενώ μετά για κάθε sweep κάνουμε ένα Metropolis sweep (`met`) και μία μέτρηση (`measure`).

Σε ένα επίπεδο παρακάτω, βλέπουμε το πρόγραμμα της `init` στην οποία περνάμε τη μεταβλητή `start`, προκειμένου να θέσει το πλέγμα στην αρχική του κατάσταση:

```
!===== init.f90 =====
! file init.f90
! init(start): start = 0: cold start
!              start = 1: hot  start
!=====
subroutine init(start)
  include 'include.inc'
  integer :: start
  integer :: i
```

```

!-----
!initialize probabilities for E_nu > E_mu
prob=0.0D0
do i=2,4,2 !i = dE/2 = (E_nu-E_mu)/2=2,4
  prob(i) = exp(-2.0D0*beta*i)
enddo
!initial configuration:
select case(start)
case(0)!cold:
  s = 1 !all s(i) = 1
case(1)!hot:
  do i=1,N
    if(drandom() .lt. 0.5D0)then
      s(i) = 1
    else
      s(i) = -1
    endif
  enddo
case default
  print *, 'init: start= ',start, ' not valid. Exiting...'
  stop
end select

end subroutine init

```

Η δομή της συνάρτησης είναι πολύ απλή. Αρχικοποιεί το array  $\text{prob}(0:4)$  με τις τιμές των πιθανοτήτων  $A(\mu \rightarrow \nu) = e^{-\beta(E_\nu - E_\mu)} = e^{-2\beta s_k^\mu (\sum_{\langle ik \rangle} s_i^\mu)}$ . Οι πιθανότητες αυτές μας ενδιαφέρουν, όταν  $s_k^\mu (\sum_{\langle ik \rangle} s_i^\mu) > 0$  και οι δυνατές τιμές που μπορεί τότε να πάρει είναι 2 και 4. Αυτές είναι και οι τιμές που αποθηκεύουμε και θυμόμαστε ότι ο δείκτης του array είναι η έκφραση  $s_k^\mu (\sum_{\langle ik \rangle} s_i^\mu)$ , όταν αυτή είναι θετική.

Στη συνέχεια, καθορίζουμε την αρχική διάταξη των σπιν ανάλογα με την τιμή που παίρνει ο ακέραιος  $\text{start}$ . Το `select case` block μας επιτρέπει να προσθέσουμε στο μέλλον και άλλες δυνατότητες, αν τυχόν αναβαθμίσουμε τον κώδικα. Αν  $\text{start}=0$ , οι τιμές όλων των σπιν τίθενται ίσες με 1, ενώ αν  $\text{start}=1$  θέτουμε τις τιμές των σπιν με ίση πιθανότητα  $\pm 1$ . Η πιθανότητα  $\text{drandom}() < 0.5$  είναι  $1/2$ , οπότε στην περίπτωση αυτή  $s(i)=1$ , αλλιώς (πιθανότητα  $1 - 1/2 = 1/2$ )  $s(i)=-1$ .

Η καρδιά του προγράμματος είναι η υπορουτίνα `met()`. Αυτή εκτελεί  $N$  Metropolis βήματα. Επισκέπτεται  $N$  φορές μία τυχαία πλεγματοτική θέση και θέτει το ερώτημα αν θα μεταβάλλει την τιμή του σπιν (spin flip). Αυτό γίνεται σύμφωνα με τον αλγόριθμο Metropolis υπολογίζοντας τη μεταβολή στην ενέργεια πριν και μετά την αλλαγή του σπιν σύμφωνα με τη σχέση (13.23):

```

!===== met.f90 =====
subroutine met()
  include 'include.inc'
  integer :: i,k
  integer :: nn,snn,dE

  do k=1,N
!pick a random site:
    i = INT(N*drandom())+1
!snn=sum of neighboring spins:
    snn = 0
    nn=i+XNN; if (nn.gt.N) nn=nn-N; snn = snn + s(nn)
    nn=i-XNN; if (nn.lt.1) nn=nn+N; snn = snn + s(nn)
    nn=i+YNN; if (nn.gt.N) nn=nn-N; snn = snn + s(nn)
    nn=i-YNN; if (nn.lt.1) nn=nn+N; snn = snn + s(nn)
!dE=change in energy/2:
    dE=snn*s(i)
!flip:
    if (dE.le.0) then
      s(i) = -s(i) !accept
    else if (drandom() < prob(dE)) then
      s(i) = -s(i) !accept
    endif
  enddo !do k=1,N: end sweep
end subroutine met

```

Η γραμμή

```
i = INT(N*drandom())+1
```

επιλέγει ισοπίθανα μια πλεγματική θέση  $i=1, \dots, N$ . Εδώ, είναι σημαντικό να μην παρουσιαστεί ποτέ η τιμή  $i=N+1$  που θα μπορούσε να γίνει, αν  $\text{drandom()}=1.0$ . Την τιμή αυτή την έχουμε αποκλείσει, όπως συζητήσαμε στην Παράγραφο 11.1.

Στη συνέχεια, υπολογίζουμε το άθροισμα  $\left(\sum_{\langle ik \rangle} s_i^\mu\right)$  της (13.23). Για τον λόγο αυτό πρέπει να βρούμε τους πλησιέστερους γείτονες της θέσης  $i$  και αυτό γίνεται με τις γραμμές

```

snn = 0
nn=i+XNN; if (nn.gt.N) nn=nn-N; snn = snn + s(nn)
nn=i-XNN; if (nn.lt.1) nn=nn+N; snn = snn + s(nn)
nn=i+YNN; if (nn.gt.N) nn=nn-N; snn = snn + s(nn)
nn=i-YNN; if (nn.lt.1) nn=nn+N; snn = snn + s(nn)

```

Στη συνέχεια, θέτουμε στη μεταβλητή delta το γινόμενο (13.23)  $s_k^\mu \left(\sum_{\langle ik \rangle} s_i^\mu\right)$ .

Αν αυτό είναι αρνητικό, τότε η μεταβολή της ενέργειας είναι αρνητική και η αλλαγή του σπιν γίνεται δεκτή. Αν όχι, τότε εφαρμόζουμε το κριτήριο (13.22) χρησιμοποιώντας την τιμή του array `prob(delta)` σύμφωνα με τον τρόπο που το κατασκευάσαμε στην `init`. Η πιθανότητα του ομοιόμορφα κατανεμημένου στο διάστημα  $(0,1)$  αριθμού `drandom() < prob(delta)` είναι `prob(delta)` και στην περίπτωση αυτή η αλλαγή γίνεται δεκτή. Σε κάθε άλλη περίπτωση, η αλλαγή δε γίνεται δεκτή και το σπιν `s(i)` διατηρεί την παλιά του τιμή.

Μετά από κάθε Metropolis sweep έχουμε επιλέξει να κάνουμε μια μέτρηση της ενέργειας και της μαγνήτισης. Εδώ επιλέγουμε μινιμαλιστική προσέγγιση του κεντρικού κώδικα και υπολογίζουμε μόνο τα αναγκαία. Αφήνουμε στη διαδικασία της ανάλυσης τον υπολογισμό των μέσων τιμών, των στατιστικών σφαλμάτων, της ειδικής θερμότητας και της μαγνητικής επιδεκτικότητας. Δίνουμε στην ανάλυση τη μέγιστη δυνατή πληροφορία που θέλουμε να παρέχουμε και αφήνουμε στη φάση εκείνη να αποφασιστεί αν και πώς θα υπολογιστεί μια φυσική ποσότητα. Αυτό, αν δεν έχουμε πρόβλημα αποθηκευτικού χώρου, είναι προτιμητέο, αν η παραγωγή των δεδομένων είναι “ακριβή” και η ανάλυση “φτηνή”. Ένας άλλος λόγος που παρέχουμε στην ανάλυση την εκάστοτε τιμή της ενέργειας και μαγνήτισης είναι ώστε να γίνει δυνατή η λεπτομερής παρακολούθηση της προσομοίωσης και να εντοπιστούν προβλήματα. Μπορούμε να δούμε αν το σύστημα έφτασε σε θερμική ισορροπία, αν παρουσιάζονται μεγάλοι αυτοσυσχετισμοί ή αν υπάρχουν λάθη στο πρόγραμμα.

Η `measure` υπολογίζει κάθε φορά που καλείται τη συνολική ενέργεια και μαγνήτιση (χωρίς την απόλυτη τιμή) καλώντας τις ξεχωριστές συναρτήσεις `E()` και `M()`. Αυτές απλά εφαρμόζουν τις σχέσεις (13.27) και (13.28) (χωρίς την απόλυτη τιμή).

```
!===== measure.f90 =====
subroutine measure()
  include 'include.inc'
  integer :: E,M
  print *, E(),M()
end subroutine measure
!=====
integer function E()
  include 'include.inc'
  integer en,sum,i,nn
  en = 0
  do i=1,N
!Sum of neighboring spins: only forward nn necessary in the sum
    sum = 0
    nn=i+XNN; if (nn.gt.N) nn=nn-N; sum = sum + s(nn)
```

```

nn=i+YNN; if (nn.gt.N) nn=nn-N; sum = sum + s(nn)
en=en+sum*s(i)
enddo
e = -en
end function E
!=====
integer function M()
include 'include.inc'
M=SUM(s)
end function M

```

Η μεταγλώττιση του κώδικα γίνεται με την εντολή

```

> gfortran main.f90 met.f90 init.f90 measure.f90 drandom.f90 \
  -o is

```

η οποία παράγει το εκτελέσιμο αρχείο `is` το οποίο μπορούμε τώρα να τρέξουμε:

```

> ./is > out.dat
> less out.dat
-52 10
-48 40
-64 44
-92 64
.....

```

Η έξοδος του προγράμματος είναι σε δύο στήλες, στην πρώτη τυπώνεται η συνολική ενέργεια και στη δεύτερη η μαγνήτιση (χωρίς απόλυτη τιμή). Για να φτιάξουμε τις λεγόμενες “χρονοσειρές” των μετρήσεών μας μπορούμε να δώσουμε τις εντολές

```

gnuplot> plot "out.dat" using 1 with lines
gnuplot> plot "out.dat" using 2 with lines
gnuplot> plot "out.dat" using (($2>0)?$2:-$2) with lines

```

Στην τελευταία γραμμή φτιάχνουμε το γράφημα της  $M$  παίρνοντας την απόλυτη τιμή του αθροίσματος  $\sum_i s_i$ . Ζητάμε από το `gnuplot` να μας κάνει τη γραφική παράσταση παίρνοντας τη δεύτερη στήλη  $\$2$ , αν αυτή είναι θετική ( $\$2>0$ )?, αλλιώς το αντίθετό της  $-\$2$ .

### 13.3.2 Βελτίωση του Interface

Στην ενότητα αυτή θα κάνουμε βελτιώσεις στον κώδικα, κυρίως στο επίπεδο αλληλεπίδρασης με το χρήστη (user interface). Τον κώδικα του προγράμματος που συζητείται παρακάτω θα τον βρείτε στον κατάλογο Ising\_Metropolis στο συνοδευτικό λογισμικό αυτού του κεφαλαίου. Το πρόγραμμα στη μορφή που του δώσαμε στην προηγούμενη ενότητα έχει προγραμματισμένες (hard coded) τις διαφορές παραμέτρους της προσομοίωσης και κάθε φορά που θέλουμε να τις αλλάξουμε, πρέπει να τις κωδικοποιήσουμε στο αντίστοιχο αρχείο και να επαναμεταγλωττίσουμε. Επίσης αλλάζουμε τον κώδικα, έτσι ώστε να αποθηκεύεται η τελευταία διάταξη στην οποία βρίσκεται το σύστημα, καθώς και να εισάγεται διάταξη του συστήματος που έχει αποθηκευτεί στο δίσκο.

Ανάμεσα στις παραμέτρους που θα μπορεί ο χρήστης να ορίζει τη στιγμή που τρέχει το πρόγραμμα είναι και το μέγεθος του πλέγματος  $L$ . Αυτό καθορίζει την απαιτούμενη μνήμη για το array  $s(N)$  την οποία θα πρέπει να ζητήσουμε δυναμικά από το σύστημα με τη συνάρτηση ALLOCATE. Το πρόβλημα είναι πως το  $s(N)$  έχει δεδομένα που πρέπει να είναι προσβάσιμα από όλες τις διαδικασίες του προγράμματος και arrays που είναι allocatable και δεν μπορούν να μπου σε common blocks. Ένας άλλος τρόπος να μοιράζονται οι διαδικασίες δεδομένα είναι με τη χρήση modules. Αυτός είναι και ο προτιμότερος σε νέα προγράμματα Fortran όπου η χρήση των common blocks συνιστάται να αποφεύγεται. Τα δεδομένα (εδώ απλά παράμετροι και μεταβλητές) τοποθετούνται ανάμεσα στις δηλώσεις

```
module global_data
  implicit none
  SAVE
  ....
end module global_data
```

Στις ... τοποθετούμε δηλώσεις μεταβλητών και χρησιμοποιούμε την εντολή SAVE. Το όνομα του module εδώ είναι global\_data και όποια διαδικασία θέλει να μοιραστεί τα δεδομένα του module πρέπει να ξεκινήσει με την εντολή use global\_data:

```
subroutine share_global_data
  use global_data
  implicit none
  ....
end subroutine share_global_data
```

Σε ένα αρχείο `global_data.f90` ορίζουμε τις μεταβλητές που θα είναι προσβάσιμες από όλες τις διαδικασίες:

```
module global_data
  implicit none
  SAVE
  integer          :: L
  integer          :: N
  integer          :: XNN, YNN
  integer, allocatable :: s(:)
  real(8), dimension(0:4) :: prob
  real(8)          :: beta
  integer          :: nsweep, start
  integer          :: seed, ranlux_level
  real(8)          :: acceptance
  character(1024)  :: prog
end module global_data
```

Το array `s(:)` είναι τώρα `allocatable` και θα το ορίσουμε κατά την αρχικοποίηση του προγράμματος στην `init`, ενώ οι μεταβλητές `L`, `N`, `XNN` και `YNN` δεν ορίζονται πια ως παράμετροι και η τιμή τους θα καθοριστεί, επίσης, στην `init`. Οι καινούργιες μεταβλητές είναι οι `acceptance` που δίνει το ποσοστό των spin flips που γίνονται αποδεκτά κατά την προσομοίωση, `ranlux_level` που καθορίζει το luxury level της RANLUX και `prog` είναι το όνομα του προγράμματος που τρέχει την προσομοίωση.

Στο κυρίως πρόγραμμα δεν έχουμε μεγάλες αλλαγές:

```
!===== main.f90 =====
program Ising2D
  use global_data
  implicit none
  integer :: isweep

  call init
  do isweep = 1, nsweep
    call met
    call measure
  end do
  call endsim
end program Ising2D
```

Παρατηρήστε τη γραμμή `use global_data` που δίνει πρόσβαση στα δεδομένα του module `global_data`. Με αυτή τη γραμμή θα ξεκινάνε όλες οι διαδικασίες του προγράμματος της προσομοίωσης. Επίσης, η υπορουτίνα `endsim` κάνει τις εργασίες που απαιτούνται πριν κλείσει το

πρόγραμμα. Ιδιαίτερα εδώ θα σταθούμε στην εγγραφή της τελικής διάταξης του συστήματος στον δίσκο για μετέπειτα χρήση.

Στην `init` έχουμε κάνει αρκετές αλλαγές, έτσι ώστε να ενσωματώσουμε τις νέες δυνατότητες του προγράμματος:

```
!===== init.f90 =====
! start = 0: cold start
! start = 1: hot start
! start = 2: use old configuration
!=====
subroutine init
  use global_data
  implicit none
  integer :: i,chk
  real(8) :: obeta=-1.0D0,r
  integer :: OL=-1
  character(1024) :: buf
  integer ,parameter :: f_in=17 !file unit
  integer :: seeds(25)
!_____
!Define parameters from options:
L=-1;beta=-1.0D0;nsweep=-1;start=-1;seed=-1
ranlux_level=3
call get_the_options
if(start.EQ.0 .OR. start.EQ.1)then
  if(L < 0 )call locerr('L has not been set.')
  if(seed < 0 )call locerr('seed has not been set.')
  if(beta < 0.0D0)call locerr('beta has not been set.')
!Derived parameters:
N=L*L;XNN=1;YNN=L
!Allocate memory for the spins:
ALLOCATE(s(N),STAT=chk)
if(chk > 0)call locerr('allocation failure for s(N)')
endif !if(start.EQ.0 .OR. start.EQ.1)
if(start < 0)call locerr('start has not been set.')
if(nsweep < 0)call locerr('nsweep has not been set.')
!_____
!initialize probabilities for E_nu > E_mu
prob=0.0D0
do i=2,4,2 !i = dE/2 = (E_nu-E_mu)/2=2,4
  prob(i) = exp(-2.0D0*beta*i)
enddo
acceptance = 0.0D0
!_____
!initial configuration: cold(0),hot(1),old(2)
!_____
select case(start)
```



```

!
case(0)!cold:
  call simmessage(6)
  call RLUXGO(ranlux_level,seed,0,0)
  s = 1 !all s(i) = 1
!
case(1)!hot:
  call simmessage(6)
  call RLUXGO(ranlux_level,seed,0,0)
  do i=1,N
    call ranlux(r,1)
    if(r .lt. 0.5D0)then
      s(i) = 1
    else
      s(i) = -1
    endif
  enddo
!
case (2)!old:
  if(beta < 0.0D0)call locerr('beta has not been set.')
  open(f_in,file='conf',status='OLD',ERR=101)
  read(f_in,*)buf !read in a comment line
  read(f_in,'(A4,I5,A4,I5,A6,G28.17,A6,25I16)')&
    buf,OL,buf,OL,buf,obeta,buf,seeds
  if(L < 0 ) L = OL !if L has not been set, read from file
  if(L /= OL) & ! /= the same as .NE. (not equal)
    call locerr('L different from the one read from conf.')
  N=L*L;XNN=1;YNN=L
!Allocate memory for the spins:
  ALLOCATE(s(N),STAT=chk);
  if(chk > 0)call locerr('allocation failure for s(N)')
  call simmessage(6)
  print '(A)', '# Reading configuration from file conf'
  do i=1,N
    read(f_in,*,END=102) s(i)
    if(s(i) /= 1 .AND. s(i) /= -1)&
      call locerr('wrong value of spin')
  enddo
  close(f_in)
  if(seed < 0) then !initialize from seeds read from file:
    call RLUXIN(seeds)
  else !option seed sets new seed:
    call RLUXGO(ranlux_level,seed,0,0)
  endif
!
case default
  print *, 'init: start= ',start, ' not valid. Exiting...'
  stop 1
end select

```

```
!
  return
!here we put error messages:
101 call locerr('Configuration file conf not found.')
102 call locerr('File conf ended before reading all spins.')
end subroutine init
```

Ξεκινάμε με τον ορισμό των παραμέτρων του προγράμματος δίνοντάς τους προκαθορισμένες τιμές, οι οποίες, όταν είναι αρνητικές, αυτό είναι ένδειξη ότι δεν έχουν οριστεί από το χρήστη. Μετά, καλούμε την υπορουτίνα<sup>9</sup> `get_the_options` η οποία διαβάζει τις τιμές που θέτει ο χρήστης από τη γραμμή εντολών:

```
L=-1;beta=-1.0D0;nsweep=-1;start=-1;seed=-1
call get_the_options
```

Όταν επιστρέφει, θεωρείται ότι οι τιμές έχουν τεθεί και πρέπει να γίνει έλεγχος αν αυτό έχει γίνει σωστά. Για παράδειγμα, αν ο χρήστης ξεκινάει μια καινούργια προσομοίωση και έχει ξεχάσει να ορίσει το `L`, τότε καλείται η συνάρτηση `locerr` η οποία σταματάει το πρόγραμμα με το μήνυμα σφάλματος που περιέχεται στην παρένθεση:

```
if(L < 0 )call locerr('L has not been set.')
```

Όταν γίνει γνωστή η τιμή του `N`, τότε το πρόγραμμα μπορεί να ζητήσει την εκχώρηση μνήμης για το `s(N)`:

```
N=L*L
ALLOCATE(s(N),STAT=chk)
if(chk > 0)call locerr('allocation failure for s(N)')
```

Η μεταβλητή `chk` που ορίζεται από το όρισμα `STATUS` είναι 0, όταν η εκχώρηση είναι επιτυχής. Αν δεν είναι, τότε το πρόγραμμα σταματάει με κλήση της `locerr`.

Στη συνέχεια, έχουμε το `SELECT CASE(start)` που αρχικοποιεί το πρόγραμμα ανάλογα με την τιμή της `start`. Αν `start=0`, ξεκινάμε όπως και πριν με όλα τα σπιν να έχουν την τιμή 1. Το μόνο καινούργιο είναι η κλήση της υπορουτίνας `simmmessage(f_unit)` που τυπώνει πληροφορίες για την προσομοίωση στο αρχείο που έχουμε συνδέσει με το `unit f_unit`. Επίσης, ξεκινάμε τη γεννήτρια τυχαίων αριθμών `RANLUX` με την `RLUXGO` όπως είχαμε πει στην Παράγραφο 11.2, σελ. 485. Εδώ γίνεται

<sup>9</sup>Θα την προγραμματίσουμε εμείς αργότερα στο αρχείο `options.f90`.

και η χρήση της global μεταβλητής `ranlux_level` της οποίας την τιμή προκαθορίσαμε παραπάνω να είναι ίση με 3 και την οποία μπορεί, αν θέλει, να αλλάξει ο χρήστης από την `get_the_options`. Αν `start=0`, ξεκινάμε από θερμή διάταξη των σπιν, όπως κάναμε και στην προηγούμενη παράγραφο.

Αν `start=2`, τότε επιχειρούμε να διαβάσουμε διάταξη των σπιν η οποία έχει προηγουμένως αποθηκευτεί στο δίσκο. Αυτή υποθέτουμε πως είναι σε ένα αρχείο με όνομα `conf` το οποίο έχει αυστηρά προκαθορισμένο `format` (το οποίο ορίζεται στην υπορουτίνα `endsim`). Αν το αρχείο δεν υπάρχει, το όρισμα `ERR=101` μεταφέρει τη ροή του προγράμματος στο τέλος, στην εντολή με ετικέτα 101, όπου τυπώνεται μήνυμα σφάλματος από την `locerr`. Για να καταλάβουμε τις επόμενες εντολές `read` πρέπει να ξέρουμε το `format` του `conf` το οποίο είναι (περίπου) της μορφής:

```
# Configuration of 2d Ising model on square lattice....
Lx= 12 Ly= 12 beta= 0.21 seed= 3718479 5267541 12092770 ....
-1
 1
 1
 1
-1
.....
```

Τα σχόλια “πετιούνται” στην character μεταβλητή `buf`, ενώ διαβάζονται δεδομένα που δίνουν το μέγεθος του πλέγματος, τη θερμοκρασία και την κατάσταση της RANLUX. Αυτά δεν αποθηκεύονται απευθείας στις μεταβλητές `L`, `beta`, αλλά σε προσωρινές μεταβλητές `OL`, `obeta`, έτσι ώστε να συγκριθούν με τις τιμές που μπορεί να έχει δώσει αρχικά ο χρήστης.

Αν ο χρήστης έχει δώσει δικό του `seed`, τότε η RANLUX αρχικοποιείται με το `seed`. Αν όχι, τότε η κατάσταση της RANLUX καθορίζεται από τα `seeds` της προηγούμενης προσομοίωσης. Και οι δύο επιλογές έχουν σοβαρούς λόγους: Μπορεί ο χρήστης να θέλει να συνεχίσει την προσομοίωση ακριβώς από το σημείο που την είχε αφήσει στο `conf`. Τότε θα θέλει η RANLUX να συνεχίσει να παράγει την ίδια ακολουθία τυχαίων αριθμών που είχε σταματήσει στο `conf`. Μπορεί όμως να θέλει να παράγει πολλά ανεξάρτητα αποτελέσματα από την ίδια διάταξη των σπιν. Τότε θα ξεκινήσει κάθε φορά τη RANLUX με διαφορετικό `seed` και θα κάνει έτσι ανεξάρτητες προσομοιώσεις<sup>10</sup>. Αυτό επιτυγχάνεται με τις γραμμές

<sup>10</sup>Οι προσομοιώσεις γίνονται ανεξάρτητες μετά από χρόνο τουλάχιστον  $2\tau$ , όπως θα

```

if(seed < 0) then !initialize from seeds read from file:
  call RLUXIN(seeds)
else             !option seed sets new seed:
  call RLUXGO(ranlux_level,seed,0,0)
endif

```

Τέλος, να σημειώσουμε πως διαβάζουμε τις τιμές των σπιν με προσοχή, έτσι ώστε να έχουν μόνο τις επιτρεπτές τιμές  $\pm 1$  και να είναι ίσα σε αριθμό με αυτόν που καθορίζεται από το μέγεθος του πλέγματος<sup>11</sup>. Το τελευταίο επιτυγχάνεται με το όρισμα END=102 της εντολής READ, όπου 102 είναι η ετικέτα της εντολής όπου θα μεταφερθεί ο έλεγχος του προγράμματος από την εντολή READ, όταν επιχειρήσει να διαβάσει μετά το τέλος του αρχείου. Αυτό θα συμβεί, αν π.χ. επιχειρήσουμε να διαβάσουμε ένα κατεστραμμένο αρχείο.

Ας δούμε τώρα την αποθήκευση της τελευταίας διάταξης των σπιν από την υπορουτίνα endsim. Αυτή βρίσκεται στο αρχείο end.f90:

```

!===== end.f90 =====
subroutine endsim()
  use global_data
  implicit none
  integer,parameter :: f_out = 17
  integer           :: i,seeds(25)

  call RLUXUT(seeds)
  call rename('conf','conf.old')
  open(unit=f_out,file='conf')
  write(f_out,'(A)')&
    '# Configuration of 2d Ising model on square lattice ...'
  write(f_out,'(A4,I5,A4,I5,A6,G28.17,A6,25I16)')&
    'Lx= ',L,' Ly= ',L,' beta= ',beta,' seed= ',seeds
  do i=1,N
    write(f_out,'(I3)')s(i)
  enddo
  close(f_out)
  print '(A,F7.3)', '# acceptance= ',&
    acceptance/DBLE(N)/DBLE(nswEEP)
end subroutine endsim

```

Καταρχήν, καλούμε την RLUXUT να σώσει την κατάσταση της RANLUX στο array seeds. Με την υπορουτίνα RENAME μετονομάζουμε, αν υπάρχει,

<sup>11</sup>δούμε όταν θα συζητήσουμε τους χρόνους αυτοσυσχετισμού.

<sup>12</sup>Χρησιμοποιούμε τους ισοδύναμους τελεστές σύγκρισης '>' ⇔ '.GT.', '>=' ⇔ '.GE.', '/=' ⇔ '.NE.', '==' ⇔ '.EQ.' κ.ο.κ.

το αρχείο `conf` σε `conf.old` για να μην καταστραφεί η παλιά διάταξη των σπιν από τη νέα. Στη συνέχεια, γράφουμε τις παραμέτρους της προσομοίωσης και το `seeds` με το format (A4, I5, A4, I5, A6, G28.17, A6, 25I16) το οποίο είναι ακριβώς το ίδιο με αυτό που χρησιμοποιούμε στην εντολή `READ` στην `init` προκειμένου να διαβάσουμε τη διάταξη των σπιν.

Τέλος, παραθέτουμε και τον κώδικα για τις υπορουτίνες `get_the_options()`, η οποία διαβάζει τις παραμέτρους μέσω `unix options`, `simmessage()`, που τυπώνει πληροφορίες που καθορίζουν τις συνθήκες της προσομοίωσης και `locerr()`, που σταματάει το πρόγραμμα τυπώνοντας ένα μήνυμα στο `stderr`. Η επιλογή να περάσουμε τις παραμέτρους μέσω `unix options` έχει τα πλεονεκτήματα ότι τις περνάμε προαιρετικά και με όποια σειρά θέλουμε. Ας δούμε, κατ' αρχήν, πώς δουλεύουν. Έστω ότι το εκτελέσιμο αρχείο του προγράμματος είναι το `is`. Η εντολή

```
> ./is -L 10 -b 0.44 -s 1 -S 5342 -n 1000
```

θα τρέξει το πρόγραμμα θέτοντας `L=10` (`-L 10`), `beta=0.44` (`-b 0.44`), `start=1` (`-s 1`), `seed=5342` (`-S 5342`) και `nsweep=1000` (`-n 1000`). Τα `-L`, `-b`, `-s`, `-S`, `-n` είναι τα `options` ή `switches` και μπορούμε να τα βάλουμε με όποια σειρά θέλουμε. Οι τιμές που ακολουθούν είναι οι τιμές που θα περάσουν στις τιμές των αντίστοιχων μεταβλητών. Υπάρχουν και `options` που δεν απαιτούν ορίσματα, αλλά μεταβάλλουν τον τρόπο που λειτουργεί το πρόγραμμα<sup>12</sup>. Στην περίπτωσή μας, το `option -h` (`help`) τυπώνει ένα σύντομο μήνυμα βοήθειας πάνω στη χρήση του προγράμματος και σταματάει το πρόγραμμα:

```
> ./is -h
Usage: ./is [options]
  -L: Lattice length (N=L*L)
  -b: beta
  -s: start (0 cold, 1 hot, 2 old config.)
  -S: seed
  -n: number of sweeps and measurements
  -u: seed from /dev/urandom
  -r: ranlux_level

Monte Carlo simulation of 2d Ising Model. Metropolis is used by
default. Using the options, the parameters of the simulations
must be set for a new run (start=0,1). If start=2, a
configuration is read from the file conf.
```

<sup>12</sup>Θυμηθείτε λ.χ. πώς το `-l` αλλάζει τον τρόπο λειτουργίας της εντολής `ls`, αν την εκτελέσουμε ως `ls -l`.

Έτσι μπορούμε να θυμόμαστε τον τρόπο χρήσης του προγράμματος.

Ας δούμε πώς προγραμματίζουμε τα παραπάνω. Παίρνουμε τον κώδικα από το αρχείο options.f90:

```
!===== options.f90 =====
subroutine get_the_options
  use global_data
  use getopt_m      !from getopt.f90
  implicit none
  call getarg(0,prog)

  do
    select case( getopt( "-hL:b:s:S:n:r:u" ))
    case( 'L' )
      read(optarg,*)L
    case( 'b' )
      read(optarg,*)beta
    case( 's' )
      read(optarg,*)start
    case( 'S' )
      read(optarg,*)seed
    case( 'n' )
      read(optarg,*)nsweep
    case( 'r' )
      read(optarg,*)ranlux_level
    case( 'u' )
      open (28, file="/dev/urandom", &
        access="stream", form="unformatted")
      read (28) seed
      seed = ABS(seed)
      close(28)
    case( 'h' )
      call usage
    case( '?' )
      print *, 'unknown option ', optopt
      stop
    case( char(0)) ! done with options
      exit
    case( '-' )   ! use -- to exit from options
      exit
    case default
      print *, 'unhandled option ', optopt
    end select
  enddo
end subroutine get_the_options
```

Με την εντολή call getarg(0,prog) αποθηκεύουμε το όνομα του προ-

γράμματος στη μεταβλητή `character prog`. Η συνάρτηση `getopt` είναι μια συνάρτηση προγραμματισμένη από τον Mark Gates που ο κώδικάς της είναι στο αρχείο `getopt.f90`. Είναι γραμμένη έτσι, ώστε να προσομοιάζει, ως προς τον τρόπο χρήσης, την αντίστοιχη συνάρτηση της C<sup>13</sup>. Το όρισμά της `"-hL:b:s:S:n:r:u"` ορίζει ως επιτρεπτά options τους χαρακτήρες '-', 'L', 'b', 's', 'S', 'n', 'r', 'u'. Κάθε φορά που ο χρήστης τους περνάει από τη γραμμή εντολών (ως λ.χ. `-L 100`, `-h`), το `do loop` μας πάει στο αντίστοιχο CASE από όπου εκτελούμε τις εντολές που θέλουμε. Αν κάποιο option δεν παίρνει argument (λ.χ. `-h`), τότε εκτελείται μια εντολή όπως η `call usage`. Αν ένα option θέλουμε να πάρει argument, τότε στο όρισμα της `getopt` μπαίνει μαζί με ':' (λ.χ. `L:`, `b:`, ...) και το argument είναι προσβάσιμο από το πρόγραμμα μέσω της μεταβλητής `character optarg`. Λ.χ. με τις γραμμές

```
case( 'L' )
  read(optarg,*)L
```

η γραμμή εντολών που περιέχει `-L 10` έχει σαν αποτέλεσμα η `optarg` να έχει την τιμή '10'. Αυτό δεν είναι αριθμός, αλλά ακολουθία χαρακτήρων! Για να μετατρέψουμε το `character '10'` στον `integer 10`, χρησιμοποιούμε την εντολή `READ` σα να διαβάζαμε από ... αρχείο. Το ίδιο κάνουμε και με τις άλλες παραμέτρους.

Η υπορουτίνα `locerr` παίρνει στην είσοδο ένα μήνυμα σφάλματος, το τυπώνει στο `stderr` με το όνομα του προγράμματος και σταματάει το πρόγραμμα:

```
subroutine locerr(errmes)
  use global_data
  implicit none
  character(*) :: errmes
  write(0, '(A,A)') ,TRIM(prog),':',TRIM(errmes), ' Exiting .... '
  stop 1
end subroutine locerr
```

Προσέξτε τη χρήση της πολύ χρήσιμης συνάρτησης `TRIM`. Αυτή “κόβει” τα τελευταία κενά μιας μεταβλητής `character`. Αν δεν τη χρησιμοποιούσαμε, η μεταβλητή `character(1024) :: prog` θα έπαιρνε στην εκτύπωση 1024 θέσεις χαρακτήρων, κάτι που δε θα μας άρεσε και πολύ...

Η υπορουτίνα `usage` παίζει και αυτή πολύ σημαντικό ρόλο: Μας δίνει σύντομες πληροφορίες για τη χρήση του προγράμματος και σύντομα θα

<sup>13</sup> Διαβάστε τα σχόλια στο αρχείο `getopt.f90` για περισσότερες πληροφορίες.

την εκτιμήσετε πολύ! Η σωστή τεκμηρίωση του προγράμματός μας είναι αναπόσπαστο μέρος της δουλειάς μας...

```
subroutine usage
  use global_data
  implicit none
  print '(3A)', 'Usage: ', TRIM(prog), ' [options]'
  print '( A)', '      -L: Lattice length (N=L*L)'
  print '( A)', '      -b: beta'
  print '( A)', '      -s: start (0 cold, 1 hot, 2 old config.)'
  print '( A)', '      -S: seed'
  print '( A)', '      -n: number of sweeps and measurements'
  print '( A)', '      -u: seed from /dev/urandom'
  print '( A)', '      -r: ranlux_level'
  print '( A)', 'Monte Carlo simulation of 2d Ising Model....'
  stop
end subroutine usage
```

Τέλος, η υπορουτίνα `simmessage` τυπώνει πληροφορίες για την προσομοίωση που τρέχουμε. Δε χρειάζεται να τονίσουμε πόσο σημαντικό είναι αυτό: Δεδομένα χωρίς ετικέτα είναι καλύτερα να τα πετάμε, γιατί είναι επικίνδυνα, όπως κάνουμε και με τα ... φάρμακα. Εκτός από τις παραμέτρους της προσομοίωσης, τυπώνουμε και πληροφορίες από το περιβάλλον που τρέχει το πρόγραμμα: Το όνομα του υπολογιστή, το είδος του λειτουργικού συστήματος, την ημερομηνία και το όνομα του χρήστη που έτρεξε το πρόγραμμα. Όλες αυτές οι πληροφορίες μπορεί να μας γίνουν πολύτιμες. Αλλάζοντας την μεταβλητή `unit`, μπορούμε να τυπώσουμε το μήνυμα σε όποιο αρχείο θέλουμε.

```
subroutine simmessage(unit)
  use global_data
  implicit none
  integer :: unit
  character(100) :: user, host, mach, tdate
  call GETLOG(user)
  call GETENV('HOST', host)
  call GETENV('HOSTTYPE', mach)
  call FDATE(tdate)
  write(unit, '( A )')&
    '# #####'
  write(unit, '( A )')&
    '# 2d Ising Model Metropolis algorithm on square lattice'
  write(unit, '( 8A )')&
    '# Run on ', TRIM(host), ' ( ', TRIM(mach), ' ) by ', TRIM(user), &
    ' on ', TRIM(tdate)
```



```

write(unit, '( A,I6,A )')'# L      = ',L, ' (N=L*L) '
write(unit, '( A,I14 )')'# seed    = ',seed
write(unit, '( A,I12,A )')'# nsweeps = ',nsweep, ' (No. sweeps) '
write(unit, '( A,G28.17 )')'# beta  = ',beta
write(unit, '( A,I4 ,A )')'# start  = ',start,&
' (0 cold, 1 hot, 2 old config) '
end subroutine simmessage

```

Για τη μεταγλώττιση του κώδικα απαιτείται η εντολή:

```

> gfortran global_data.f90 getopt.f90 \
  main.f90 init.f90 met.f90 measure.f90 end.f90 \
  options.f90 ranlux.F -o is

```

Είναι σημαντικό ότι τα αρχεία με τα modules `global_data.f90`, `getopt.f90` προηγούνται των αρχείων με κώδικα που χρησιμοποιεί τα modules που περιέχουν.

Για να τρέξουμε το πρόγραμμα, περνάμε τις παραμέτρους μέσω των options, όπως για παράδειγμα:

```

> /usr/bin/time ./is -L 10 -b 0.44 -s 1 -S 5342 -n 10000 \
  >& out.dat &

```

όπου προσθέσαμε και την εντολή `time` για τη μέτρηση των αποδόσεων του προγράμματος.

Ένα χρήσιμο εργαλείο για τη μεταγλώττιση του προγράμματος είναι το πρόγραμμα `make`. Θα μπορούσε να γραφτεί ολόκληρο κεφάλαιο για το `make`, όμως παραπέμπουμε τον ενδιαφερόμενο αναγνώστη στις σχετικές `info pages`<sup>14</sup>. Ας αρκεστούμε να πούμε πως, αν στον κατάλογο που βρίσκεστε, υπάρχει ένα αρχείο με όνομα `Makefile` που έχει περιεχόμενα<sup>15</sup>:

```

# ##### Makefile #####
FC      = gfortran
OBJS    = global_data.o getopt.o \
          main.o init.o met.o measure.o end.o \
          options.o ranlux.o

```

<sup>14</sup>Με την εντολή: `info make` ή στο διαδίκτυο στη διεύθυνση [www.gnu.org/software/make/manual/make.html](http://www.gnu.org/software/make/manual/make.html)

<sup>15</sup>Προσοχή, ένα από τα ... βίτσια του `make` είναι ότι οι εκτελέσιμες εντολές (`$(FC)`, `/bin/rm` στο παράδειγμα) βρίσκονται σε μια γραμμή που αρχίζει από TAB και όχι από κενά ή άλλους χαρακτήρες. Άρα, στο παραπάνω παράδειγμα `Makefile` ο κενός χώρος που φαίνεται στις γραμμές αυτές είναι ένα TAB και όχι 8 κενοί χαρακτήρες.

```
FFLAGS = -O2

is: $(OBJS)
    $(FC) $(FFLAGS) $^ -o $@

$(OBJS):    global_data.f90
options.o:  getopt.f90
%.o: %.f90
    $(FC) $(FFLAGS)    -c -o $@ $<
```

είναι αρκετό για να “φτιάξει” το εκτελέσιμο αρχείο `is`. Τι κερδίσαμε; Για να δείτε, κάντε μια μετατροπή σε ένα από τα αρχεία του κώδικα, λ.χ. στο `main.f90` και εκτελέστε την εντολή `make` ξανά. Θα δείτε πως μόνο το αρχείο που μεταβάλατε ξαναμεταγλωττίζεται και όχι τα άλλα. Θέτοντας κανόνες εξάρτησης (dependencies), μπορείτε να πετύχετε ακόμα πιο πολύπλοκες μεταγλωττίσεις. Με τη χρήση μεταβλητών μπορούμε να κωδικοποιήσουμε τη σύνδεση με βιβλιοθήκες, επίπεδα βελτιστοποίησης, μεταγλώττιση υπό συνθήκες (αρχιτεκτονική, μεταγλωττίστης κλπ) και άλλα ... πολλά! Ενθαρρύνουμε τον φοιτητή με ενδιαφέρον στον προγραμματισμό να επενδύσει λίγο χρόνο για να μάθει περισσότερα<sup>16</sup>.

## 13.4 Θερμική Ισορροπία

Στην ενότητα αυτή θα συζητήσουμε τα τεχνικά προβλήματα που συναντά μια προσομοίωση μέχρι να φέρουμε το σύστημα σε θερμική ισορροπία. Το πρόβλημα αυτό μπορεί να είναι από αμελητέο μέχρι και απαγορευτικό για την προσομοίωση κάποιων συστημάτων. Έτσι παρόλο που δεν είναι ιδιαίτερα σημαντικό πρόβλημα για το δισδιάστατο πρότυπο Ising, θα το μελετήσουμε λόγω της σπουδαιότητάς του για άλλα προβλήματα.

Όπως ήδη είδαμε, όταν ξεκινούμε μια προσομοίωση, χρειάζεται να τοποθετήσουμε το σύστημά σε μια αρχική κατάσταση, έτσι ώστε με τον αλγόριθμο που έχουμε επιλέξει να ξεκινήσουμε μια διαδικασία Markov για την παραγωγή του δείγματός μας. Γνωρίζουμε όμως από την ενότητα 12.2, πως όταν το σύστημα είναι σε θερμική ισορροπία σε μια δεδομένη θερμοκρασία, τότε βρίσκεται κυρίως σε καταστάσεις με ενέργεια που διαφέρει λίγο από τη μέση τιμή της. Καταστάσεις με ενέργεια

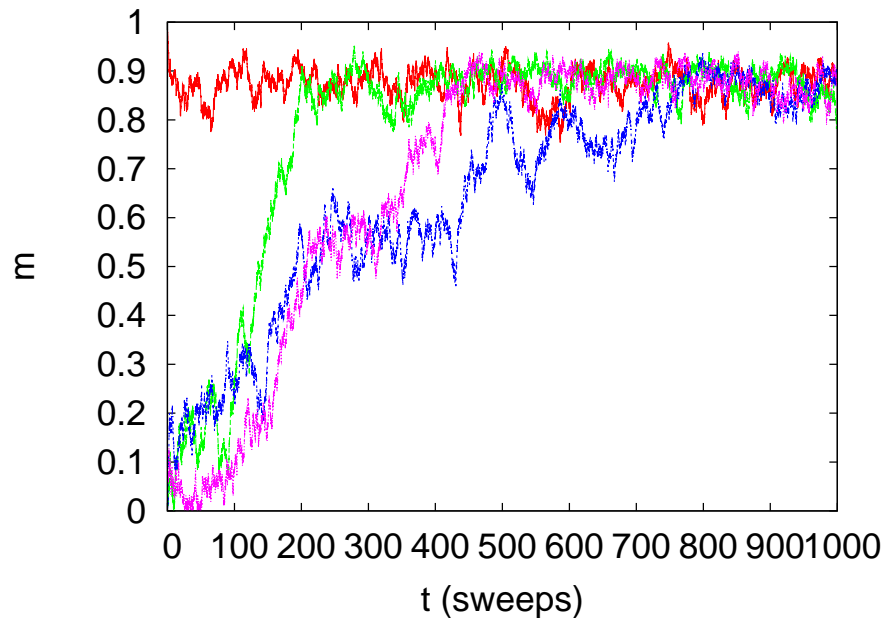
<sup>16</sup>Την εποχή που γραφόταν το βιβλίο, το `make` είχε ορισμένα προβλήματα με πολύπλοκα προγράμματα γραμμένα σε Fortran. Μπορείτε να δείτε και το πρόγραμμα `foray` στο [code.google.com/p/foraytool](http://code.google.com/p/foraytool) ή να αναζητήσετε την τρέχουσα “state of the art”.

που διαφέρει κατά πολύ από τη μέση τιμή της, τις επισκέπτεται πολύ σπάνια<sup>17</sup>. Έτσι, αν η αρχική κατάσταση που θα επιλέξουμε δεν είναι μια τυπική κατάσταση, θα πρέπει το σύστημά μας να κάνει έναν τυχαίο ... περίπατο στον χώρο των καταστάσεων μέχρι να “βρει” την περιοχή των καταστάσεων εκείνη που αντιστοιχεί στη θερμοκρασία που έχουμε επιλέξει. Αυτή είναι η διαδικασία της εύρεσης της κατάστασης θερμικής ισορροπίας (thermalization).

Εδώ παρουσιάζονται δύο προβλήματα: Το πρώτο είναι η κατάλληλη επιλογή της αρχικής κατάστασης και το δεύτερο είναι να εντοπίσουμε πότε το σύστημά μας έχει βρεθεί σε κατάσταση θερμικής ισορροπίας. Ως προς το πρώτο, έχουμε ήδη δει πως η αρχική κατάσταση για το πρότυπο Ising μπορεί να επιλεγεί να είναι (α) παγωμένη, (β) θερμή, (γ) σε μια άλλη - πιθανώς γειτονική - θερμοκρασία από μια προηγούμενη προσομοίωση. Είναι φυσικά κατανοητό πως, αν επιλέξουμε την αρχική κατάσταση να είναι λ.χ. καυτή για να προσομοιώσουμε το σύστημα σε μια χαμηλή θερμοκρασία, το σύστημα θα αργήσει περισσότερο να βρεθεί σε θερμική ισορροπία από ότι αν επιλέγαμε την αρχική κατάσταση να είναι παγωμένη (ή σε κάποια κοντινή θερμοκρασία). Αυτό φαίνεται καθαρά στο σχήμα 13.7. Η δυσκολία της διαδικασίας εξαρτάται από τη φυσική ποσότητα, τη θερμοκρασία και το μέγεθος του συστήματος. Η ενέργεια βρίσκει πιο γρήγορα τις τιμές που παίρνει στην κατάσταση θερμικής ισορροπίας, παρά η μαγνήτιση. Μια τοπική ποσότητα (local quantity, fast mode) μελετάται ευκολότερα από μια που έχει μη τοπικές εξαρτήσεις (non--local quantity, slow mode). Η κατάσταση θερμικής ισορροπίας βρίσκεται ευκολότερα, όταν είμαστε μακριά από την κρίσιμη θερμοκρασία. Είναι ευκολότερο να φέρουμε σε θερμική ισορροπία ένα μικρό σύστημα, παρά ένα μεγάλο. Το δεύτερο πρόβλημα είναι να εντοπίσουμε πότε το σύστημά έχει βρεθεί σε κατάσταση θερμικής ισορροπίας και φυσικά να αγνοήσουμε όλες τις μετρήσεις που έχουμε κάνει πιο πριν. Από το σχήμα 13.7 βλέπουμε πως η στιγμή αυτή, ακόμα και για την ίδια αρχική κατάσταση, μπορεί να έχει τιμή με μεγάλες διακυμάνσεις. Ο πιο απλός τρόπος είναι να ξεκινήσουμε αρκετές προσομοιώσεις με διαφορετικές αρχικές καταστάσεις ή/και από την ίδια αρχική κατάσταση, αλλά ακολουθώντας διαφορετική στοχαστική ... πορεία και να παρακολουθήσουμε την εξέλιξη στον (Μόντε Κάρλο) χρόνο των προς μέτρηση φυσικών ποσοτήτων. Όταν όλα τα μονοπάτια συναντηθούν, τότε μάλλον έχουμε πετύχει θερμική ισορροπία και μπορούμε να ξεκινήσουμε τις μετρήσεις μας.

Πιο συστηματικά όμως αποκτούμε πεποίθηση ότι όντως έχουμε πε-

<sup>17</sup>Φυσικά αυτό γίνεται εντονότερο όσο το σύστημά μας έχει μεγαλύτερο μέγεθος.

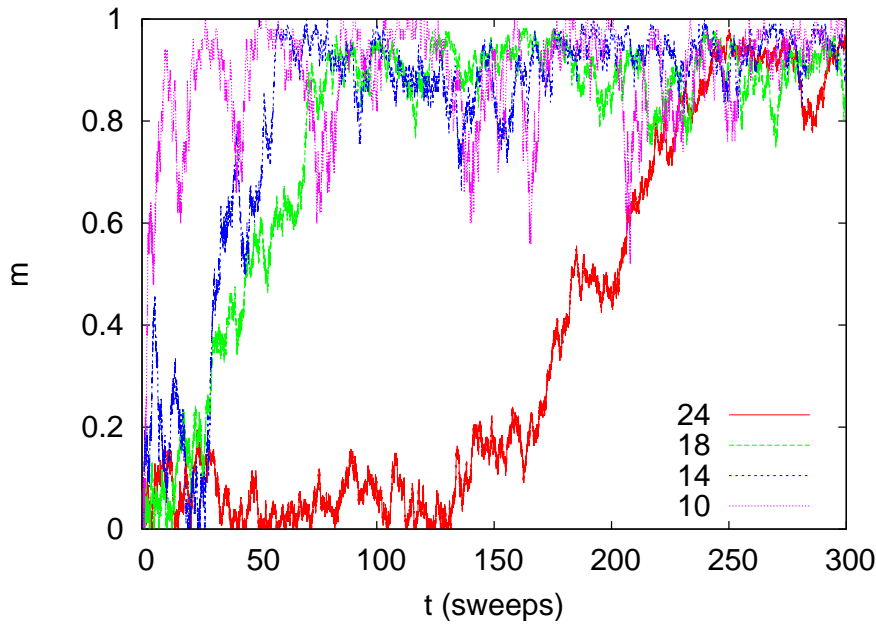


Σχήμα 13.7: Μαγνήτιση ανά πλεγματική θέση για το πρότυπο Ising με  $L = 40$ ,  $\beta = 0.48$ . Φαίνεται η διαδικασία εύρεσης θερμικής ισορροπίας για μια αρχική κατάσταση παγωμένη και τρεις αρχικές καταστάσεις καυτές. Επαναλαμβάνοντας τη διαδικασία αυτή για τη δεύτερη περίπτωση, φαίνεται πως για βρεθεί το σύστημα σε κατάσταση θερμικής ισορροπίας χρειαζόμαστε περισσότερα από  $\sim 1000$  sweeps.

τύχει θερμική ισορροπία, όταν τα αποτελέσματά μας έχουν σταθερή τιμή (μέσα στα όρια του στατιστικού σφάλματος), καθώς αποκόπτουμε από το δείγμα μας όλο και μεγαλύτερο αριθμό από μετρήσεις. Η διαδικασία φαίνεται στο σχήμα 13.10 όπου για τις μετρήσεις του πάνω σχήματος αποκόπτουμε διαδοχικά 0, 20, 50, 100, 200, 400, 800, 1600, 3200 και 6400 αρχικές μετρήσεις μέχρι η μέση τιμή της μαγνήτισης να πάρει μια σταθερή τιμή μέσα στα όρια του στατιστικού σφάλματος.

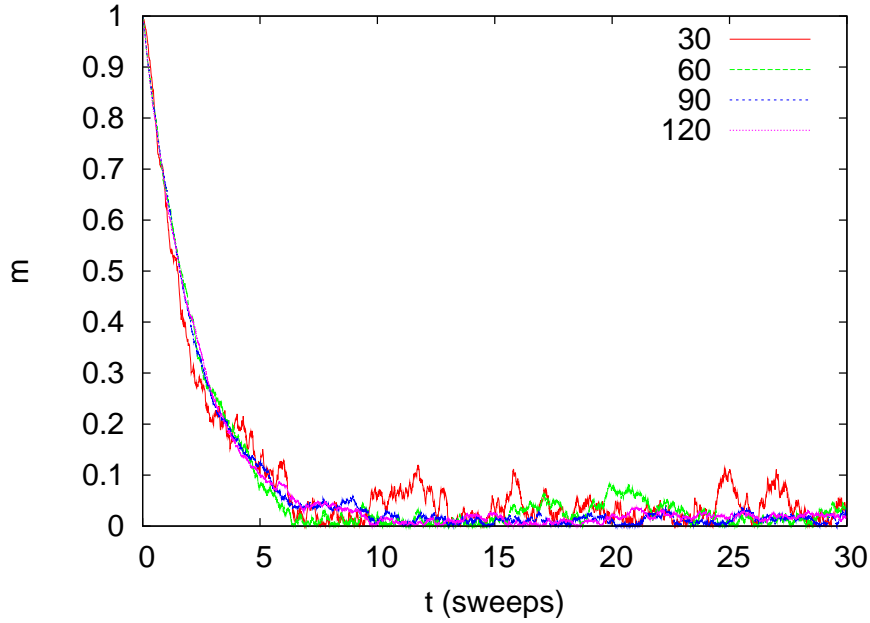
### 13.5 Αυτοσυσχετισμοί

Για να κατασκευάσουμε ένα δείγμα από στατιστικά ανεξάρτητες μετρήσεις από μια διαδικασία Markov θα πρέπει το επόμενο βήμα να μην έχει στατιστικό συσχετισμό με το προηγούμενο. Είναι, όμως, φανερό πως με τους αλγόριθμους που χρησιμοποιούμε αυτό δεν είναι δυνατόν. Στον αλγόριθμο Metropolis το επόμενο βήμα διαφέρει από το προηγούμενο το πολύ κατά την τιμή ενός σπιν. Άρα, η επόμενη διάταξη των



Σχήμα 13.8: Μαγνήτιση ανά πλεγματική θέση για το πρότυπο Ising με  $L = 10, 14, 18, 24$  και  $\beta = 0.50$ . Φαίνεται η διαδικασία εύρεσης θερμικής ισορροπίας για μια αρχική κατάσταση καυτή και ότι η δυσκολία αυξάνει με το μέγεθος του συστήματος.

σπιν είναι στατιστικά πολύ ισχυρά συσχετισμένη με την προηγούμενη. Θα περίμενε κανείς να πάρει μία στατιστικά ανεξάρτητη διάταξη, όταν κάνει ένα βήμα Metropolis ανά πλεγματική θέση, αυτό που ονομάζεται ένα “sweep” του πλέγματος. Ενώ αυτό όντως συμβαίνει στην πράξη για πολλές περιπτώσεις, λ.χ. στο πρότυπο Ising για θερμοκρασίες αρκετά διαφορετικές από την κρίσιμη  $\beta_c$ , στις περισσότερες ενδιαφέρουσες περιπτώσεις χρειαζόμαστε πολύ μεγαλύτερο αριθμό από sweeps για να πάρουμε μια στατιστικά ανεξάρτητη διάταξη. Αυτό γίνεται λ.χ. όταν το μήκος συσχετισμού  $\xi$  (12.46) γίνεται πολύ μεγαλύτερο από μια πλεγματική θέση, κάτι που συμβαίνει στην κρίσιμη περιοχή μιας συνεχούς μετάβασης φάσης. Αυτό μπορεί να γίνει κατανοητό από τα σχήματα 13.36. Καθώς πλησιάζουμε την κρίσιμη θερμοκρασία από τη θερμή περιοχή όπου τα σπιν είναι σε αταξία, δημιουργούνται μεγάλες περιοχές (clusters) από όμοια σπιν. Καταλαβαίνουμε ότι για να έχουμε δύο ασυσχέτιστες διατάξεις σπιν στην περίπτωση αυτή, πρέπει το μέγεθος, το σχήμα και η θέση αυτών των clusters να είναι ασυσχέτιστα. Για να γίνει κάτι τέτοιο, πρέπει να καταστραφούν τα υπάρχοντα clusters και να δημιουργηθούν κάπου αλλού. Με ένα single-flip αλγόριθμο όπως ο Metropolis η διαδικασία αυτή παίρνει αρκετό χρόνο.



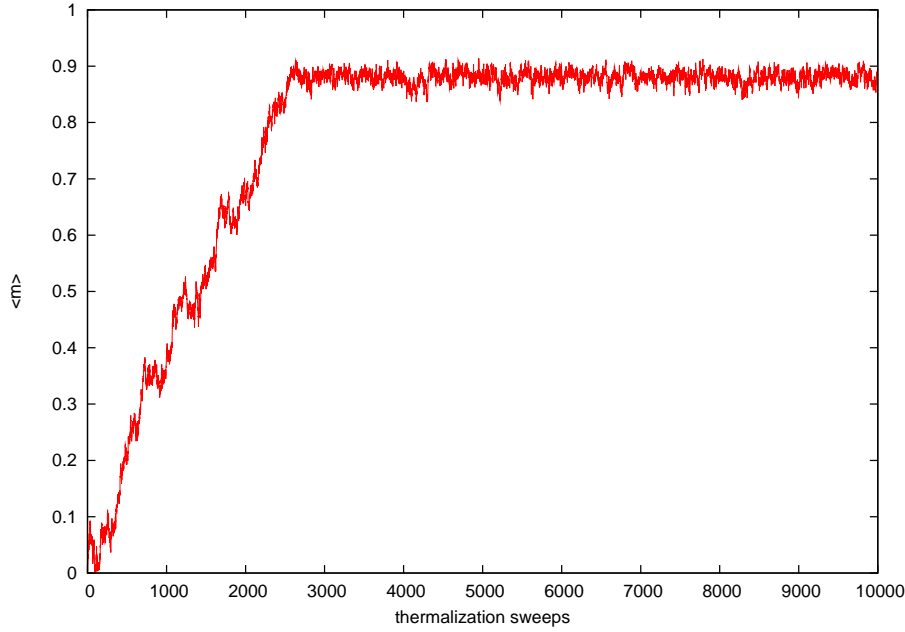
Σχήμα 13.9: Μαγνήτιση ανά πλεγματική θέση για το πρότυπο Ising με  $L = 30, 60, 90, 120$  και  $\beta = 0.20$ . Φαίνεται η διαδικασία εύρεσης θερμικής ισορροπίας για μια αρχική κατάσταση παγωμένη. Εδώ, η δυσκολία δεν αυξάνει με το μέγεθος του συστήματος.

Για την ποσοτική μελέτη του αυτοσυσχετισμού των διατάξεων σπιν που παράγονται στην προσομοίωσή μας, χρησιμοποιούμε τη συνάρτηση αυτοσυσχετισμού (autocorrelation function). Έστω μια φυσική ποσότητα  $\mathcal{O}$  (λ.χ. ενέργεια, μαγνήτιση, μαγνητική επιδεκτικότητα κλπ) και  $\mathcal{O}(t)$  η τιμή της μετά από Μόντε Κάρλο “χρόνο”  $t$ . Οι μονάδες του  $t$  μπορεί να είναι sweeps ή (υπο)πολλαπλάσια των sweeps. Η συνάρτηση αυτοσυσχετισμού  $\rho_{\mathcal{O}}(t)$  της φυσικής ποσότητας  $\mathcal{O}$  ορίζεται για κάθε χρόνο  $t$  να είναι η

$$\rho_{\mathcal{O}}(t) = \frac{\langle (\mathcal{O}(t') - \langle \mathcal{O} \rangle)(\mathcal{O}(t' + t) - \langle \mathcal{O} \rangle) \rangle_{t'}}{\langle (\mathcal{O} - \langle \mathcal{O} \rangle)^2 \rangle}, \quad (13.34)$$

όπου  $\langle \dots \rangle_{t'}$  είναι η μέση τιμή πάνω σε όλα τα στοιχεία του δείγματος με  $t' < t_{\max} - t$ . Η κανονικοποίηση που βάζουμε στον παρονομαστή είναι τέτοια, ώστε  $\rho_{\mathcal{O}}(0) = 1$ .

Ο ορισμός θυμίζει τη συνάρτηση συσχετισμού των σπιν στον χώρο (βλ. εξ. (12.45)). Η συζήτηση για τη σημασία του ορισμού είναι παράλληλη με αυτή της ενότητας 12.4. Με λίγα λόγια, όταν η τιμή μιας ποσότητας μετά από χρόνο  $t$  είναι ισχυρά συσχετισμένη με την αρχική,



Σχήμα 13.10: Μαγνήτιση ανά πλεγματική θέση για το πρότυπο Ising με  $L = 100$  και  $\beta = 0.48$ . Φαίνεται η διαδικασία εύρεσης θερμικής ισορροπίας για μια αρχική κατάσταση καυτή.

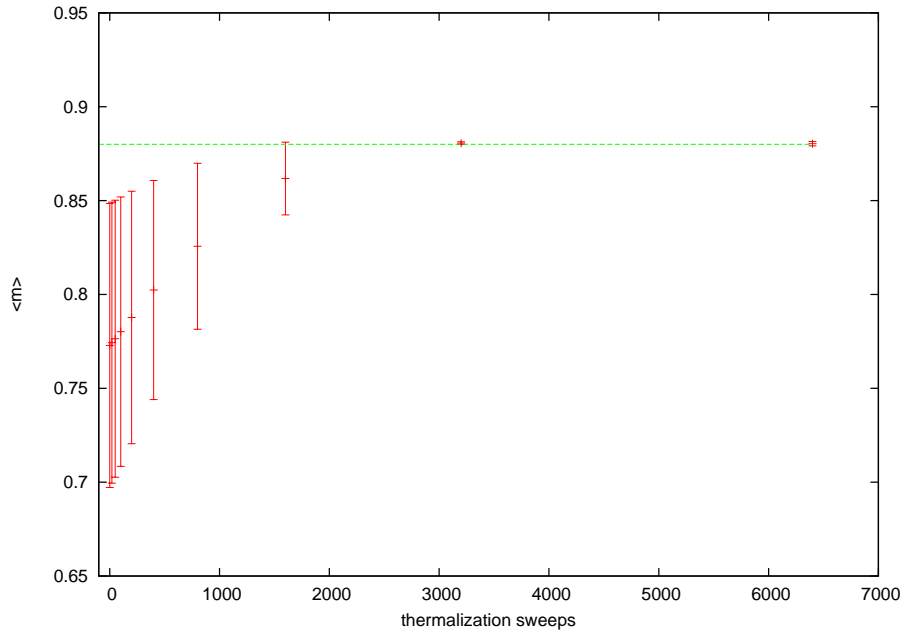
τότε το γινόμενο στον αριθμητή της (13.34) θα είναι σχεδόν πάντοτε θετικό, με αποτέλεσμα η μέση τιμή να είναι θετική. Όταν ο συσχετισμός είναι ασθενής, τότε το γινόμενο θα είναι πότε θετικό και πότε αρνητικό στο δείγμα, έτσι ώστε ο μέσος όρος να είναι μικρός. Αν υπάρχει αντισυσχετισμός, τότε το κλάσμα είναι αρνητικό. Αρνητικές τιμές για την  $\rho_O(t)$  οφείλονται στο πεπερασμένο μέγεθος του δείγματος και πρέπει να απορρίπτονται.

Ασυμπτωτικά, η συνάρτηση  $\rho_O(t)$  ελαττώνεται με εκθετικό τρόπο

$$\rho_O(t) \sim e^{-t/\tau_O} . \quad (13.35)$$

Η κλίμακα χρόνου  $\tau_O$  χαρακτηρίζει τον αποσυσχετισμό των μετρήσεων της  $O$  και ονομάζεται χρόνος αυτοσυσχετισμού της  $O$ . Μετά από χρόνο  $2\tau_O$ , η  $\rho_O(t)$  έχει πέσει στο  $1/e^2 \approx 14\%$  της αρχικής της τιμής. Συμβατικά θεωρούμε ότι τότε έχουμε μία ανεξάρτητη μέτρηση της  $O$ <sup>18</sup>. Άρα, αν έχουμε  $t_{\max}$  μετρήσεις, ο αριθμός των ανεξάρτητων μετρήσεων της  $O$

<sup>18</sup>Οι χρόνοι αυτοσυσχετισμού μπορεί να είναι πολύ διαφορετικοί για διαφορετικές φυσικές ποσότητες.



Σχήμα 13.11: Μαγνήτιση ανά πλεγματική θέση για το πρότυπο Ising με  $L = 100$  και  $\beta = 0.48$ . Υπολογίζουμε τη μέση τιμή  $\langle m \rangle$  αποκόπτοντας “thermalization sweeps” μετρήσεις και τις δείχνουμε στο δεύτερο σχήμα μαζί με τα αντίστοιχα σφάλματα. Παρατηρούμε ότι όταν αποκόψουμε ικανό αριθμό από αρχικές μετρήσεις, σε συμφωνία με το πάνω σχήμα, το αποτέλεσμα σταθεροποιείται [ $\langle m \rangle = 0.880(1)$ ] και βρίσκουμε πως το σύστημα είναι σε θερμική ισορροπία.

θα είναι

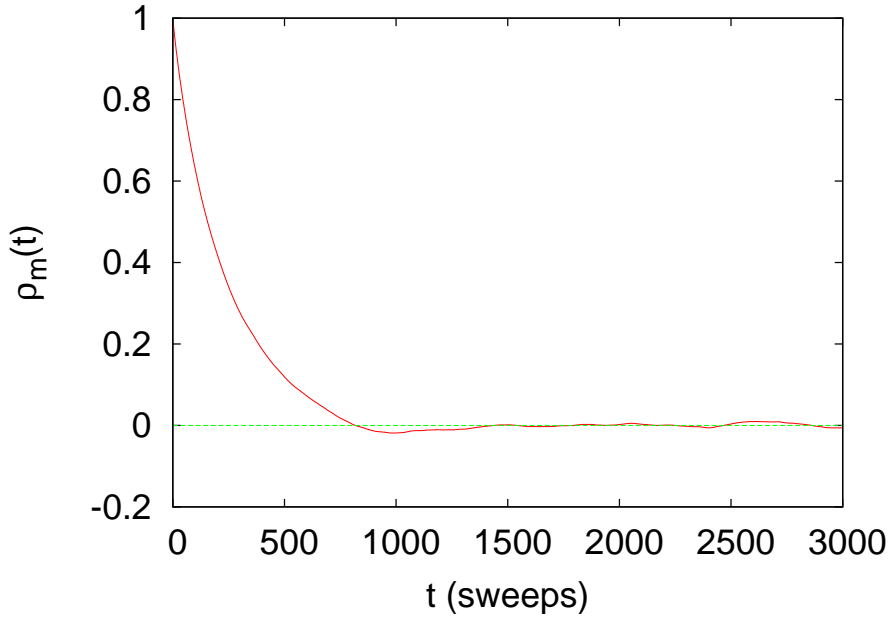
$$n_{\mathcal{O}} = \frac{t_{\max}}{2\tau_{\mathcal{O}}}. \quad (13.36)$$

Αν μία μέτρηση κοστίζει σε χρόνο, τότε από τα παραπάνω βλέπουμε ότι θα τη μετράμε σε διαστήματα χρόνου  $\sim \tau_{\mathcal{O}}$ . Αν όμως το κόστος της μέτρησης δεν είναι μεγάλο, τότε μετράμε συχνότερα. Αυτό συμβαίνει γιατί υπάρχει στατιστική πληροφορία και στις μετρήσεις που είναι ελαφρά συσχετισμένες. Η ακριβής μέτρηση του  $\tau_{\mathcal{O}}$ , ειδικά αν είναι μεγάλο, είναι δύσκολη, γιατί πρέπει να πάρουμε μετρήσεις για  $t \gg \tau_{\mathcal{O}}$ .

Ένα παράδειγμα φαίνεται στο σχήμα 13.12 για την περίπτωση της μαγνήτισης ( $\mathcal{O} = m$ ). Υπολογίζουμε τη συνάρτηση  $\rho_m(t)$  και βλέπουμε ότι η προσαρμογή στην (13.35) είναι πολύ καλή με  $\tau_m = 235 \pm 3$  sweeps. Η μέτρηση έγινε σε ένα δείγμα από  $10^6$  μετρήσεις με 1 μέτρηση/sweep. Άρα, οι ανεξάρτητες μετρήσεις μας ήταν στην πραγματικότητα  $\approx 10^6 / (2 \times 235) \approx 2128$ .

Ένας άλλος εκτιμητής του χρόνου αυτοσυσχετισμού είναι ο ολο-





Σχήμα 13.12: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης  $\rho_m(t)$  για το πρότυπο Ising και  $L = 100$ ,  $\beta = 0.42$ . Φαίνεται η εκθετική πτώση της και ότι  $\tau_m \approx 200$  sweeps. Φαίνονται και τα φαινόμενα πεπερασμένου μεγέθους το δείγματος (1,000,000 μετρήσεις) όταν η  $\rho$  αρχίζει να έχει διακυμάνσεις γύρω από το 0.

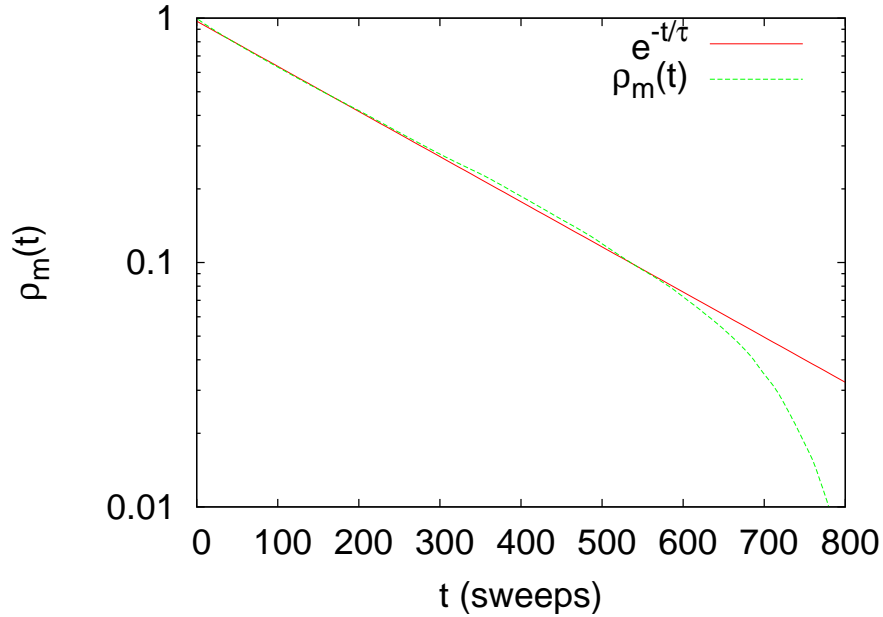
κλιρωμένος χρόνος αυτοσυσχετισμού  $\tau_{\text{int},\mathcal{O}}$  (integrated autocorrelation time). Ο ορισμός του προκύπτει από την υπόθεση (13.35) από όπου παίρνουμε

$$\tau_{\text{int},\mathcal{O}} = \int_0^{+\infty} dt \rho_{\mathcal{O}}(t) \sim \int_0^{+\infty} dt e^{-t/\tau_{\mathcal{O}}} = \tau_{\mathcal{O}}. \quad (13.37)$$

Προφανώς  $\tau_{\text{int},\mathcal{O}} \sim \tau_{\mathcal{O}}$ , αλλά στον υπολογισμό υπεισέρχονται συστηματικά σφάλματα, γι' αυτό οι δύο ορισμοί μπορεί να δίνουν τιμές που διαφέρουν λίγο στην πράξη. Αν το δείγμα μας είναι καλό, τότε θα έχουμε καλύτερη συμφωνία. Επίσης, επειδή το δείγμα μας δεν είναι άπειρο, πρέπει να επιλέξουμε ένα  $t_{\text{max}}$  όπου θα κόψουμε την ολοκλήρωση<sup>19</sup>

$$\tau_{\text{int},\mathcal{O}}(t_{\text{max}}) = \int_0^{t_{\text{max}}} dt \rho_{\mathcal{O}}(t), \quad (13.38)$$

<sup>19</sup>Για μικρούς χρόνους  $t_{\text{max}}$  δε θα έχουμε ακριβώς τη συμπεριφορά (13.35) και αυτό θα προσθέσει στο συστηματικό σφάλμα. Αλλά παρόμοια προβλήματα έχουμε και όταν προσπαθούμε να υπολογίσουμε τον  $\tau_{\mathcal{O}}$  με προσαρμογή των δεδομένων: Πρέπει να γίνει εκτίμηση των ορίων στο διάστημα του χρόνου που θα γίνει η προσαρμογή που θα αντιστοιχεί στην (13.35), κάτι το οποίο αποτελεί ... τέχνη!

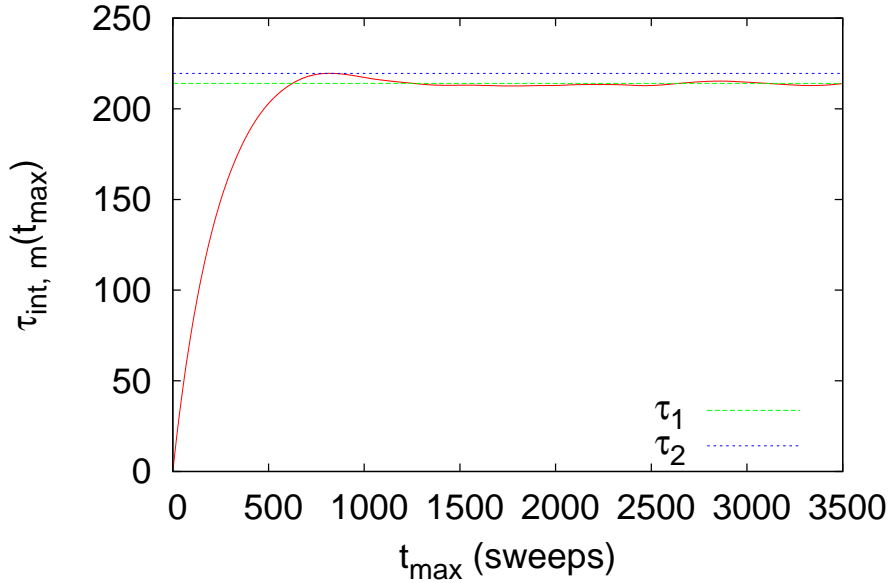


Σχήμα 13.13: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης  $\rho_m(t)$  για το πρότυπο Ising και  $L = 100$ ,  $\beta = 0.42$ . Γίνεται προσρμογή στην εκθετική πτώση  $Ce^{-t/\tau}$  (βλ. εξ. (13.35)) με  $\tau = 235(3)$  sweeps.

και κάνουμε τη γραφική παράσταση  $\tau_{\text{int},\mathcal{O}}(t_{\text{max}})$ . Όταν παρατηρήσουμε πως η τιμή του  $\tau_{\text{int},\mathcal{O}}(t_{\text{max}})$  σταθεροποιείται για κάποιο διάστημα του  $t_{\text{max}}$ , παίρνουμε αυτή την τιμή ως εκτιμητή του  $\tau_{\text{int},\mathcal{O}}$ .

Η διαδικασία αυτή φαίνεται στο σχήμα 13.14, όπου χρησιμοποιήσαμε τις ίδιες μετρήσεις με αυτές του σχήματος 13.12. Το αποτέλεσμα είναι  $\tau_{\text{int},m} = 217(3)$  sweeps που είναι μικρότερο από αυτό που υπολογίσαμε προηγουμένως. Η διαφορά αυτή συχνά δεν είναι σημαντική. Συνήθως, αυτό που μας ενδιαφέρει είναι η εξάρτηση του χρόνου αυτοσυσχετισμού από τις παραμέτρους του συστήματος, εδώ το μέγεθος  $L$  και τη θερμοκρασία  $\beta$ . Και οι δύο εκτιμητές θα μας δώσουν το ίδιο αποτέλεσμα για την κύρια (ασυμπτωτική) συμπεριφορά για αρκετά μεγάλα  $L$  και για  $\beta$  αρκετά κοντά στην κρίσιμη θερμοκρασία  $\beta_c$ . Συνήθως, προτιμάμε να υπολογίσουμε τον  $\tau_{\text{int},\mathcal{O}}$  λόγω της ευκολότερης διαδικασίας.

Ο χρόνος αυτοσυσχετισμού δεν είναι σοβαρό πρόβλημα, όταν το σύστημα βρίσκεται μακριά από την κρίσιμη περιοχή θερμοκρασιών. Στο σχήμα 13.15 βλέπουμε πως αυτός είναι μόνο μερικά sweeps και ότι το γεγονός αυτό είναι ανεξάρτητο από το μέγεθος του συστήματος  $L$ . Όταν όμως πλησιάζουμε την κρίσιμη περιοχή, ο χρόνος αυτοσυσχετισμού μεγαλώνει. Στην κρίσιμη περιοχή η αύξηση αυτή παρουσιάζει



Σχήμα 13.14: Υπολογισμός του ολοκληρωμένου χρόνου αυτοσυσχετισμού της μαγνήτισης (integrated autocorrelation time) για τα δεδομένα του σχήματος 13.19. Παρατηρούμε ένα πλατό στις τιμές του  $\tau_{int, m}$  για  $\tau_1 = 214(1)$  sweeps, αλλά και ένα μέγιστο για  $\tau_2 \approx 219.5$  sweeps. Η πτώση των τιμών από  $\tau_1$  σε  $\tau_2$  οφείλεται σε αρνητικές τιμές της  $\rho_m(t)$  λόγω θορύβου του δείγματος των μετρήσεων. Άρα, θα πάρουμε  $\tau_{int, m} = 217(3)$  sweeps να είναι το αποτέλεσμα μας.

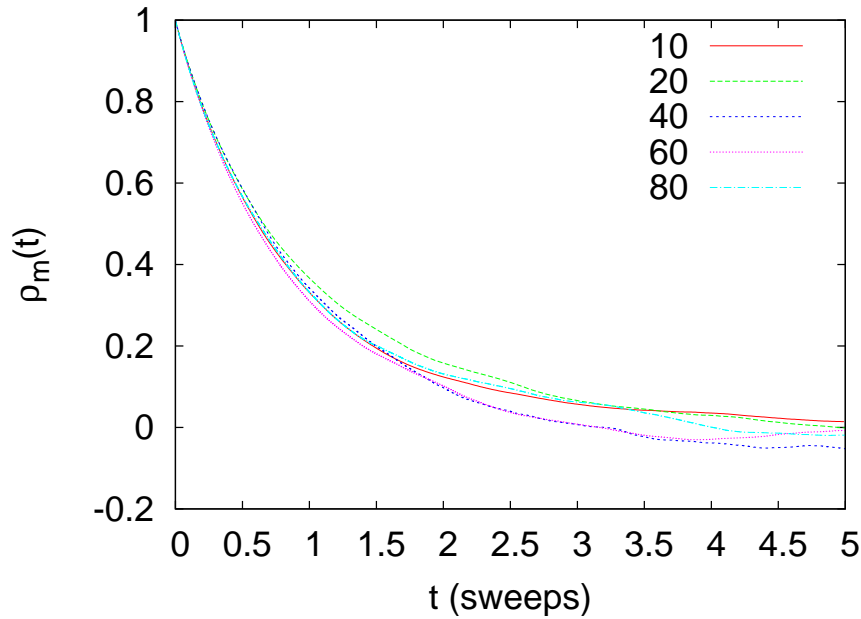
βάθμιση (scaling). Για τον αλγόριθμο του Metropolis αυτό σημαίνει ότι για μεγάλα  $L$  έχουμε

$$\tau \sim L^z, \quad (13.39)$$

όπου για την περίπτωση της μαγνήτισης έχουμε  $z = 2.1665 \pm 0.0012$  [59]. Αυτή η τιμή είναι μεγάλη και κάνει τον αλγόριθμο ακριβό. Σημαίνει πως ο χρόνος για την προσομοίωση του συστήματος για δεδομένο αριθμό ανεξάρτητων διατάξεων σπιν θα αυξάνει σαν

$$t_{\text{CPU}} \sim L^{d+z} \approx L^{4.17}. \quad (13.40)$$

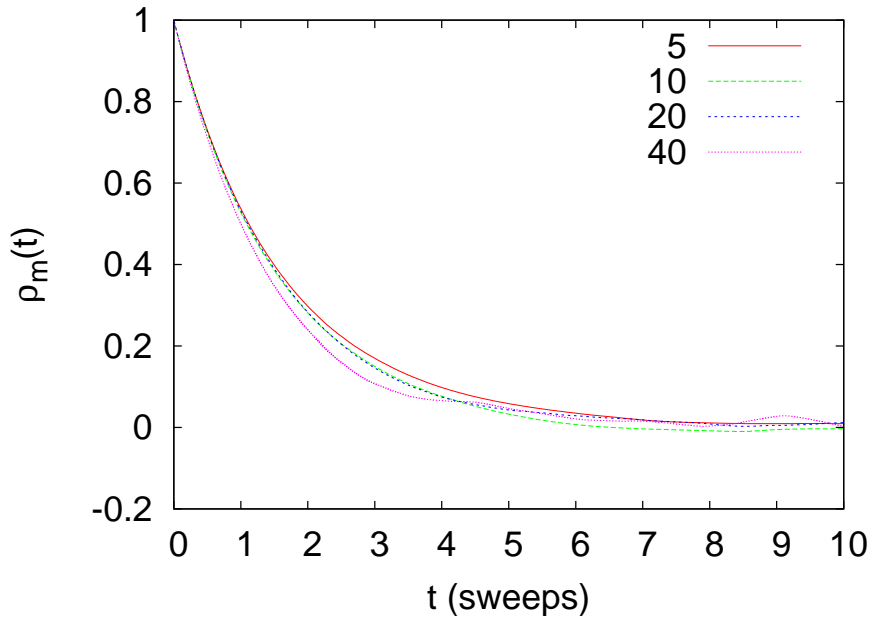
Αυτό είναι το φαινόμενο της κρίσιμης επιβράδυνσης. Σε επόμενα κεφάλαια θα συζητήσουμε με περισσότερη λεπτομέρεια τη σχέση βάθμισης (13.39), καθώς και καλύτερους αλγόριθμους που βελτιώνουν δραστικά τη συμπεριφορά αυτή.



Σχήμα 13.15: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης για το πρότυπο Ising σε υψηλή θερμοκρασία  $\beta = 0.20$  για  $L = 10, 20, 40, 60, 80$ . Ο χρόνος αυτοσυσχετισμού σε sweeps είναι ανεξάρτητος του  $L$ .

### 13.6 Στατιστικά Σφάλματα

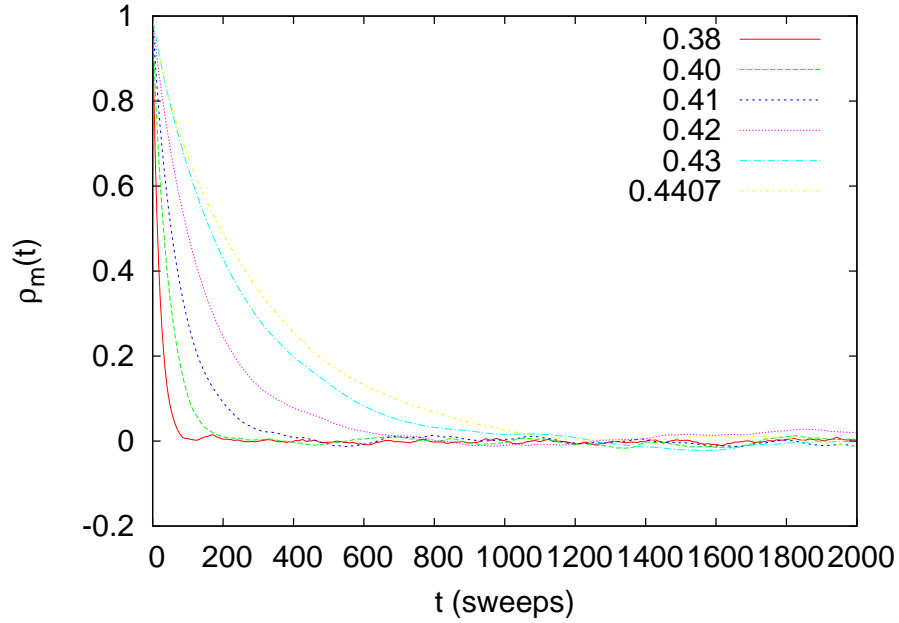
Σε μια τυχαία μεταβλητή, όπως είναι μία παρατηρήσιμη ποσότητα σε μία προσομοίωση Μόντε Κάρλο, η μέση τιμή της, εκτιμώμενη από ένα δείγμα, δε δίνει καμιά πληροφορία για την αξία της μέτρησης. Η γνώση της πραγματικής στατιστικής κατανομής δίνει όλη τη σχετική πληροφορία, αλλά στην πράξη, συνήθως, αρκούμαστε από τη γνώση του “σφάλματος” της μέτρησης. Για να ορίσουμε αυτό που θα ονομάζουμε στατιστικό σφάλμα στο βιβλίο αυτό, θα κάνουμε την πολύ απλή, αλλά πρακτική, υπόθεση ότι η κατανομή των μετρήσεων είναι Gaussian. Αυτό γενικά είναι μια πολύ καλή προσέγγιση, όταν έχουμε ανεξάρτητες μετρήσεις σε ένα πρότυπο στατιστικής φυσικής. Η αιτία του στατιστικού σφάλματος είναι οι θερμικές διακυμάνσεις γύρω από τη μέση τιμή που όπως έχουμε συζητήσει στην ενότητα 12.2 (βλ. και σχέση (12.27)) έχουν συνήθως κατά πολύ καλή προσέγγιση Gaussian κατανομή. Χαρακτηριστικό των στατιστικών σφαλμάτων είναι ότι εξαλείφονται αυξάνοντας τον αριθμό των μετρήσεων. Αυτό γίνεται σχετικά σιγά αφού μειώνονται αντιστρόφως ανάλογα με την τετραγωνική ρίζα του μεγέθους το δείγματος.



Σχήμα 13.16: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης για το πρότυπο Ising σε χαμηλή θερμοκρασία  $\beta = 0.65$  για  $L = 5, 10, 20, 40$ . Ο χρόνος αυτοσυσχετισμού σε sweeps είναι ανεξάρτητος του  $L$ .

Μια άλλη κατηγορία σφαλμάτων είναι τα συστηματικά σφάλματα τα οποία, συνήθως, δεν έχουν ... συστηματικό τρόπο να τα εκτιμήσουμε. Αυτά, στην περίπτωσή μας, μπορούν να έχουν απλή αιτία (λ.χ. να μην είναι το σύστημα σε θερμική ισορροπία) ή αρκετά δυσκολότερη να εντοπιστεί (λ.χ. μία κακή γεννήτρια ψευδοτυχαίων αριθμών).

Τέλος, στην περίπτωση που κάποιος χρησιμοποιεί ένα διακριτό και πεπερασμένο πρότυπο (όπως το Ising) για να προσομοιώσει ένα συνεχές και άπειρο μοντέλο, στους εκτιμητές υπεισέρχονται σφάλματα λόγω της διακριτοποίησης και του πεπερασμένου μεγέθους του συστήματος. Αυτά τα σφάλματα μειώνονται με την προσομοίωση μεγαλύτερων συστημάτων και αντιμετωπίζονται συστηματικά με μεθόδους όπως η βάθμιση πεπερασμένου μεγέθους (finite size scaling). Χρησιμοποιώντας βάθμιση πεπερασμένου μεγέθους, μπορούμε από τις μετρήσεις που παίρνουμε για πεπερασμένο μέγεθος συστήματος να εκτιμήσουμε την τιμή της μετρήσιμης ποσότητας για το απείρως μεγάλο σύστημα. Αυτό θα είναι αντικείμενο της μελέτης μας σε επόμενο κεφάλαιο.



Σχήμα 13.17: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης για το πρότυπο Ising για  $L = 40$ . Φαίνεται πώς αυξάνει ο χρόνος αυτοσυσχετισμού όταν πλησιάζουμε την κρίσιμη θερμοκρασία από τη θερμή φάση (φάση αταξίας).

### 13.6.1 Σφάλματα Ανεξάρτητων Μετρήσεων

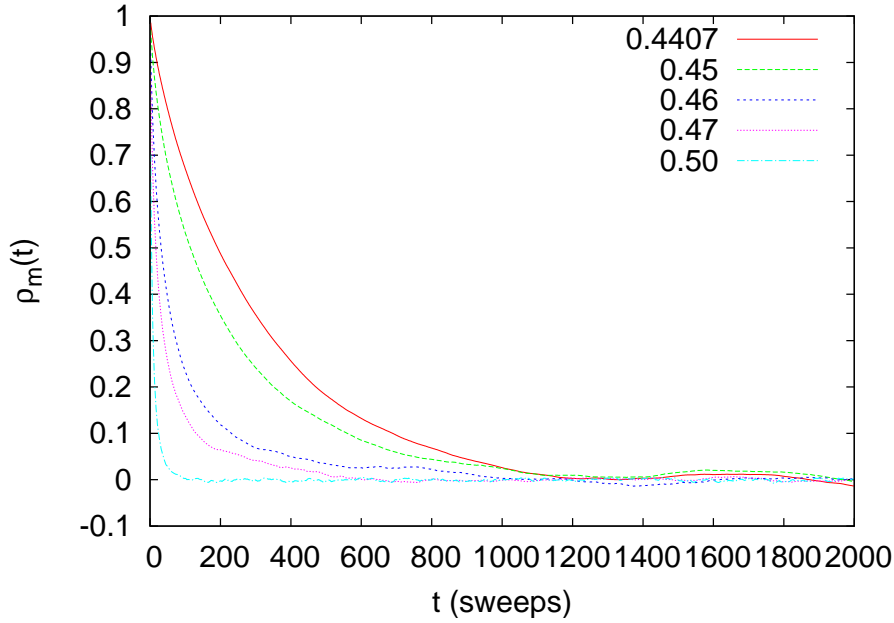
Από την υπόθεση ότι η πηγή των σφαλμάτων στις μετρήσεις μας είναι οι θερμικές διακυμάνσεις προκύπτει ότι η πραγματική τιμή μπορεί να εκτιμηθεί από τη μέση τιμή του δείγματος και ότι το σφάλμα θα είναι το σφάλμα της μέσης τιμής. Έτσι, αν έχουμε ένα δείγμα από  $n$  μετρήσεις  $\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{n-1}$ , η μέση τιμή τους είναι ένας εκτιμητής της  $\langle \mathcal{O} \rangle$  και παίρνουμε

$$\langle \mathcal{O} \rangle = \frac{1}{n} \sum_{i=0}^{n-1} \mathcal{O}_i. \quad (13.41)$$

Ενώ το σφάλμα της μέσης τιμής του δείγματος θα είναι εκτιμητής του σφάλματος  $\delta \mathcal{O}$ , οπότε έχουμε

$$(\delta \mathcal{O})^2 \equiv \sigma_{\mathcal{O}}^2 = \frac{1}{n-1} \left\{ \frac{1}{n} \sum_{i=0}^{n-1} (\mathcal{O}_i - \langle \mathcal{O} \rangle)^2 \right\} = \frac{1}{n-1} (\langle \mathcal{O}^2 \rangle - \langle \mathcal{O} \rangle^2). \quad (13.42)$$

Οι παραπάνω σχέσεις υποθέτουν πως το δείγμα μας αποτελείται από ανεξάρτητες μετρήσεις. Αυτό όπως είδαμε δε συμβαίνει σε μια προ-



Σχήμα 13.18: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης για το πρότυπο Ising για  $L = 40$ . Φαίνεται πώς αυξάνει ο χρόνος αυτοσυσχετισμού όταν πλησιάζουμε την κρίσιμη θερμοκρασία από την ψυχρή φάση (φάση τάξης).

σομοίωση Μόντε Κάρλο λόγω του αυτοσυσχετισμού των μετρήσεων. Είδαμε πως, αν ο χρόνος αυτοσυσχετισμού, μετρημένος σε μονάδες “αριθμός μετρήσεων”, είναι  $\tau_O$ , σύμφωνα με την 13.36 θα έχουμε  $n_O = n/(2\tau_O)$  ανεξάρτητες μετρήσεις. Στην περίπτωση αυτή, μπορεί ναδειχθεί ότι το πραγματικό στατιστικό σφάλμα στη μέτρηση της  $O$  είναι [60] (δείτε επίσης Κεφ. 4.1 του [5])

$$(\delta O)^2 = \frac{1 + 2\tau_O}{n - 1} (\langle O^2 \rangle - \langle O \rangle^2). \quad (13.43)$$

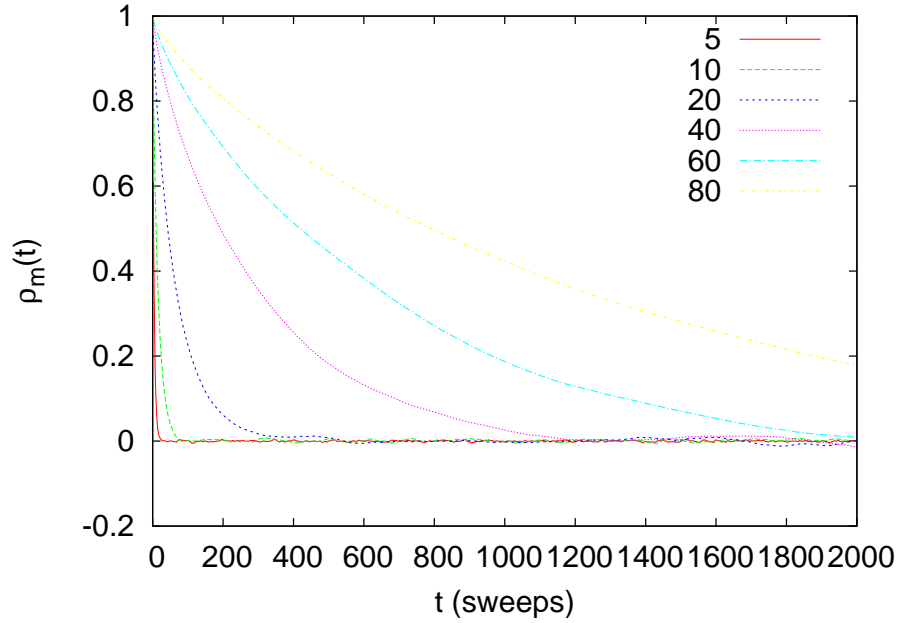
Αν  $\tau_O \ll 1$ , τότε παίρνουμε την (13.42). Αν  $\tau_O \gg 1$ , τότε παίρνουμε

$$(\delta O)^2 \approx \frac{2\tau_O}{n - 1} (\langle O^2 \rangle - \langle O \rangle^2) \quad (13.44)$$

$$\approx \frac{1}{(n/2\tau_O)} (\langle O^2 \rangle - \langle O \rangle^2) \quad (13.45)$$

$$\approx \frac{1}{n_O - 1} (\langle O^2 \rangle - \langle O \rangle^2) \quad (13.46)$$

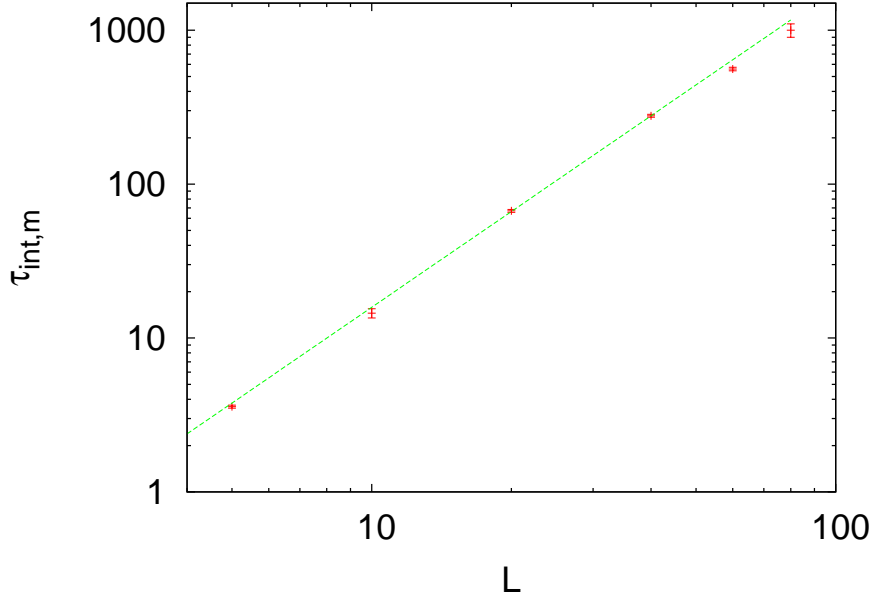
που δεν είναι παρά η (13.42) για  $n_O$  ανεξάρτητες μετρήσεις (υποθέσαμε ότι  $1 \ll n_O \ll n$ ). Η παραπάνω σχέση δικαιολογεί το γεγονός ότι θεωρήσαμε τις μετρήσεις πρακτικά ανεξάρτητες μετά από χρόνο  $\sim 2\tau_O$ .



Σχήμα 13.19: Η συνάρτηση αυτοσυσχετισμού της μαγνήτισης για το πρότυπο Ising για  $\beta = 0.4407 \approx \beta_c$  για διαφορετικά  $L$ . Παρατηρούμε τη μεγάλη αύξηση του χρόνου αυτοσυσχετισμού με το μέγεθος του συστήματος στην κρίσιμη περιοχή.

Οι παραπάνω σχέσεις μας δίνουν την ελευθερία να επιλέξουμε όπως μας βολεύει το πόσο συχνά θα παίρνουμε μετρήσεις πάνω στο σύστημα. Αλλά έχουν μερικές δυσκολίες στην εφαρμογή τους. Η πρώτη είναι πως πρέπει να κάνουμε τη μέτρηση του χρόνου αυτοσυσχετισμού με τις γνωστές δυσκολίες που περιγράψαμε στην ενότητα 13.5. Η δεύτερη είναι ότι δεν ξέρουμε τα αποτελέσματα που θα έχει στον υπολογισμό του σφάλματος η πραγματική κατανομή των μετρήσεων, ιδιαίτερα αν η ποσότητα που μετράμε δεν είναι μια απλή τοπική ποσότητα όπως λ.χ. η ενέργεια. Ας πάρουμε για παράδειγμα τη μαγνητική επιδεκτικότητα (13.33). Αυτό προϋποθέτει τη μέτρηση των  $\langle m \rangle$  και  $\langle m^2 \rangle$  [ή ακόμα καλύτερα την ποσότητα  $(m_i - \langle m \rangle)$  πάνω σε κάθε διάταξη  $i$ ] και δεν μπορεί να οριστεί πάνω σε μία μόνο διάταξη των σπιν. Αν υπολογίσουμε τις παραπάνω ποσότητες, αφού συγκεντρώσουμε το δείγμα μπαίνει το ερώτημα πώς θα υπολογιστεί το σφάλμα  $\delta\chi$ . Αυτό δεν είναι απλή συνάρτηση των  $\delta\langle m \rangle$  και  $\delta\langle m^2 \rangle$  λόγω του συσχετισμού των δύο ποσοτήτων και δεν ισχύει ο γνωστός τύπος διάδοσης των σφαλμάτων  $(\delta(\langle m^2 \rangle - \langle m \rangle^2))^2 = (\delta\langle m^2 \rangle)^2 + (\delta\langle m \rangle^2)^2$ .





Σχήμα 13.20: Ο ολοκληρωμένος χρόνος αυτοσυσχετισμού  $\tau_{\text{int},m}$  για  $\beta = \beta_c$  σε διάγραμμα λογαριθμικής κλίμακας. Η συνεχής καμπύλη είναι προσαρμογή στη συνάρτηση  $0.136(10)L^{2.067(21)}$ . Η αναμενόμενη τιμή του εκθέτη είναι  $2.1665(12)$  και η διαφορά οφείλεται στα σχετικά μικρά μεγέθη του συστήματος.

### 13.6.2 Jackknife

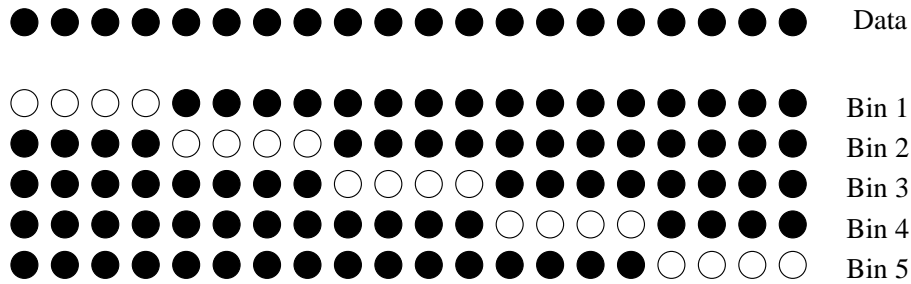
Η πιο απλή λύση που βρίσκει κανείς στα προβλήματα που αναφέραμε στην προηγούμενη υποενότητα είναι να χωρίσει κανείς το δείγμα του σε blocks ή bins. Το όνομα προέρχεται από την εικόνα ότι τις  $n$  μετρήσεις μας τις βάζουμε σε  $n_b$  “καλάθια” και κάθε καλάθι το χρησιμοποιούμε ως μία ανεξάρτητη μέτρηση. Αυτό θα είναι σωστό, αν ο αριθμός των μετρήσεων σε κάθε καλάθι είναι  $b = (n/n_b) \gg \tau_{\mathcal{O}}$ . Αν  $\mathcal{O}_i^b$   $i = 0, \dots, n_b - 1$  είναι οι μέσες τιμές της  $\mathcal{O}$  στο καλάθι  $i$ , τότε το σφάλμα θα δίνεται από την (13.42)

$$(\delta\mathcal{O})^2 = \frac{1}{n_b - 1} \left\{ \frac{1}{n_b} \sum_{i=0}^{n_b-1} (\mathcal{O}_i^b - \langle \mathcal{O}^b \rangle)^2 \right\} \quad (13.47)$$

Αυτή είναι η μέθοδος binning ή blocking και είναι απλή στη χρήση της. Παρατηρήστε ότι ποσότητες όπως η μαγνητική επιδεκτικότητα υπολογίζονται σε κάθε καλάθι σαν να ήταν ένα ανεξάρτητο δείγμα και το σφάλμα είναι τότε εύκολο να υπολογιστεί από την (13.47). Όπως είπαμε, για να είναι δυνατή η μέθοδος αυτή θα πρέπει κάθε καλάθι

να είναι στατιστικά ανεξάρτητο από το άλλο. Αν δεν είναι, τότε το σφάλμα θα είναι υποεκτιμημένο κατά τον παράγοντα  $2\tau_{\mathcal{O}}/(n_b - 1)$  της σχέσης (13.43). Τα καλάθια είναι στην πράξη στατιστικά ανεξάρτητα όταν  $b \sim 2\tau_{\mathcal{O}}$ . Αν δεν γνωρίζουμε τον  $\tau_{\mathcal{O}}$ , υπολογίζουμε επανειλημμένα το σφάλμα (13.47) μειώνοντας διαδοχικά τον αριθμό των καλάθιων  $n_b$ . Όταν το σφάλμα παύει να αυξάνεται και πάρει σταθερή τιμή, τότε θεωρούμε πως η μέθοδος συγκλίνει στη σωστή τιμή.

Η μέθοδος όμως που θα επιλέξουμε στις προσομοιώσεις μας είναι η μέθοδος του ... σουγιά (jackknife method). Είναι πιο σταθερή και δίνει πιο αξιόπιστα αποτελέσματα. Η βασική ιδέα είναι ίδια με τη μέθοδο binning με τη μόνη διαφορά ότι τα “καλάθια” φτιάχνονται με διαφορετικό τρόπο και η σχέση (13.47) ελαφρά τροποποιείται. Χωρίζουμε τα δεδομένα μας σε  $n_b$  bins τα οποία περιέχουν από  $b = n - (n/n_b)$  στοιχεία ως εξής: Το bin  $j$  προκύπτει από τα αρχικά δεδομένα  $\mathcal{O}_0, \dots, \mathcal{O}_{n-1}$ , αν διαγράψουμε τα περιεχόμενα του  $j$ -καλαθιού της μεθόδου binning. Η διαδικασία φαίνεται σχηματικά στο σχήμα 13.21. Σε κάθε bin υπολογίζουμε



Σχήμα 13.21: Η μέθοδος jackknife για ένα δείγμα από  $n = 20$  μετρήσεις. Χωρίζουμε τα δεδομένα σε  $n_b = 5$  bins το οποίο το καθένα έχει  $b = n - (n/n_b) = 20 - 4 = 16$  μετρήσεις (οι γεμάτοι δίσκοι). Σε κάθε δείγμα υπολογίζουμε τη μέση τιμή  $\mathcal{O}_i^b$  και από αυτές το σφάλμα  $\delta\mathcal{O} = \sqrt{n_b (\langle (\mathcal{O}^b)^2 \rangle - \langle \mathcal{O}^b \rangle^2)}$ .

τη μέση τιμή της  $\mathcal{O}$ . Προκύπτουν τότε οι μέσες τιμές  $\mathcal{O}_0^b, \mathcal{O}_1^b, \dots, \mathcal{O}_{n_b-1}^b$ . Τότε το σφάλμα στην μέτρηση της  $\mathcal{O}$  είναι

$$(\delta\mathcal{O})^2 = \sum_{j=0}^{n_b-1} (\mathcal{O}_j^b - \langle \mathcal{O}^b \rangle)^2 = n_b (\langle (\mathcal{O}^b)^2 \rangle - \langle \mathcal{O}^b \rangle^2). \quad (13.48)$$

Όπως και στη μέθοδο binning, ο αριθμός των bins καθορίζεται μεταβάλλοντάς τον και προσδιορίζοντας εκείνες τις τιμές για τις οποίες η τιμή του σφάλματος σταθεροποιείται.

Για περισσότερες λεπτομέρειες και αποδείξεις των ισχυρισμών στην υποενότητα αυτή, παραπέμπουμε τον αναγνώστη στο βιβλίο του Berg

[5]. Στο παράρτημα 13.8.1 δίνουμε πρόγραμμα για τη χρήση της μεθόδου και σχετικά παραδείγματα.

Η μέθοδος jackknife είναι η μέθοδος υπολογισμού των σφαλμάτων που προτιμάται από την πλειοψηφία της επιστημονικής κοινότητας και αυτή θα ακολουθήσουμε στο βιβλίο αυτό.

### 13.6.3 Bootstrap

Μια άλλη μέθοδος, χρήσιμη στην εκτίμηση των σφαλμάτων, είναι η μέθοδος bootstrap. Έστω ότι έχουμε  $n$  ανεξάρτητες μετρήσεις. Από αυτές δημιουργούμε  $n_S$  τυχαία δείγματα: Με ομοιόμορφη πιθανότητα διαλέγουμε μία μέτρηση από τις  $n$  μετρήσεις. Αυτό το κάνουμε  $n$  φορές από το ίδιο σύνολο των  $n$  μετρήσεων - δηλ. χωρίς να αφαιρούμε τη μέτρηση που διαλέξαμε από αυτές που θα διαλέξουμε την επόμενη φορά και έτσι φτιάχνουμε ένα δείγμα. Αυτό σημαίνει ότι το δείγμα θα περιέχει  $\sim 1 - 1/e \approx 63\%$  ίδιες μετρήσεις. Σε κάθε δείγμα  $i = 0, \dots, n_S - 1$  υπολογίζουμε την ποσότητα  $\mathcal{O}_i^S$  και από αυτές τη μέση τιμή

$$\langle \mathcal{O}^S \rangle = \frac{1}{n_S} \sum_{i=0}^{n_S-1} \mathcal{O}_i^S, \quad (13.49)$$

και την

$$\langle (\mathcal{O}^S)^2 \rangle = \frac{1}{n_S} \sum_{i=0}^{n_S-1} (\mathcal{O}_i^S)^2. \quad (13.50)$$

Η εκτίμηση για το σφάλμα στη μέτρηση του  $\langle \mathcal{O} \rangle$  είναι τότε<sup>20</sup>

$$(\delta \mathcal{O})^2 = \langle (\mathcal{O}^S)^2 \rangle - \langle \mathcal{O}^S \rangle^2. \quad (13.51)$$

Πρέπει να τονίσουμε ότι ο παραπάνω τύπος δίνει το σφάλμα για ανεξάρτητες μετρήσεις. Αν έχουμε μη αμελητέους χρόνους αυτοσυσχετισμού, πρέπει να χρησιμοποιήσουμε τη διόρθωση που δίνει η σχέση

$$(\delta \mathcal{O})^2 = (1 + 2\tau_{\mathcal{O}}) \left( \langle (\mathcal{O}^S)^2 \rangle - \langle \mathcal{O}^S \rangle^2 \right) \quad (13.52)$$

Στο παράρτημα 13.8.2 δείχνουμε πώς να χρησιμοποιήσουμε τη μέθοδο bootstrap για να υπολογίσουμε το πραγματικό σφάλμα  $\delta \mathcal{O}$ , χωρίς a priori γνώση του  $\tau_{\mathcal{O}}$ .

Για περισσότερες λεπτομέρειες παραπέμπουμε τον αναγνώστη στα άρθρα του Bradley Efron [61]. Στο παράρτημα 13.8.2 δίνουμε πρόγραμμα για τη χρήση της μεθόδου και σχετικά παραδείγματα.

<sup>20</sup>Παρατηρήστε ότι το δεξί μέλος της (13.51) δεν διαιρείται με  $1/(n_S - 1)$ .

### 13.7 Παράρτημα: Συνάρτηση Αυτοσυσχετισμού

Στο παράρτημα αυτό θα δείξουμε τις τεχνικές λεπτομέρειες του υπολογισμού της συνάρτησης αυτοσυσχετισμού (13.34) και του χρόνου αυτοσυσχετισμού (13.35) και (13.37). Τα προγράμματα σε αυτό και το επόμενο παράρτημα θα τα βρείτε στον κατάλογο Tools στο συνοδευτικό λογισμικό.

Σε ένα δείγμα από  $n$  μετρήσεις  $\mathcal{O}(0), \mathcal{O}(1), \dots, \mathcal{O}(n-1)$  θα χρησιμοποιήσουμε ως εκτιμητή της σχέσης (13.34) την

$$\rho_{\mathcal{O}}(t) = \frac{1}{\rho_0} \frac{1}{n-t} \sum_{t'=0}^{n-1-t} (\mathcal{O}(t') - \langle \mathcal{O} \rangle_0)(\mathcal{O}(t'+t) - \langle \mathcal{O} \rangle_t) \quad (13.53)$$

όπου οι μέσες τιμές ορίζονται από τις σχέσεις<sup>21</sup>

$$\langle \mathcal{O} \rangle_0 \equiv \frac{1}{n-t} \sum_{t'=0}^{n-1-t} \mathcal{O}(t') \quad \langle \mathcal{O} \rangle_t \equiv \frac{1}{n-t} \sum_{t'=0}^{n-1-t} \mathcal{O}(t'+t). \quad (13.54)$$

Η σταθερά  $\rho_0$  επιλέγεται έτσι, ώστε  $\rho_{\mathcal{O}}(0) = 1$ .

Το πρόγραμμα για τον υπολογισμό της συνάρτησης (13.34) και του χρόνου (13.37) είναι απλό να γραφτεί. Παραθέτουμε το αρχείο autoc.f90 από το συνοδευτικό λογισμικό. Διαβάστε τα ένθετα σχόλια για επεξηγήσεις των σημαντικών βημάτων.

```
!=====
!file: autoc.f90
MODULE rho_function
  implicit none
  SAVE
  integer :: NMAX, tmax
  character(200) :: prog
  CONTAINS
!-----
!rho is the unnormalized autocorrelation function at t:
  real(8) function rho(x, ndat, t)
    implicit none
    integer :: ndat, t
    real(8), dimension(0:) :: x
```

<sup>21</sup>Θα μπορούσαμε να πάρουμε  $\langle \mathcal{O} \rangle_0 = \langle \mathcal{O} \rangle_t = (1/n) \sum_{t'=0}^n \mathcal{O}(t')$ , χωρίς να έχουμε μεγάλη διαφορά στα αποτελέσματα, όταν έχουμε  $t \ll n$ . Με τον ορισμό που χρησιμοποιούμε έχουμε ελαφρά μικρότερα φαινόμενα πεπερασμένου μεγέθους του δείγματος. Επίσης, η επιλογή μας (13.53) αντί για την  $\rho_{\mathcal{O}}(t) \propto (1/(n-t)) \sum_{t'=0}^{n-1-t} \mathcal{O}(t')\mathcal{O}(t'+t) - \langle \mathcal{O} \rangle_0 \langle \mathcal{O} \rangle_t$  μειώνει τα σφάλματα στρογγυλοποίησης.

```

integer                :: n,t0
real(8)                :: xav0,xavt,r
!
n=ndat-t
if(n<1) call locerr('rho: n<1')
!Calculate the two averages: xav0=<x>_0, xavt=<x>_t
xav0 = SUM( x(0:n-1) ) / n
xavt = SUM( x(t:n-1+t) ) / n
rho  = SUM((x(0:n-1)-xav0)*(x(t:n-1+t)-xavt))/n
end function rho
!
subroutine locerr(errmes)
implicit none
character(*) :: errmes
write(0,'(A,A)'),TRIM(prog),': ',TRIM(errmes),' Exiting....'
stop 1
end subroutine locerr
END MODULE rho_function
!=====
program autocorrelations
USE rho_function
implicit none
real(8),allocatable,dimension(:) :: r,tau,x
real(8) :: norm
integer :: i,ndat,t,tcut,chk
!
!Default values for max number of data and max time for
!rho and tau:
NMAX=2000000;tmax=1000 !NMAX=2e6 requires ~ 2e6*8=16MB
call get_the_options
ALLOCATE(x(0:NMAX-1),STAT=chk)
if(chk > 0) call locerr('Not enough memory for x')
ndat=0
do while ( ndat < NMAX)
  read(*,*,END=101)x(ndat)
  ndat = ndat+1
enddo !
101 continue
if(ndat >= NMAX) write(0,'(3A,I14,A,I14)') &
  '# ',TRIM(prog), &
  ': Warning: read ndat=', ndat, &
  ' and reached the limit: ',NMAX
!We decrease tmax if it is comparable or large of ndat
if(tmax > (ndat/10) ) tmax = ndat/10
!r(t) stores the values of the autocorrelation function rho(t)
ALLOCATE(r(0:tmax-1))
do t=0,tmax-1
  r(t) = rho(x,ndat,t)
enddo

```

```

norm = 1.0D0/r(0); r = norm*r
!tau(t) stores integrated autocorrelation times with tcut=t
ALLOCATE(tau(0:tmax-1))
do tcut=0,tmax-1
  tau(tcut)=0.0D0
  do t=0,tcut
    tau(tcut) = tau(tcut)+r(t)
  enddo
enddo
!Output:
print '(A)', '# =====',
print '(A)', '# Autoc function rho and int autoc time tau ',
print '(A,I12,A,I8)', '# ndat= ', ndat, ',   tmax= ', tmax
print '(A)', '# t           rho(t)           tau(tcut=t) ',
print '(A)', '# =====',
do t=0,tmax-1
  print '(I8,2G28.17)', t, r(t), tau(t)
enddo
end program autocorrelations
!=====
subroutine get_the_options
  use rho_function
  use getopt_m      !from getopt.f90
  implicit none
  call getarg(0,prog)

  do
    select case( getopt( "--ht:n:" ))
    case( 't' )
      read(optarg,*)tmax
    case( 'n' )
      read(optarg,*)NMAX
    case( 'h' )
      call usage
    case( '?' )
      print *, 'unknown option ', optopt
      stop
    case( char(0)) ! done with options
      exit
    case( '-' )   ! use -- to exit from options
      exit
    case default
      print *, 'unhandled option ', optopt
    end select
  enddo

end subroutine get_the_options
!=====
subroutine usage

```

```

use rho_function
implicit none
print '(3A)', 'Usage:', TRIM(prog), ' [-t <maxt>] [-n <ndat>]',
print '( A)', '      Reads data from stdin (one column) and',
print '( A)', '      computes autocorrelation function and',
print '( A)', '      integrated autocorrelation time.'
stop
end subroutine usage
!=====

```

Να σημειώσουμε ότι τον υπολογισμό της συνάρτησης αυτοσυσχετισμού την “πακετάραμε” σε ένα module `rho_function` και μπορείτε να τη χρησιμοποιήσετε σε οποιοδήποτε πρόγραμμα το οποίο ξεκινάει με την εντολή `use rho_function`. Αυτό γίνεται στο module μετά τη λέξη-κλειδί `CONTAINS`. Μετά, μπορούμε να προσθέσουμε κώδικα για υπορουτίνες και συναρτήσεις στις οποίες έχει πρόσβαση<sup>22</sup> οποιοδήποτε πρόγραμμα χρησιμοποιεί το module. Φυσικά, όπως και σε άλλα προγράμματα, χρησιμοποιούμε το module για να ορίσουμε τις μεταβλητές `NMAX`, `tmax` και `prog`, έτσι ώστε να είναι προσβάσιμες από όλα τα μέρη του προγράμματος που χρησιμοποιούν το module.

Για τον μεταγλωττισμό του προγράμματος χρησιμοποιούμε την εντολή

```
> gfortran -O2 getopt.f90 autoc.f90 -o autoc
```

Αν έχουμε τα δεδομένα μας σε μία στήλη στο αρχείο `data` υπολογίζουμε τη συνάρτηση αυτοσυσχετισμού και τον ολοκληρωμένο χρόνο αυτοσυσχετισμού με την εντολή

```
> cat data | ./autoc > data.rho
```

όπου τα αποτελέσματα στο αρχείο `data.rho` είναι σε τρεις στήλες. Η πρώτη είναι ο χρόνος  $t$ , η δεύτερη η  $\rho(t)$  και η τρίτη η  $\tau_{\text{int},O}(t)$  [σχέση (13.38)]. Τα αντίστοιχα σχήματα φτιάχνονται με τις εντολές `gnuplot`:

```

gnuplot> plot "data.rho" using 1:2 with lines
gnuplot> plot "data.rho" using 1:3 with lines

```

Αν θέλουμε να αλλάξουμε τον μέγιστο αριθμό δεδομένων `NMAX` (λ.χ. αν έχουμε περισσότερα δεδομένα ή αν θέλουμε να μειώσουμε τη μνήμη

<sup>22</sup>Στην πραγματικότητα είναι γνωστό το `explicit interface` της συνάρτησης ή υπορουτίνας σε όλα τα μέρη του προγράμματος που χρησιμοποιούν το module.

που δεσμεύει το πρόγραμμα) ή το μέγιστο χρόνο `tmax` στη συνάρτηση αυτοσυσχετισμού, χρησιμοποιούμε τα options `-n` και `-t` αντίστοιχα:

```
> cat data | autoc -n 20000000 -t 20000 > data.rho
```

Για την απευθείας εισαγωγή των δεδομένων δοκιμάστε την εντολή:

```
gnuplot> plot "<./is -L 20 -b 0.4407 -s 1 -S 345 -n 400000|\n\ngrep -v '#lawk' '{print ($2>0)?$2:-$2;}' |\\n\\nautoc -t 500" using 1:2 with lines
```

Η παραπάνω gnuplot εντολή είναι σπασμένη σε 3 γραμμές για να φαίνεται στη σελίδα. Για να τη χρησιμοποιήσετε, ενώστε τις γραμμές και αφαιρέστε τις `\\n\\n`.

Στη συνέχεια, παραθέτουμε το script `autoc_L` το οποίο κάνει τους απαραίτητους υπολογισμούς για το σχήμα 13.19.

```
#!/bin/tcsh -f

set nmeas = 2100000
set Ls = (5 10 20 40 60 80)
set beta = 0.4407
set tmax = 2000
foreach L ($Ls)
    set N = 'awk -v L=$L 'BEGIN{ print L*L }''
    set rand = 'perl -e 'srand();print int(3000000*rand()+1);''
    set out = outL${L}b${beta}
    echo "Running L${L}b${beta}"
    ./is -L $L -b $beta -s 1 -S $rand -n $nmeas > $out
    echo "Autocorrelations L${L}b${beta}"
    grep -v '#' $out | \\
    awk -v N=$N 'NR>100000{ print ($2>0)?($2/N):(-$2/N) }' | \\
    autoc -t $tmax > $out.rhom
end
```

Στη συνέχεια, κατασκευάζουμε το σχήμα με το gnuplot:

```
gnuplot> plot "outL5b0.4407.rhom" u 1:2 w lines t "5"
gnuplot> replot "outL10b0.4407.rhom" u 1:2 w lines t "10"
gnuplot> replot "outL20b0.4407.rhom" u 1:2 w lines t "20"
gnuplot> replot "outL40b0.4407.rhom" u 1:2 w lines t "40"
gnuplot> replot "outL60b0.4407.rhom" u 1:2 w lines t "60"
gnuplot> replot "outL80b0.4407.rhom" u 1:2 w lines t "80"
```

Με παρόμοιο τρόπο φτιάχνουμε και τα σχήματα 13.17.



Για τον υπολογισμό του  $\tau_m$  εργαζόμαστε ως εξής:

```
gnuplot> f(x) = c * exp(-x/t)
gnuplot> set log y
gnuplot> plot [:1000] "outL40b0.4407.rhom" u 1:2 w lines
gnuplot> c = 1 ; t = 300
gnuplot> fit [150:650] f(x) "outL40b0.4407.rhom" u 1:2 via c,t
gnuplot> plot [:1000] "outL40b0.4407.rhom" u 1:2 w lines,f(x)
gnuplot> plot [:] "outL40b0.4407.rhom" u 1:3 w lines
```

όπου με την τελευταία γραμμή συγκρίνουμε με την τιμή του  $\tau_{\text{int},m}$ . Η εντολή `fit` είναι ενδεικτική. Πρέπει να δοκιμαστούν διάφορες τιμές για τα όρια τα οποία εδώ επιλέχτηκαν να είναι τα [150:650]. Αφού με την πρώτη εντολή `plot` δούμε τότε η συνάρτηση έχει την αναμενόμενη ασυμπτωτική συμπεριφορά (τη συνάρτηση  $f(x) = c \exp(-x/t)$ ), μεταβάλλουμε τα άνω και κάτω όρια, έτσι ώστε η τιμή για τον  $\tau_m$  να σταθεροποιηθεί ενώ το <sup>23</sup>  $\chi^2/\text{dof}$  της προσαρμογής να είναι ελάχιστο<sup>24</sup>. Το  $\chi^2/\text{dof}$  της προσαρμογής το διαβάζουμε από το αποτέλεσμα της εντολής `fit`

```
.....
degrees of freedom (FIT_NDF) : 449
rms of residuals (FIT_STDFIT)=sqrt(WSSR/ndf): 0.000939
variance of residuals(reduced chisquare)=WSSR/ndf: 8.82e-07

Final set of parameters          Asymptotic Standard Error
=====
c = 0.925371                    +/- 0.0003773 (0.04078%)
t = 285.736                     +/- 0.1141 (0.03995%)
.....
```

από τη γραμμή “variance of residuals”. Από τις επόμενες γραμμές διαβάζουμε τις τιμές των παραμέτρων με τα σφάλματά τους<sup>25</sup> και συμπεραίνουμε ότι  $\tau_m = 285.7 \pm 0.1$ .

<sup>23</sup> Αν έχουμε τα δεδομένα  $\{(x_i, y_i)\}$ ,  $i = 1, \dots, n$  με σφάλμα  $\delta y_i$  τα οποία θέλουμε να προσαρμόσουμε στην  $f(x; c, t) = c e^{-x/t}$ , τότε το  $\chi^2(c, t) = \sum_{i=1}^n (y_i - f(x_i; c, t))^2 / \delta y_i^2$ . Το  $\chi^2/\text{dof}$  είναι κανονικοποιημένο στον αριθμό των βαθμών ελευθερίας ( $\text{dof} = \text{degrees of freedom} = n - 2$ ) που είναι ο αριθμός των σημείων στην προσαρμογή  $n$  μείον τον αριθμό των ελεύθερων παραμέτρων της προσαρμογής (εδώ οι  $c, t$  είναι 2).

<sup>24</sup> Αποδεκτά  $\chi^2/\text{dof} \sim 1$ , αλλά επειδή εδώ δεν παρέχουμε τα σφάλματα στην τιμή της συνάρτησης αυτοσυσχετισμού, το  $\chi^2/\text{dof}$  δεν είναι σωστά κανονικοποιημένο. Το πρόγραμμα θέτει  $\delta y_i = 1 \forall i$ .

<sup>25</sup> Στις παρενθέσεις είναι το επίπεδο εμπιστοσύνης (confidence level). Αυτό ορίζεται να είναι η πιθανότητα οι τιμές των παραμέτρων να είναι μέσα στο διάστημα που ορίζει

Πρέπει να τονιστεί ότι το σφάλμα αυτό είναι το στατιστικό σφάλμα της προσαρμογής για τα δεδομένα όρια που επιλέξαμε και όχι το πραγματικό σφάλμα. Συνήθως, όπως και εδώ, τα μεγαλύτερα σφάλματα στην προσαρμογή των δεδομένων σε μία συνάρτηση είναι συστηματικά που οφείλονται (εδώ) κυρίως στην επιλογή των ορίων<sup>26</sup>. Δοκιμάζοντας διάφορες τιμές μέχρι να διπλασιαστεί η τιμή του  $\chi^2/\text{dof}$ , βρίσκουμε  $\tau_m = 285(2)$ .

Στη συγκεκριμένη περίπτωση, το μεγαλύτερο συστηματικό σφάλμα προέρχεται από την παράλειψη της επίδρασης και άλλων χρόνων αυτοσυσχετισμού του συστήματος. Λάβαμε υπόψη μόνο την κύρια συνεισφορά από τον μεγαλύτερο χρόνο αυτοσυσχετισμού προσαρμόζοντας τα δεδομένα στη συνάρτηση

$$f(t) = c e^{-t/\tau} . \quad (13.55)$$

Στην πραγματικότητα, λαμβάνοντας υπόψη και μικρότερους χρόνους αυτοσυσχετισμού, περιμένουμε  $\rho_m(t) \sim a_1 e^{-t/\tau_1} + a_2 e^{-t/\tau_2} + \dots$ . Βρίσκουμε πως τα δεδομένα μας προσαρμόζονται με πολύ σταθερό τρόπο στη συνάρτηση

$$h(x) = a_1 e^{-x/\tau_1} + a_2 e^{-x/\tau_2} + a_3 e^{-x/\tau_3} . \quad (13.56)$$

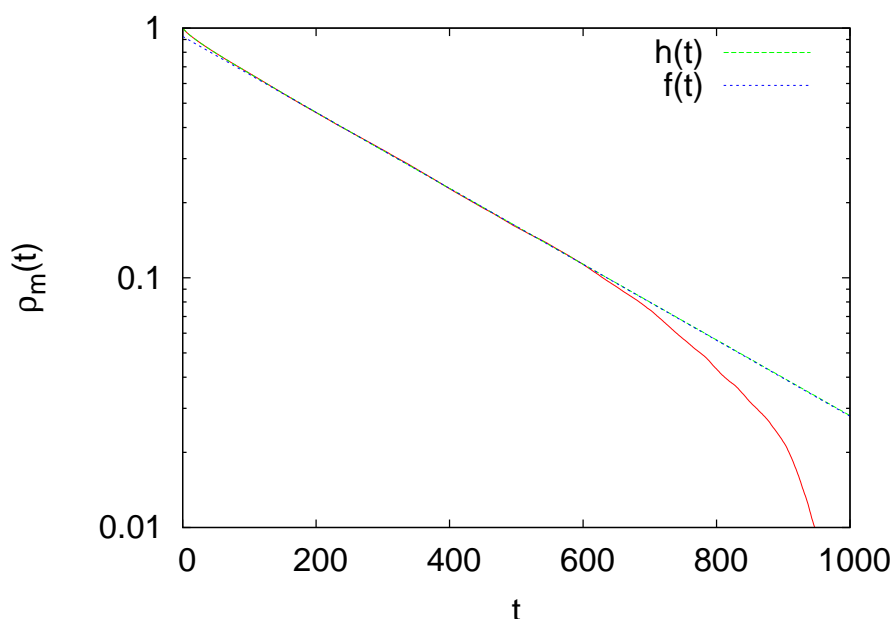
Όπως φαίνεται και στο σχήμα 13.22, πετυχαίνουμε άριστη προσαρμογή για πολύ μικρούς χρόνους και τα αποτελέσματά μας για τον κύριο χρόνο αυτοσυσχετισμού  $\tau_m \equiv \tau_1 = 286.3(3)$ . Οι δευτερεύοντες χρόνοι αυτοσυσχετισμού είναι  $\tau_2 = 57(3)$ ,  $\tau_3 = 10.5(8)$  οι οποίοι είναι αρκετά μικρότεροι από τον  $\tau_1$ .

Για διευκόλυνση του αναγνώστη παραθέτουμε τις βασικές εντολές για την αναφερόμενη ανάλυση:

```
gnuplot> h(x) = a1*exp(-x/t1) + a2*exp(-x/t2) + a3*exp(-x/t3)
gnuplot> a1 = 1; t1 = 285; a2 = 0.04; t2 = 56;
gnuplot> a3 = 0.03; t3 = 10
gnuplot> fit [1:600] h(x) "outL40b0.4407.rhom" \
      using 1:2 via a1,t1,a2,t2,a3,t3
```

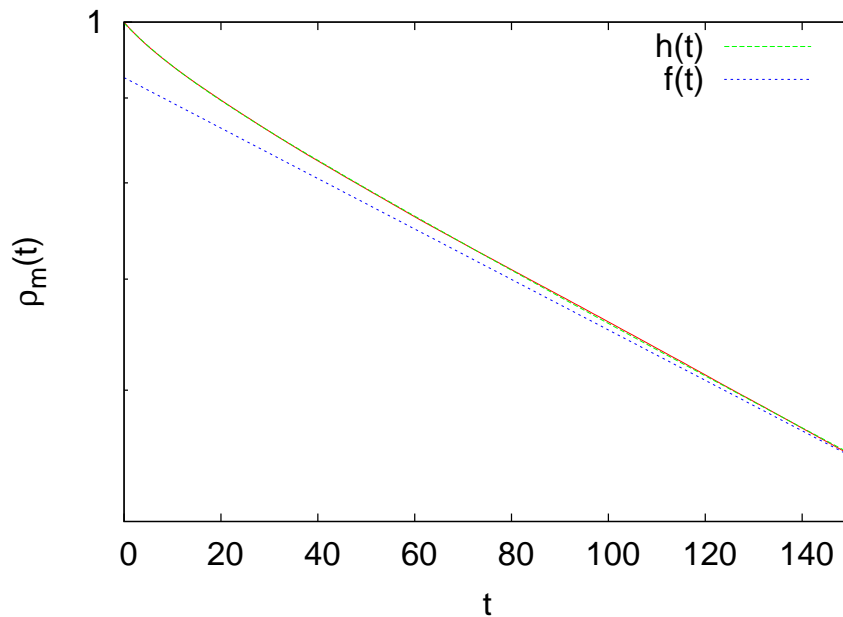
το σφάλμα. Αυτό υποθέτει ότι έχουμε το σωστό στατιστικό βάρος σε κάθε σημείο που δίνεται από το σφάλμα - στο παράδειγμα δεν το έχουμε γι' αυτό και το επίπεδο είναι τόσο χαμηλό. Μια τιμή κάτω από 5% είναι πολύ χαμηλή και υποδηλώνει ότι το μοντέλο έχει ανάγκη από διορθώσεις. Οι αριθμητικές τιμές υποθέτουν Gaussian κατανομή των μετρήσεων και αν η υπόθεση αυτή δεν ισχύει πρέπει να ληφθούν απλά σαν μια ποιοτική ένδειξη.

<sup>26</sup>Για ένα προσεκτικό υπολογισμό μιας ποσότητας που προκύπτει από προσαρμογή δεδομένων, δοκιμάζουμε και άλλες συναρτήσεις που μπορεί να περιέχουν διορθώσεις της ασυμπτωτικής συμπεριφοράς.



Σχήμα 13.22: Προσαρμογή της συνάρτησης αυτοσυσχετισμού  $\rho_m(t)$  στις συναρτήσεις  $f(t) = c e^{-t/\tau}$  και  $h(t) = a_1 e^{-t/\tau_1} + a_2 e^{-t/\tau_2} + a_3 e^{-t/\tau_3}$ . Για μεγάλους χρόνους  $f(t) \approx h(t)$ , αλλά η  $h(t)$  είναι αναγκαία για την προσαρμογή των μικρών χρόνων. Η επιλογή αυτή δίνει σταθερή τιμή για τον κύριο χρόνο αυτοσυσχετισμού  $\tau_m = \tau = \tau_1$ . Οι τιμές των παραμέτρων δίνονται στο κείμενο.

```
...
Final set of parameters
=====
a1          = 0.922111          +/- 0.001046      (0.1135%)
t1          = 286.325          +/- 0.2354          (0.08221%)
a2          = 0.0462523        +/- 0.001219        (2.635%)
t2          = 56.6783          +/- 2.824           (4.982%)
a3          = 0.0300761        +/- 0.001558        (5.18%)
t3          = 10.5227          +/- 0.8382           (7.965%)
gnuplot> plot [:150][0.5:] "outL40b0.4407.rhom" using 1:2 \
with lines notit,h(x) ,f(x)
gnuplot> plot [:1000][0.01:] "outL40b0.4407.rhom" using 1:2 \
with lines notit,h(x) ,f(x)
```



Σχήμα 13.23: Το σχήμα 13.22 σε μικρότερη κλίμακα χρόνου, όπου φαίνεται καθαρότερα η επίδραση των μικρότερων χρόνων αυτοσυσχετισμού.

## 13.8 Παράρτημα: Υπολογισμός Σφαλμάτων

### 13.8.1 Η Μέθοδος Jackknife

Εδώ δίνουμε το πρόγραμμα που υπολογίζει τα σφάλματα με τη μέθοδο jackknife που περιγράψαμε στην ενότητα 13.6.2. Στο σχήμα 13.21 φαίνεται πώς χωρίζουμε τα δεδομένα μας σε bins. Σε κάθε bin υπολογίζουμε τη μέση τιμή της ποσότητας  $\mathcal{O}$  που μας ενδιαφέρει και στη συνέχεια, από τη σχέση (13.48) τα σφάλματα. Για την εφαρμογή της μεθόδου, παραθέτουμε πρόγραμμα από το αρχείο `jack.f90` (θα το βρείτε στον υποκατάλογο `Tools/` στο συνοδευτικό λογισμικό) το οποίο υπολογίζει τις ποσότητες  $\langle \mathcal{O} \rangle$ ,  $\delta \mathcal{O}$ ,  $\chi \equiv \langle (\mathcal{O} - \langle \mathcal{O} \rangle)^2 \rangle$  και  $\delta \chi$ .

```
!=====
!file: jack.f90
MODULE jack_function
  implicit none
  SAVE
  integer :: JACK, MAXDAT
  character(200) :: prog
```

```

CONTAINS
!
!jackknife function:
subroutine jackknife(ndat,jack,x,&
    av0,er0,avchi,erchi)
    integer                :: ndat,jack !local jack...
    real(8),dimension(0:) :: x
    real(8)                :: av0,er0,avchi,erchi
    integer                :: i,j,binw,bin
    real(8),allocatable    :: 0(:),chi(:)
!
    ALLOCATE(0(0:jack-1));ALLOCATE(chi(0:jack-1))
    0=0.0D0;chi=0.0D0;
    binw=ndat/jack
    if(binw<1)call locerr('jackknife: binw < 1')
!Average value:
    do i=0,ndat-1
        do j=0,jack-1
            if((i/binw) /= j) &
                0 (j) = 0 (j) + x(i)
        enddo
    enddo
    0 = 0 /(ndat-binw) !normalize
!Susceptibility:
    do i=0,ndat-1
        do j=0,jack-1
            if((i/binw) /= j) &
                chi(j) = chi(j) + (x(i)-0(j))*(x(i)-0(j))
        enddo
    enddo
    chi = chi/(ndat-binw) !normalize
!
    av0 = SUM(0)/jack;avchi=SUM(chi)/jack
    er0 = sqrt(SUM((0 -av0 )*(0 -av0 )))
    erchi = sqrt(SUM((chi-avchi)*(chi-avchi)))
!
    DEALLOCATE(0);DEALLOCATE(chi)
end subroutine jackknife
!
subroutine locerr(errmes)
    implicit none
    character(*) :: errmes
    write(0,'(A,A) '),TRIM(prog),': ',TRIM(errmes),' Exiting .... '
    stop 1
end subroutine locerr
END MODULE jack_function
=====
program jackknife_errors
    use jack_function

```

```

implicit none
integer                :: ndat,chk
real(8)                :: 0,d0,chi,dchi
real(8),allocatable :: x(:)
MAXDAT=1000000;JACK=10
call get_the_options
ALLOCATE(x(0:MAXDAT-1),STAT=chk)
if(chk > 0) call locerr('Not enough memory for x')
ndat=0
do while ( ndat < MAXDAT)
  read(*,*,END=101)x(ndat)
  ndat = ndat+1
enddo
101 continue
if(ndat >= MAXDAT) write(0,'(3A,I14,A,I14)') &
  '# ',TRIM(prog), &
  ': Warning: read ndat=', ndat, &
  ' and reached the limit: ',MAXDAT
call jackknife(ndat,JACK,x,0,d0,chi,dchi)
print '(A,I14,A,I12,A)', '# NDAT = ',ndat, &
  ' data. JACK = ',JACK, ' groups'
print '(A)', '# <o>, chi= (<o^2>-<o>^2)'
print '(A)', '# <o> +/- err' chi +/- err'
print '(4G28.17)',0,d0,chi,dchi
end program jackknife_errors
!=====
subroutine get_the_options
  use jack_function
  use getopt_m !from getopt.f90
  implicit none
  call getarg(0,prog)

  do
    select case( getopt( "-hj:d:" ))
    case( 'j' )
      read(optarg,*) JACK
    case( 'd' )
      read(optarg,*) MAXDAT
    case( 'h' )
      call usage
    case( '?' )
      print *, 'unknown option ', optopt
      stop
    case( char(0)) ! done with options
      exit
    case( '-' ) ! use — to exit from options
      exit
    case default
      print *, 'unhandled option ', optopt

```

```

end select
enddo

end subroutine get_the_options
!=====
subroutine usage
  use jack_function
  implicit none
  print '(3A)', 'Usage: ', TRIM(prog), ' [options]'
  print '( A)', '      -j   : No. jack groups Def. 10'
  print '( A)', '      -d   : Max. no. of data points read'
  print '( A)', 'Computes <o>, chi= (<o^2>-<o>^2)'
  print '( A)', 'Data is in one column from stdin.'
  stop
end subroutine usage

```

Για τη μεταγλώττιση χρησιμοποιούμε την εντολή

```
> gfortran -O2 getopt.f90 jack.f90 -o jack
```

Αν τα δεδομένα μας είναι σε μία στήλη στο αρχείο data, τότε η εντολή για να κάνουμε τον υπολογισμό των σφαλμάτων με 50 jack bins είναι:

```
> cat data | jack -j 50
```

Το πρόγραμμα δέχεται το πολύ MAXDAT=1,000,000 μετρήσεις. Αν έχουμε παραπάνω, πρέπει να χρησιμοποιήσουμε τον διακόπτη -d 2000000 για 2,000,000 μετρήσεις λ.χ. Η επιλογή να διαβάζουμε από το stdin γίνεται για να μπορεί το πρόγραμμα να χρησιμοποιηθεί σε συνδυασμό με φίλτρα. Για παράδειγμα, για την ανάλυση της μαγνήτισης με το πρόγραμμα που έχουμε γράψει για το πρότυπο Ising μπορούμε να χρησιμοποιήσουμε την εντολή:

```
> is -L 20 -b 0.4407 -s 1 -S 342 -n 2000000 | grep -v # | \
  awk -v L=20 '{print ($2>0)?($2/(L*L)):(-$2/(L*L))}' | \
  jack -j 50 -d 2000000 | grep -v # | \
  awk -v b=0.4407 -v L=20 '{print $1,$2,b*L*L*$3,b*L*L*$4}'
```

Η παραπάνω εντολή είναι μία γραμμή, αν αφαιρέσουμε τις τελευταίες ανάποδα κάθετες γραμμές (backslash - '\'). Για πληρότητα ας την εξηγήσουμε: Η πρώτη γραμμή τρέχει το πρόγραμμα is για το πρότυπο Ising για  $N = L \times L = 20 \times 20$  πλεγματικές θέσεις (-L 20) και  $\beta = 0.4407$  (-b 0.4407). Αρχίζει από “καυτή” διάταξη σπιν (-s 1) και κάνει 2,000,000 μετρήσεις (-n 2000000). Φιλτράρουμε με την εντολή grep -v τα σχό-

για από την έξοδο του προγράμματος. Στη δεύτερη γραμμή, καλούμε την `awk` ορίζοντας τη μεταβλητή  $L=20$  να τυπώσει την απόλυτη τιμή της δεύτερης στήλης διαιρεμένης με τον αριθμό των πλεγματικών θέσεων  $L \cdot L$ . Στην τρίτη γραμμή, γίνεται ο υπολογισμός της μέσης τιμής  $\langle m \rangle$  και  $\langle (m - \langle m \rangle)^2 \rangle$  με τα σφάλματά τους από το πρόγραμμα `jack`. Από τα αποτελέσματα αφαιρούνται τα σχόλια με την εντολή `grep -v`. Η τέταρτη γραμμή γίνεται απλά για τον υπολογισμό της μαγνητικής επιδεκτικότητας (13.33), όπου πρέπει να πολλαπλασιάσουμε με τον παράγοντα  $\beta N = \beta L^2$  τις διακυμάνσεις  $\langle (m - \langle m \rangle)^2 \rangle$  και το σφάλμα τους για να πάρουμε την  $\chi$ .

### 13.8.2 Η Μέθοδος Bootstrap

Στην υποενότητα αυτή, παραθέτουμε το πρόγραμμα για τον υπολογισμό των σφαλμάτων με τη μέθοδο `bootstrap` σύμφωνα με τα όσα είπαμε στην ενότητα 13.6.3. Το πρόγραμμα κωδικοποιείται στο αρχείο `boot.f90`:

```
!=====
!file: boot.f90
MODULE boot_function
  implicit none
  SAVE
  integer :: SAMPLES, MAXDAT
  character(200) :: prog
  integer :: seed
  CONTAINS
!-----
!jackknife function:
  subroutine bootstrap(ndat,samples,x,&
    av0,er0,avchi,erchi)
    integer :: ndat,samples !local samples...
    real(8),dimension(0:) :: x
    real(8) :: av0,er0,avchi,erchi
    integer :: i,j,k
    real(8),allocatable :: O(:),O2(:),chi(:)
!-----
    ALLOCATE(O(0:samples-1));ALLOCATE(O2(0:samples-1));
    ALLOCATE(chi(0:samples-1))
    O=0.0D0;O2=0.0D0;chi=0.0D0;
    do j=0,samples-1
      do i=0,ndat-1
        k = INT(ndat*drandom()) ! 0,...,ndat-1
        O(j) = O(j) + x(k)
        O2(j) = O2(j) + x(k)*x(k)
      enddo
    enddo
```



```

      O (j) = O(j)/ndat; O2(j) = O2(j)/ndat
      chi(j) = O2(j)-O(j)*O(j)
    enddo
!-----
    av0 = SUM(O)/samples; avchi=SUM(chi)/samples
    er0 = sqrt(SUM((O -av0 )*(O -av0 ))/samples)
    erchi = sqrt(SUM((chi-avchi)*(chi-avchi))/samples)
!compute the real avO:
    av0 = SUM(x(0:ndat-1))/ndat
!-----
    DEALLOCATE(O);DEALLOCATE(chi)
  end subroutine bootstrap
!-----
  real(8) function drandom()
    implicit none
    integer ,parameter :: a = 16807
    integer ,parameter :: m = 2147483647
    integer ,parameter :: q = 127773
    integer ,parameter :: r = 2836
    real(8),parameter :: f = (1.0D0/m)
    integer :: p
    real(8) :: dr
101 continue
    p = seed/q
    seed = a*(seed- q*p) - r*p
    if(seed .lt. 0) seed = seed + m
    dr = f*seed
    if( dr .le. 0.0D0 .or. dr .ge. 1.0D0) goto 101
    drandom = dr
  end function drandom
!-----
  subroutine locerr(errmes)
    implicit none
    character(*) :: errmes
    write(0,'(A,A)') ,TRIM(prog),':',TRIM(errmes),' Exiting .... '
    stop 1
  end subroutine locerr
END MODULE boot_function
!=====
program bootstrap_errors
  use boot_function
  implicit none
  integer :: ndat,chk
  real(8) :: O,dO,chi,dchi
  real(8),allocatable :: x(:)
  MAXDAT=1000000;SAMPLES=1000
  call get_the_options
  ALLOCATE(x(0:MAXDAT-1),STAT=chk)
  if(chk > 0) call locerr('Not enough memory for x')

```

```

ndat=0
do while ( ndat < MAXDAT)
  read(*,*,END=101)x(ndat)
  ndat = ndat+1
enddo
101 continue
if(ndat >= MAXDAT) write(0, '(3A,I14,A,I14)') &
  '# ',TRIM(prog), &
  ': Warning: read ndat=', ndat, &
  ' and reached the limit: ',MAXDAT
open (28, file="/dev/urandom", access="stream",&
  form="unformatted")
read (28) seed
seed = ABS(seed)
close(28)
call bootstrap(ndat,SAMPLES,x,0,d0,chi,dchi)
print '(A,I14,A,I12,A)',&
  '# NDAT = ',ndat,' data. SAMPLES = ',SAMPLES,' groups'
print '(A',&
  '# <o>, chi= (<o^2>-<o>^2)',&
  '# <o> +/- err',&
  'chi +/- err'
print '(4G28.17)',0,d0,chi,dchi
end program bootstrap_errors
!=====
subroutine get_the_options
  use boot_function
  use getopt_m !from getopt.f90
  implicit none
  call getarg(0,prog)

  do
    select case( getopt( "--hs:d:" ))
    case( 's' )
      read(optarg,*) SAMPLES
    case( 'd' )
      read(optarg,*) MAXDAT
    case( 'h' )
      call usage
    case( '?' )
      print *, 'unknown option ', optopt
      stop
    case( char(0)) ! done with options
      exit
    case( '-' ) ! use -- to exit from options
      exit
    case default
      print *, 'unhandled option ', optopt
    end select
  end do

```

```

enddo

end subroutine get_the_options
!=====
subroutine usage
...
end subroutine usage
!=====

```

Για τη μεταγλώττιση χρησιμοποιούμε την εντολή

```
> gfortran -O2 getopt.f90 boot.f90 -o boot
```

Αν τα δεδομένα μας είναι σε μία στήλη στο αρχείο data, τότε η εντολή για να κάνουμε τον υπολογισμό των σφαλμάτων με 500 samples είναι:

```
> cat data | boot -s 500
```

Το πρόγραμμα δέχεται το πολύ 1,000,000 μετρήσεις, όπως και στο πρόγραμμα jack. Αν έχουμε παραπάνω, πρέπει να χρησιμοποιήσουμε το διακόπτη -d 2000000 για 2,000,000 μετρήσεις λ.χ. Για παράδειγμα, για την ανάλυση της μαγνήτισης με το πρόγραμμα που έχουμε γράψει για το πρότυπο Ising μπορούμε να χρησιμοποιήσουμε την εντολή:

```

> is -L 20 -b 0.4407 -s 1 -S 342 -n 2000000 | grep -v # | \
awk -v L=20 '{print ($2>0)?($2/(L*L)):(-$2/(L*L))}' | \
boot -s 1000 -d 2000000 | grep -v # | \
awk -v b=0.4407 -v L=20 '{print $1,$2,b*L*L*$3,b*L*L*$4}'

```

Η παραπάνω εντολή είναι μία γραμμή, αν αφαιρέσουμε τις τελευταίες ανάποδα κάθετες γραμμές (backslash - '\').

### 13.8.3 Σύγκριση των Μεθόδων

Στην υποενότητα αυτή θα χρησιμοποιήσουμε και θα συγκρίνουμε τις μεθόδους που αναφέραμε στις προηγούμενες ενότητες για τον υπολογισμό των σφαλμάτων. Θα χρησιμοποιήσουμε τη σχέση (13.43), τη μέθοδο jackknife (13.48) και τη μέθοδο bootstrap (13.52). Για να φανούν οι διαφορές, θα χρησιμοποιήσουμε δεδομένα με ισχυρό αυτοσυσχετισμό. Επιλέγουμε στο πρότυπο Ising τον αλγόριθμο Metropolis για  $L = 40$ ,  $\beta = 0.4407 \approx \beta_c$ . Ειδικότερα, θα μελετήσουμε τη μαγνήτιση ανά πλεγματική θέση (13.31). Κάνουμε 1,000,000 μετρήσεις με τις εντολές:

```

> ./is -L 40 -b 0.4407 -s 1 -S 5434365 -n 1000000 \
> outL40b0.4407.dat &
> grep -v # outL40b0.4407.dat | \
awk -v L=40 '{ if ($2<0){ $2=-$2 }; print $2/(L*L) }' \
> outL40b0.4407.m
> cat outL40b0.4407.m | autoc -t 10000 -n 1000000 \
> outL40b0.4407.rhom

```

Το αρχείο outL40b0.4407.m έχει σε μία στήλη τις μετρήσεις της μαγνήτισης, ενώ το αρχείο outL40b0.4407.rhom τη συνάρτηση αυτοσυσχετισμού και τον ολοκληρωμένο χρόνο αυτοσυσχετισμού. Με το gnuplot μελετούμε το χρόνο αυτοσυσχετισμού, όπως περιγράψαμε από τη σελίδα 577 και μετά. Παίρνουμε  $\tau_m = 286.3(3)$ . Υπολογίζουμε και τον ολοκληρωμένο χρόνο αυτοσυσχετισμού με αποτέλεσμα  $\tau_{\text{int},m} = 254(1)$ .

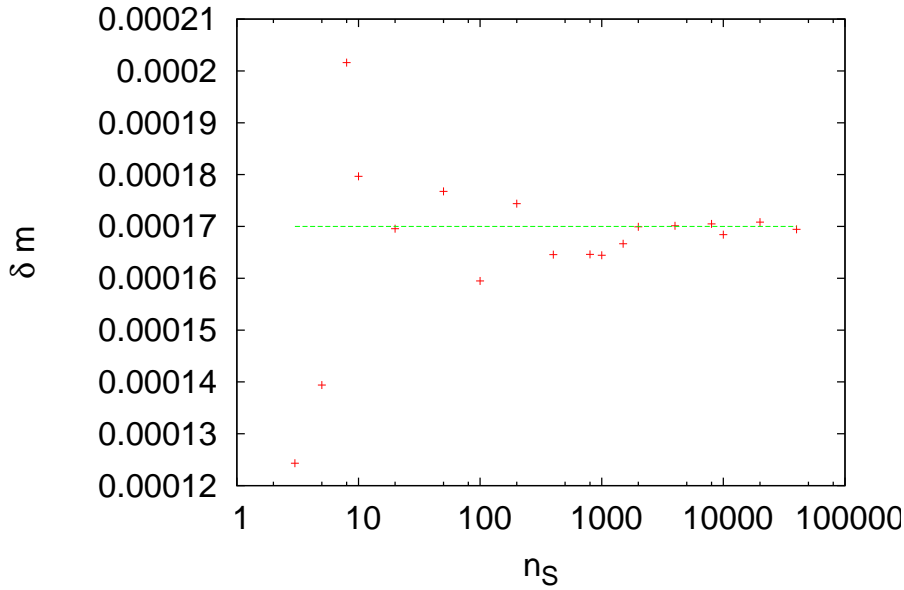
Η μέση τιμή  $\langle m \rangle = 0.638682$ . Η εφαρμογή της σχέσης (13.42) (που προϋποθέτει στατιστικά ανεξάρτητες μετρήσεις) δίνει το (υποεκτιμημένο) σφάλμα στατιστικής απόκλισης  $\delta_c m = 0.00017$ . Χρησιμοποιώντας τη σχέση (13.43), έχουμε  $\delta m = \sqrt{1 + 2\tau\delta_c m} \approx 0.004$ . Τη μαγνητική επιδεκτικότητα δεν είναι δυνατόν να την υπολογίσουμε με τον τρόπο αυτό. Άρα, δίνουμε το αποτέλεσμα

$$\langle m \rangle = 0.639 \pm 0.004 \equiv 0.639(4). \quad (13.57)$$

Για τον υπολογισμό της μαγνητικής επιδεκτικότητας είναι αναγκαίο να χρησιμοποιήσουμε μία από τις μεθόδους jackknife ή bootstrap. Η δεύτερη εφαρμόζεται αρχικά με μεταβλητό αριθμό δειγμάτων  $n_S$ , έτσι ώστε να προσδιορίσουμε τον βέλτιστο αριθμό από αυτά για να χρησιμοποιήσουμε στην ανάλυσή μας. Στο σχήμα 13.24 φαίνονται τα αποτελέσματα για τη μαγνήτιση. Παρατηρούμε πολύ γρήγορη σύγκλιση στο σφάλμα της στατιστικής απόκλισης  $\delta_c m = 0.00017$  για μικρό αριθμό δειγμάτων. Θα μπορούσαμε με ασφάλεια για την ανάλυση των δεδομένων της μαγνήτισης να επιλέγαμε  $n_S = 100$ . Για την περίπτωση της μαγνητικής επιδεκτικότητας η σύγκλιση είναι πιο αργή, αλλά πάλι θα μπορούσαμε να πάρουμε  $n_S = 500$ . Παίρνουμε  $\chi = 20.39$  και για το σφάλμα  $\delta_c \chi = 0.0435$ . Η τιμή του σφάλματος, όμως, αφορά ανεξάρτητες μετρήσεις, κάτι που δε συμβαίνει στην περίπτωσή μας. Η τιμή αυτή θα πρέπει να διορθωθεί από τον παράγοντα  $\sqrt{1 + 2\tau_m}$  και να δώσει  $\delta \chi = 1$ . Άρα,

$$\chi = 20 \pm 1 \equiv 20(1). \quad (13.58)$$

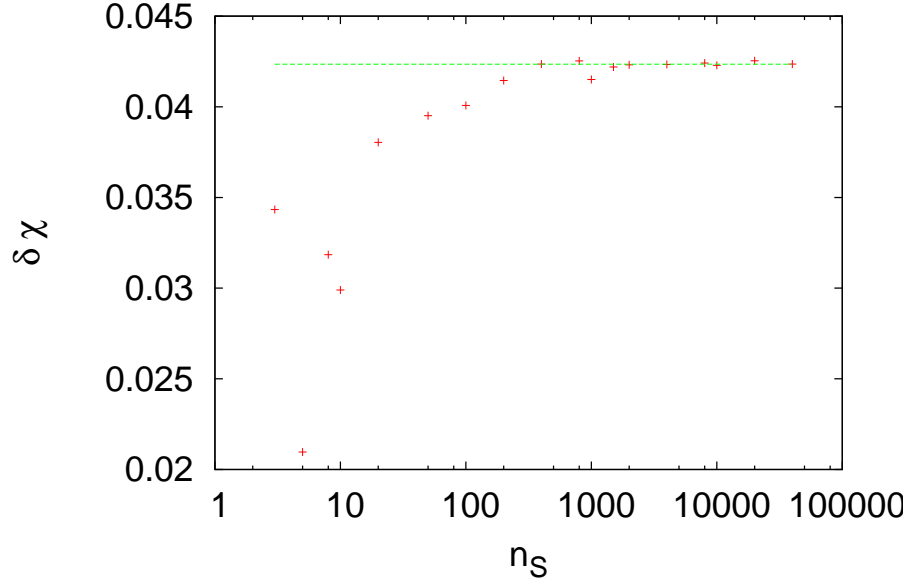
Αξίζει να παρατηρηθεί ότι το σφάλμα είναι αρκετά μεγάλο, κάτι που δείχνει την πρακτική δυσκολία που φέρνουν οι μεγάλοι χρόνοι αυτοσυσχετισμού [στην πραγματικότητα έχουμε μόνο  $n/(2\tau_m) \approx 1,000,000/(2 \times$



Σχήμα 13.24: Το σφάλμα  $\delta m$  της μαγνήτισης υπολογισμένο με τη μέθοδο bootstrap ως συνάρτηση του αριθμού των δειγμάτων  $n_S$ . Παρατηρούμε πολύ γρήγορη σύγκλιση στην τιμή του σφάλματος που παίρνουμε από τη σχέση (13.42)  $\delta_c m = 0.00017$ .

286)  $\approx 1750$  ανεξάρτητες μετρήσεις]. Επίσης, σημειώνουμε ότι για τον υπολογισμό του σφάλματος με τον τρόπο αυτό είναι αναγκαία η γνώση του  $\tau_m$ .

Πιο ασφαλής, γρήγορη και ευσταθής μέθοδος είναι η μέθοδος jackknife. Εδώ, δε θα χρειαστεί η εκ των προτέρων γνώση του  $\tau_m$ . Για τον προσδιορισμό του σφάλματος αρκεί να μελετήσουμε τα δεδομένα με μεταβλητό αριθμό από jackknife bins  $n_b$ . Στο σχήμα 13.26 φαίνονται τα αποτελέσματά μας για τη μαγνήτιση. Όταν τα jackknife bins  $n_b = n$ , τότε τα δείγματά μας αποτελούνται από όλες τις μετρήσεις εκτός από ένα στοιχείο. Τότε το σφάλμα θα είναι το ίδιο με το σφάλμα της στατιστικής απόκλισης και θα είναι υποεκτιμημένο κατά τον γνωστό παράγοντα  $\sqrt{1 + 2\tau_m}$ . Αυτό φαίνεται στο σχήμα 13.26 όπου παρατηρείται μια πολύ αργή σύγκλιση στην τιμή  $\delta_c m = 0.00017$ . Η επίδραση των αυτοσυσχετισμών εξαφανίζεται, καθώς από το δείγμα μας αφαιρούμε (bin width)  $\approx 2\tau_m$  μετρήσεις. Αυτό γίνεται, όταν  $n_b \approx n/(\text{bin width}) = n/(2\tau_m) = 1,000,000/572 \approx 1750$ . Φυσικά, αυτή η εκτίμηση δίνει την τάξη μεγέθους και μια προσεκτική μελέτη είναι αναγκαία για τον προσδιορισμό του σωστού αριθμού  $n_b$ . Στο σχήμα 13.26 βλέπουμε ότι το σφάλμα συγχλίνει για  $100 < n_b < 800$  στην τιμή  $\delta m = 0.0036$  που είναι αρκετά

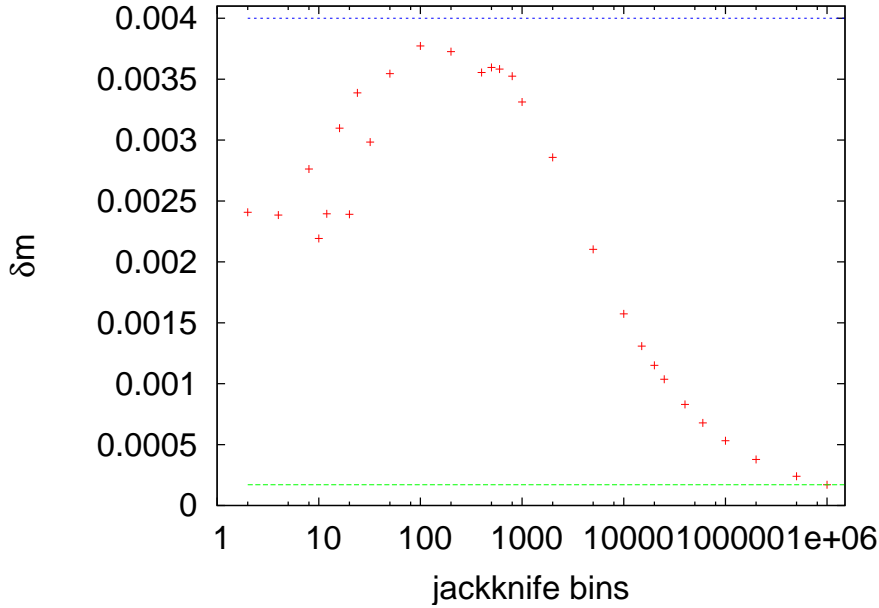


Σχήμα 13.25: Το σφάλμα  $\delta\chi$  της μαγνητικής επιδεκτικότητας υπολογισμένο με τη μέθοδο bootstrap ως συνάρτηση του αριθμού των δειγμάτων  $n_S$ . Παρατηρούμε σύγκλιση για  $n_S > 1000$  στην τιμή  $\delta_c\chi = 0.0435$ .

κοντά στην τιμή  $\sqrt{1 + 2\tau_m\delta_c}m \approx 0.004$  (και ακόμα κοντύτερα, αν χρησιμοποιήσουμε τον ολοκληρωμένο χρόνο αυτοσυσχετισμού  $\tau_{\text{int},m} = 254$ ). Επίσης, παρατηρούμε ότι χοντρική εκτίμηση του σφάλματος μπορούμε να πάρουμε για πολύ μικρό αριθμό  $n_b \approx 20 - 40$ , κάτι που μπορεί να χρησιμοποιηθεί για γρήγορους υπολογισμούς.

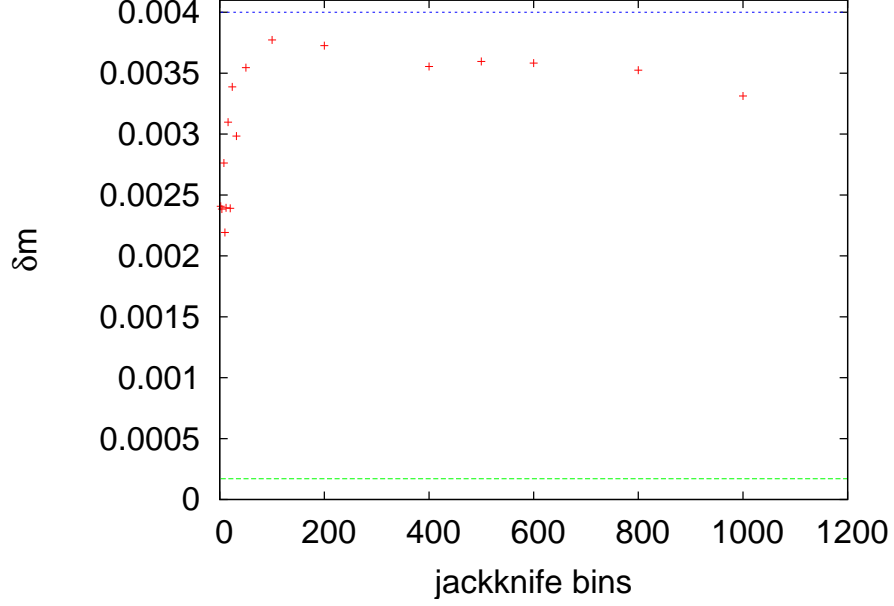
Παρόμοια αποτελέσματα παίρνουμε και για τη μαγνητική επιδεκτικότητα  $\chi$ , όπου το σφάλμα συγκλίνει στην τιμή  $\delta\chi = 0.86$  σε συμφωνία με τις προηγούμενες εκτιμήσεις μας. Για  $n_b \rightarrow n$ , το σφάλμα συγκλίνει στο (υποεκτιμημένο) σφάλμα  $\delta_c\chi = 0.0421$ . Γεννιέται το ερώτημα αν η μέθοδος bootstrap μπορεί να χρησιμοποιηθεί με ανάλογο τρόπο με αυτό της jackknife για τον προσδιορισμό του πραγματικού σφάλματος  $\delta m$ ,  $\delta\chi$  χωρίς τη γνώση του  $\tau_m$ . Η απάντηση είναι ναι. Θα χρησιμοποιήσουμε τη διαίσθησή μας από τη μέθοδο jackknife και binning. Χωρίζουμε τα δεδομένα μας σε  $n_b$  bins<sup>27</sup> των οποίων το εύρος (bin width)  $= n/n_b$ . Κάθε bin θεωρείται ως μια ανεξάρτητη μέτρηση που είναι η μέση τιμή της πα-

<sup>27</sup>Προσοχή, αν το  $n_b$  είναι πολύ μικρό, τότε έχουμε συστηματικό σφάλμα στην τιμή της  $\chi$ . Οι τιμές, όπως και τα σφάλματα, πρέπει να παρακολουθούνται, έτσι ώστε να βρεθεί το εύρος των τιμών του  $n_b$  όπου το αποτέλεσμα είναι ανεξάρτητο του  $n_b$ .



Σχήμα 13.26: Το σφάλμα  $\delta m$  της μαγνήτισης υπολογισμένο με τη μέθοδο jackknife ως συνάρτηση του αριθμού των δειγμάτων jackknife bins  $n_b$ . Παρατηρούμε σύγκλιση για  $100 < n_b < 800$  στην τιμή  $\delta m = 0.0036$ . Φαίνεται καθαρά ότι καθώς φτάνουμε στο όριο να αφαιρούμε μόνο ένα στοιχείο από τα δεδομένα ( $n_b = n$ ), το σφάλμα πλησιάζει στην τιμή που υπολογίζεται από τη σχέση (13.42)  $\delta_c m = 0.00017$ . Οι οριζόντιες γραμμές αντιστοιχούν στις τιμές  $\delta_c m$  και  $\sqrt{1 + 2\tau_m} \delta_c m \approx 0.004$  όπου  $\tau_m = 286.3$ . Ο λόγος  $\delta m / \delta_c m \approx \sqrt{1 + 2\tau_m}$ .

ρατηρήσιμης ποσότητας μέσα στο bin αυτό. Παίρνουμε  $n_S = 1000$  και εφαρμόζουμε τη μέθοδο bootstrap στο δείγμα των  $n_b$  μετρήσεων. Στα σχήματα 13.30 και 13.32 δείχνουμε τα αποτελέσματά μας ως συνάρτηση του (bin width). Όταν αυτό γίνει (bin width)  $\approx 2\tau_m$  οι μετρήσεις μας θα είναι πρακτικά ανεξάρτητες και θα πρέπει να πάρουμε το πραγματικό σφάλμα. Στο σχήμα 13.30 δείχνουμε τα αποτελέσματα για τη μαγνήτιση όπου για bin width=1, έχουμε το αποτέλεσμα  $\delta_c m = 0.00017$ . Όταν το εύρος γίνει  $> 2\tau_m$ , το σφάλμα γίνεται  $\delta m = 0.0036$ , το οποίο το διαβάζουμε στο plateau για  $1100 < (\text{bin width}) < 16000$ , σε συμφωνία με τη μέθοδο jackknife. Στο σχήμα 13.30 δείχνουμε τα αποτελέσματα για τη μαγνητική επιδεκτικότητα. Πάλι για bin width=1, έχουμε το αποτέλεσμα  $\delta_c \chi = 0.0421$ . Όταν το εύρος γίνει  $> 2\tau_m$ , το σφάλμα γίνεται  $\delta \chi = 0.615$ , το οποίο το διαβάζουμε στο plateau για  $500 < (\text{bin width}) < 1000$  που δεν είναι πολύ διαφορετικό από το αποτέλεσμα που πήραμε με τη μέθοδο jackknife.



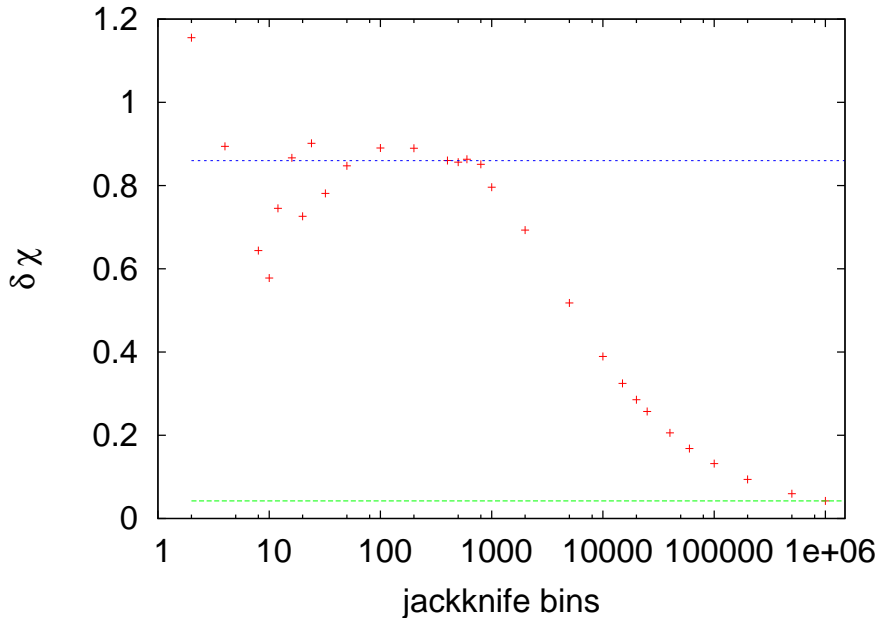
Σχήμα 13.27: Το σχήμα 13.26 μεγενθυμένο στην περιοχή που εμφανίζεται πλατό στις τιμές του  $\delta m$ . Οι οριζόντιες γραμμές αντιστοιχούν στις τιμές  $\delta_c m$  και  $\sqrt{1 + 2\tau_m} \delta_c m \approx 0.004$  όπου  $\tau_m = 286.3$ . Ο λόγος  $\delta m / \delta_c m \approx \sqrt{1 + 2\tau_m}$ .

Κλείνουμε την υποενότητα αυτή με μια σημαντική παρατήρηση. Στην παρουσίασή μας υπολογίσαμε πρώτα το χρόνο αυτοσυσχετισμού  $\tau_m$  από τη συνάρτηση αυτοσυσχετισμού και στη συνέχεια, προσδιορίσαμε τότε οι μέθοδοι που χρησιμοποιήσαμε έδωσαν το (υποεκτιμημένο) σφάλμα στατιστικής απόκλισης των μετρήσεων και τότε το πραγματικό που λαμβάνει υπόψη τους αυτοσυσχετισμούς. Στην πράξη όμως, οι μέθοδοι αυτοί αποτελούν μία ανεξάρτητη μέθοδο υπολογισμού του  $\tau_m$  από τις σχέσεις  $\delta \mathcal{O} / \delta_c \mathcal{O} = \sqrt{1 + 2\tau_{\mathcal{O}}}$ . Έτσι, από τα παραπάνω συνάγεται ότι

$$\tau_m = \frac{1}{2} \left( \left( \frac{\delta m}{\delta_c m} \right)^2 - 1 \right) = \frac{1}{2} \left( \left( \frac{\delta \chi}{\delta_c \chi} \right)^2 - 1 \right) \dots \quad (13.59)$$

σχέσεις που μπορούν να χρησιμοποιηθούν για τον έλεγχο της αυτοσυνέπειας της ανάλυσης των δεδομένων μας. Φυσικά θα υπενθυμίσουμε ότι ένα σύστημα έχει πολλούς χρόνους αυτοσυσχετισμού και διαφορετικές παρατηρήσιμες ποσότητες και μπορεί η συμπεριφορά τους να καθορίζεται από διαφορετικούς χρόνους αυτοσυσχετισμού (fast modes, slow modes).

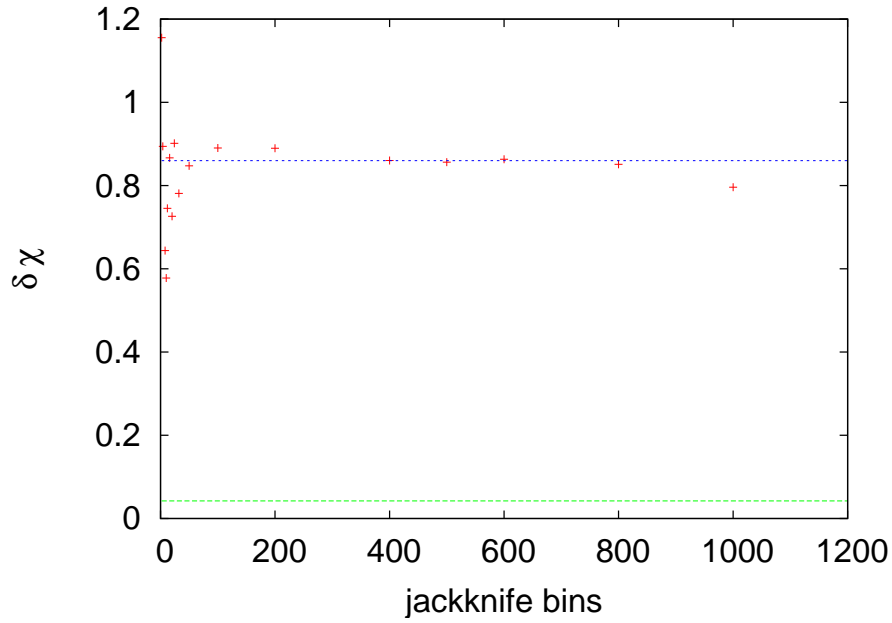




Σχήμα 13.28: Το σφάλμα  $\delta\chi$  της μαγνητικής επιδεκτικότητας υπολογισμένο με τη μέθοδο jackknife ως συνάρτηση του αριθμού των δειγμάτων jackknife bins  $n_b$ . Παρατηρούμε σύγκλιση για  $100 < n_b < 800$  στην τιμή  $\delta\chi = 0.86$ . Φαίνεται καθαρά ότι, καθώς φτάνουμε στο όριο να αφαιρούμε μόνο ένα στοιχείο από τα δεδομένα ( $n_b = n$ ), το σφάλμα πλησιάζει στην τιμή που θα παίρναμε, αν θεωρούσαμε τις μετρήσεις ανεξάρτητες  $\delta_{c\chi} = 0.0421$ . Η τιμή αυτή είναι πολύ κοντά στην τιμή που πήραμε με τη μέθοδο bootstrap. Οι τιμές αυτές απεικονίζονται στις δύο οριζόντιες γραμμές. Ο λόγος  $\delta\chi/\delta_{c\chi} \approx \sqrt{1 + 2\tau_m}$ .

## 13.9 Ασκήσεις

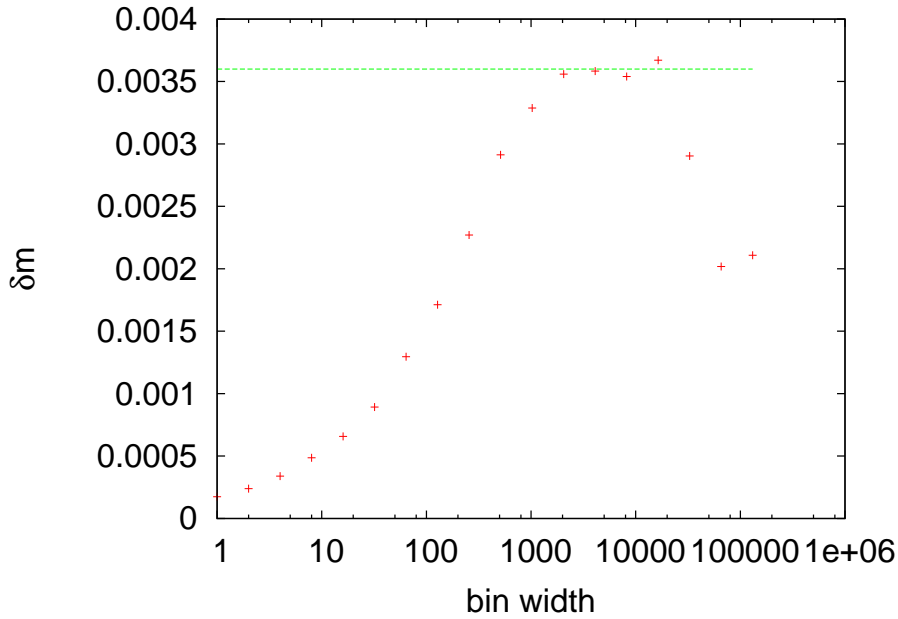
1. Αποδείξτε ότι η (13.22) ικανοποιεί τη συνθήκη λεπτομερούς ισοροπίας.
2. Γράψτε πρόγραμμα που θα σας τυπώνει την ποσότητα μνήμης σε bytes που πιάνουν οι ακόλουθοι τύποι μεταβλητών: `character`, `integer`, `integer(8)`, `real`, `real(8)`. Υπολογίστε πόση μνήμη απαιτείται για ένα array 2,000,000 θέσεων για κάθε ένα τύπο μεταβλητών.
3. Μεταβάλετε τον πρόγραμμα, έτσι ώστε να μετράει τον μέσο λόγο αποδοχής acceptance των βημάτων Metropolis. Δηλ. μετρήστε τον αριθμό των δεκτών αλλαγών στα σπιν σε σχέση με τον αριθμό των επιχειρούμενων αλλαγών. Εξετάστε τη σχέση του αριθμού με τη



Σχήμα 13.29: Το σχήμα 13.28 μεγενθυμένο στην περιοχή που εμφανίζεται πλατό στις τιμές του  $\delta\chi$ . Παρατηρούμε σύγκλιση για  $100 < n_b < 800$  στην τιμή  $\delta\chi = 0.86$ . Οι δύο οριζόντιες γραμμές αντιστοιχούν στις τιμές των  $\delta\chi$  και  $\delta_c\chi$ . Ο λόγος  $\delta\chi/\delta_c\chi \approx \sqrt{1 + 2\tau_m}$ .

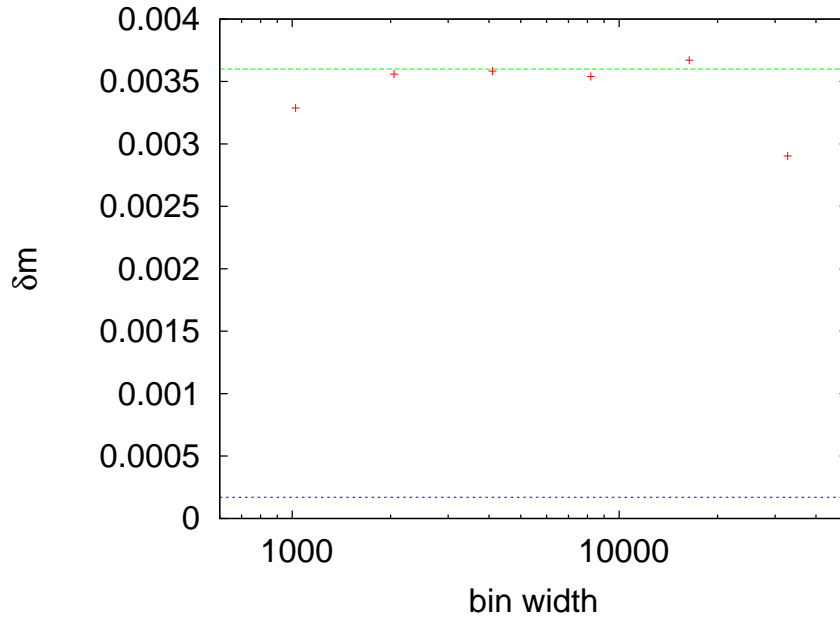
θερμοκρασία και το μέγεθος του συστήματος. Πάρτε  $L = 20$  και  $\beta = 0.20, 0.30, 0.40, 0.42, 0.44, 0.46, 0.48, 0.50$ . Στη συνέχεια, πάρτε  $\beta = 0.20, L = 10, 20, 40, 80, 100$ . Επαναλάβετε για τις ίδιες τιμές του  $L$  για  $\beta = 0.44$  και  $\beta = 0.48$ .

4. Αναπαράγετε τα σχήματα 13.12 και υπολογίστε τον  $\tau_m$ . Επαναλάβετε το ίδιο για την ενέργεια και υπολογίστε τον  $\tau_e$ . Συγκρίνετε τις τιμές που βγάλατε με τις τιμές των  $\tau_{\text{int},m}$  και  $\tau_{\text{int},e}$ .
5. Αναπαράγετε τα σχήματα 13.15 και επαναλάβετε τον υπολογισμό σας για την ενέργεια.
6. Αναπαράγετε τα σχήματα 13.17. Επαναλάβετε το ίδιο και για την ενέργεια. Στη συνέχεια, κάντε τα ανάλογα σχήματα για τους  $\tau_{\text{int},m}$  και  $\tau_{\text{int},e}$  σαν συνάρτηση του  $t_{\text{max}}$  (βλ. σχήμα 13.14)
7. Αναπαράγετε τα σχήματα 13.19 και 13.20. Επαναλάβετε το ίδιο και για την ενέργεια. Στη συνέχεια, κάντε τα ανάλογα σχήματα για τους  $\tau_{\text{int},m}$  και  $\tau_{\text{int},e}$  σαν συνάρτηση του  $t_{\text{max}}$  (βλ. σχήμα 13.14)



Σχήμα 13.30: Το σφάλμα  $\delta m$  της μαγνήτισης υπολογισμένο με τη μέθοδο bootstrap. Παίρνουμε  $n_S = 1000$  και χωρίζουμε τα δεδομένα μας σε bins των οποίων το εύρος bin width φαίνεται στον οριζόντιο άξονα. Κάθε bin θεωρείται ως μια ανεξάρτητη μέτρηση. Για bin width=1, έχουμε το αποτέλεσμα  $\delta_c m = 0.00017$ . Όταν το εύρος γίνει  $> 2\tau_m$ , το σφάλμα γίνεται  $\delta m = 0.0036$ , το οποίο το διαβάζουμε στο plateau για  $1100 < (\text{bin width}) < 16000$ , σε συμφωνία με τη μέθοδο jackknife. Οι οριζόντιες γραμμές αντιστοιχούν στις τιμές των  $\delta_c m$  και  $\delta m$  που αναφέρθηκαν παραπάνω.

8. Κάνετε τις απαραίτητες μετατροπές στον κώδικα, ώστε να προσομοιώσετε το πρότυπο Ising παρουσία εξωτερικού μαγνητικού πεδίου  $B$  (βλ. εξ. (13.2)). Υπολογίστε τη μαγνήτιση  $m(\beta, B)$  για  $L = 32$  και  $B = 0.2, 0.4, 0.6, 0.8, 1.0$  στο ενδιαφέρον φάσμα των θερμοκρασιών. Παρατηρήστε τη διαδικασία εύρεσης της θερμικής ισορροπίας, καθώς αυξάνει το  $B$  από διαφορετικές αρχικές καταστάσεις: Παγωμένη με σπιν παράλληλα στο  $B$ , παγωμένη με σπιν αντιπαράλληλα στο  $B$ , καυτή και την προηγούμενη κατάσταση σε γειτονική θερμοκρασία. Μελετήστε πως μεταβάλλεται η θερμοκρασία στην οποία έχουμε μετάβαση από μη μαγνητισμένη σε μαγνητισμένη κατάσταση, καθώς αυξάνει το  $B$ .
9. Φαινόμενο υστέρησης: Στην προηγούμενη άσκηση το σύστημα για  $B \neq 0$  παρουσιάζει μετάβαση φάση πρώτης τάξης, δηλ. ασυνέχεια στην τιμή της παραμέτρου τάξης. Εδώ, αυτή είναι η μαγνήτιση ως συνάρτηση του  $B$ . Παρουσιάζεται στο σύστημα αυτό το φαινόμενο



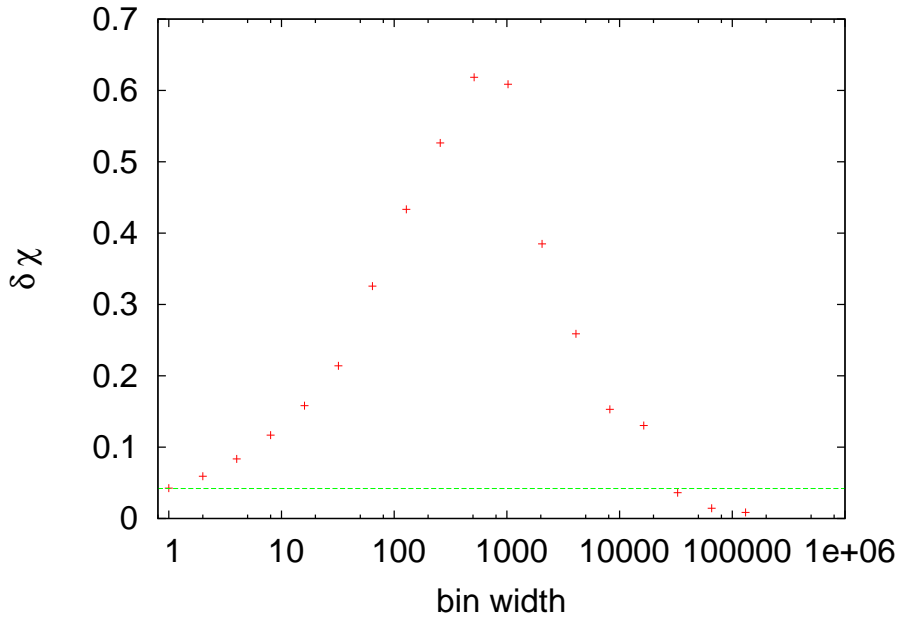
Σχήμα 13.31: Το σχήμα 13.30 στην περιοχή του plateau των τιμών του  $\delta m$ .

της υστέρησης. Για να το δείτε, θέστε  $L = 32$  και  $\beta = 0.55$ .

- (α') Φέρτε το σύστημα σε θερμική ισορροπία για  $B = 0$ .
- (β') Προσομοιώστε το σύστημα για  $B = 0.2$  χρησιμοποιώντας ως αρχική κατάσταση αυτή του προηγούμενου βήματος. Κάντε 100 sweeps και υπολογίστε την  $\langle m \rangle$ .
- (γ') Συνεχίστε την ίδια διαδικασία αυξάνοντας το μαγνητικό πεδίο κατά  $\delta B = 0.2$  κάθε φορά. Σταματήστε, όταν  $\langle m \rangle \approx 0.95$ .
- (δ') Από την τελευταία διάταξη του προηγούμενου βήματος αρχίστε να ελαττώνετε διαδοχικά το μαγν. πεδίο κατά  $\delta B = -0.2$  μέχρι  $\langle m \rangle \approx -0.95$ .
- (ε') Επαναλάβετε ξανά από την τελευταία διάταξη του συστήματος, αυξάνοντας το μαγν. πεδίο ξανά κατά  $\delta B = 0.2$  μέχρι  $\langle m \rangle \approx 0.95$ .

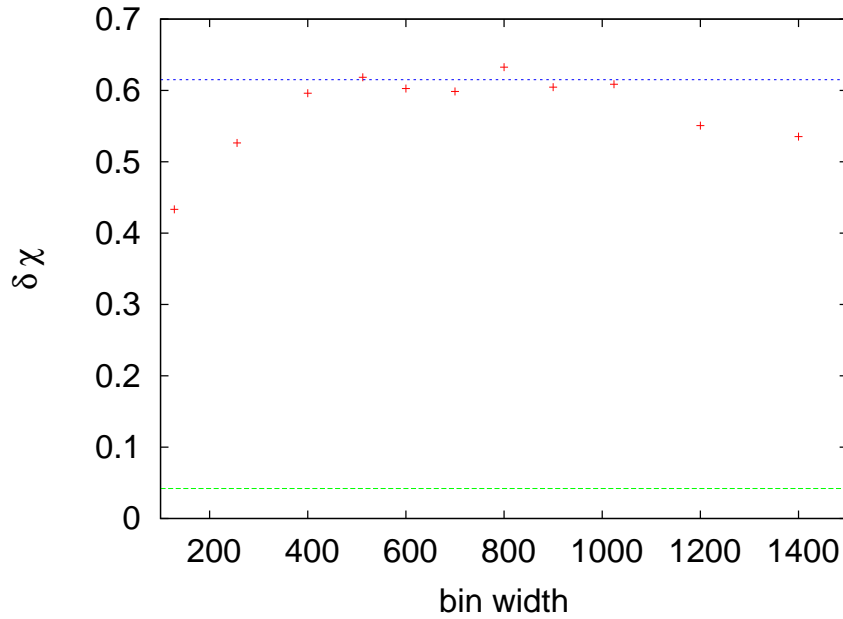
Τι παρατηρείτε; Φτιάξτε μια κλειστή καμπύλη στο επίπεδο  $(B, m)$ ; Ποιες είναι οι τιμές της μαγνήτισης για  $B = 0$ ; Γιατί είναι διαφορετικές;

Τα συστήματα που παρουσιάζουν μετάβαση πρώτης τάξης, παρουσιάζουν καταστάσεις που είναι τοπικά ελάχιστα της ελεύθε-



Σχήμα 13.32: Το σφάλμα  $\delta\chi$  της μαγνητικής επιδεκτικότητας υπολογισμένο με τη μέθοδο bootstrap. Παίρνουμε  $n_S = 1000$  και χωρίζουμε τα δεδομένα μας σε bins των οποίων το εύρος bin width φαίνεται στον οριζόντιο άξονα. Κάθε bin θεωρείται ως μια ανεξάρτητη μέτρηση. Για bin width=1, έχουμε το αποτέλεσμα  $\delta_c\chi = 0.0421$ . Όταν το εύρος γίνει  $> 2\tau_m$ , το σφάλμα γίνεται  $\delta\chi = 0.615$ , το οποίο το διαβάζουμε στο plateau για  $500 < (\text{bin width}) < 1000$  που δεν είναι πολύ διαφορετικό από το αποτέλεσμα που πήραμε με τη μέθοδο jackknife. Οι οριζόντιες γραμμές αντιστοιχούν στις τιμές των  $\delta_c\chi$  και  $\delta\chi$  που αναφέρθηκαν παραπάνω.

ρης ενέργειας, αλλά υπάρχει μόνο ένα πραγματικό ελάχιστο. Αυτό φαίνεται στο σχήμα 12.2 όπου απεικονίζονται οι δύο καταστάσεις να έχουν την ίδια ελεύθερη ενέργεια. Αυτό συμβαίνει ακριβώς στο κρίσιμο σημείο. Όταν φύγουμε από αυτό, τότε το ένα ελάχιστο είναι το πραγματικό, ενώ το άλλο έχει μεγαλύτερη ελεύθερη ενέργεια. Όταν το σύστημα βρεθεί σε αυτή την κατάσταση, τότε χρειάζεται μεγάλο χρόνο για να μπορέσει να υπερπηδήσει το φράγμα ελεύθερης ενέργειας και να βρει το πραγματικό ελάχιστο. Μία τέτοια κατάσταση ονομάζεται μετασταθής. Σε μια προσομοίωση Μόντε Κάρλο μια τέτοια περίπτωση παρουσιάζει μεγάλη δυσκολία, αφού μπορούμε να θεωρήσουμε κατά λάθος ότι βρισκόμαστε στη θεμελιώδη κατάσταση. Εσείς επαναλάβετε τις προσομοιώσεις που κάνατε παραπάνω, αλλά κάντε αυτή τη φορά 100,000 sweeps. Παρατηρήστε τη χρονοσειρά της μαγνήτισης και παρατηρήστε τις



Σχήμα 13.33: Το σχήμα 13.32 στην περιοχή του plateau των τιμών του  $\delta\chi$ .

μεταπτώσεις από τη μια μετασταθή κατάσταση στην άλλη. Κάνετε το ιστόγραμμα των τιμών της μαγνήτισης και εντοπίστε τη μετασταθή κατάσταση. Πώς μεταβάλλεται το ιστόγραμμα, καθώς αυξάνεται το  $B$ ;

10. Γράψτε κώδικα για το πρότυπο Ising στις 2 διαστάσεις, αλλά σε ένα πλέγμα τριγωνικό. Η κύρια διαφορά είναι ότι ο αριθμός των πλησιέστερων γειτόνων είναι  $z = 6$  αντί για  $z = 4$ . Για τη δομή του πλέγματος θα βρείτε χρήσιμη πληροφορία στο κεφ. 13.1.2 των Newman και Barkema (ειδικά από το σχήμα 13.4). Υπολογίστε ξανά τη μεταβολή της ενέργειας σε κάθε spin flip για το βήμα Metropolis. Από τα μέγιστα της μαγνητικής επιδεκτικότητας και της ειδικής θερμότητας εξετάστε αν η κρίσιμη θερμοκρασία του συστήματος είναι πράγματι  $\beta_c \approx 0.274653072$ . Παρατηρήστε ότι ενώ η τιμή της  $\beta_c$  αλλάζει σε σχέση με το τετραγωνικό πλέγμα, οι κρίσιμοι εκθέτες παραμένουν οι ίδιοι (παγκοσμιότητα).
11. Γράψτε κώδικα για το πρότυπο Ising στις τρεις διαστάσεις πάνω σε κυβικό πλέγμα. Χρησιμοποιήστε ελικοειδείς συνοριακές συνθήκες και τότε θα χρειαστούν πολύ λίγες μετατροπές στον κώδικα (μαζί με τα  $XNN=1$  και  $YNN=L$  θα έχετε και  $ZNN=L*L$ ).

12. Γράψτε κώδικα για το πρότυπο Ising στις τρεις διαστάσεις πάνω σε κυβικό πλέγμα. Χρησιμοποιήστε περιοδικές συνοριακές συνθήκες. [Υπόδειξη: Χρησιμοποιήστε πάλι ένα array  $s(N)$ . Πριν ξεκινήσει το πρόγραμμα υπολογίστε arrays  $XNN(-N:N)$ ,  $YNN(-N:N)$ ,  $ZNN(-N:N)$  που να δίνουν τους πλησιέστερους γείτονες της θέσης  $i$  από τις τιμές  $XNN(i)$ ,  $XNN(-i)$ ,  $YNN(i)$ ,  $YNN(-i)$ ,  $ZNN(i)$ ,  $ZNN(-i)$ .]

13. Προσομοιώστε το αντισιδηρομαγνητικό πρότυπο Ising. Στον κώδικα που έχετε αρκεί να πάρετε αρνητικές θερμοκρασίες. Ποια είναι η θεμελιώδης κατάσταση;

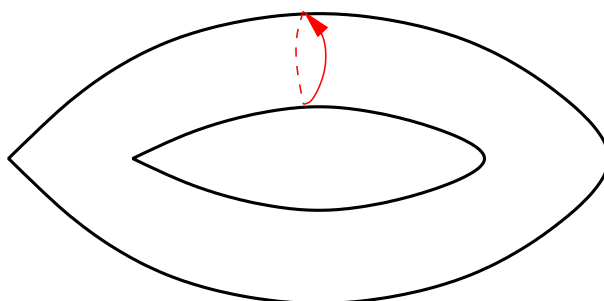
Ορίστε την staggered magnetization  $m_s$  να είναι η μαγνήτιση ανά πλεγματοειδή θέση ενός υποπλέγματος το οποίο περιέχει τα πλεγματοειδή σημεία με περιττή  $x$  και  $y$  συντεταγμένη. Θέστε  $L = 32$  και υπολογίστε την εξάρτηση της ενέργειας,  $m_s$ , ειδικής θερμότητας και μαγνητικής επιδεκτικότητας  $\chi$  και της staggered μαγνητικής επιδεκτικότητας  $\chi_s = \beta N / 4 \langle (m_s - \langle m_s \rangle)^2 \rangle$ .

Η  $\chi$  παρουσιάζει ένα μέγιστο στην περιοχή  $\beta \approx 0.4407$ . Υπολογίστε την τιμή της στη θερμοκρασία αυτή για  $L = 32 - 120$ . Δείξτε ότι η τιμή αυτή δεν αποκλίνει για  $L \rightarrow \infty$ , άρα δεν έχουμε ένδειξη μετάβασης φάσης από την  $\chi$ .

Επαναλάβετε για την  $\chi_s$ . Τι συμπεραίνετε; Συγκρίνετε τη συμπεριφορά της  $\langle m_s \rangle$  για το αντισιδηρομαγνητικό πρότυπο Ising με την  $\langle m \rangle$  του σιδηρομαγνητικού.

14. Μετατρέψτε το πρόγραμμα `boot.f90`, έτσι ώστε να κάνει binning στα δεδομένα. Αναπαράγετε τα σχήματα 13.30 και 13.32. (Λύση: Το πρόγραμμα `boot_bin.f90` στον υποκατάλογο `Tools/` του συνοδευτικού λογισμικού.)

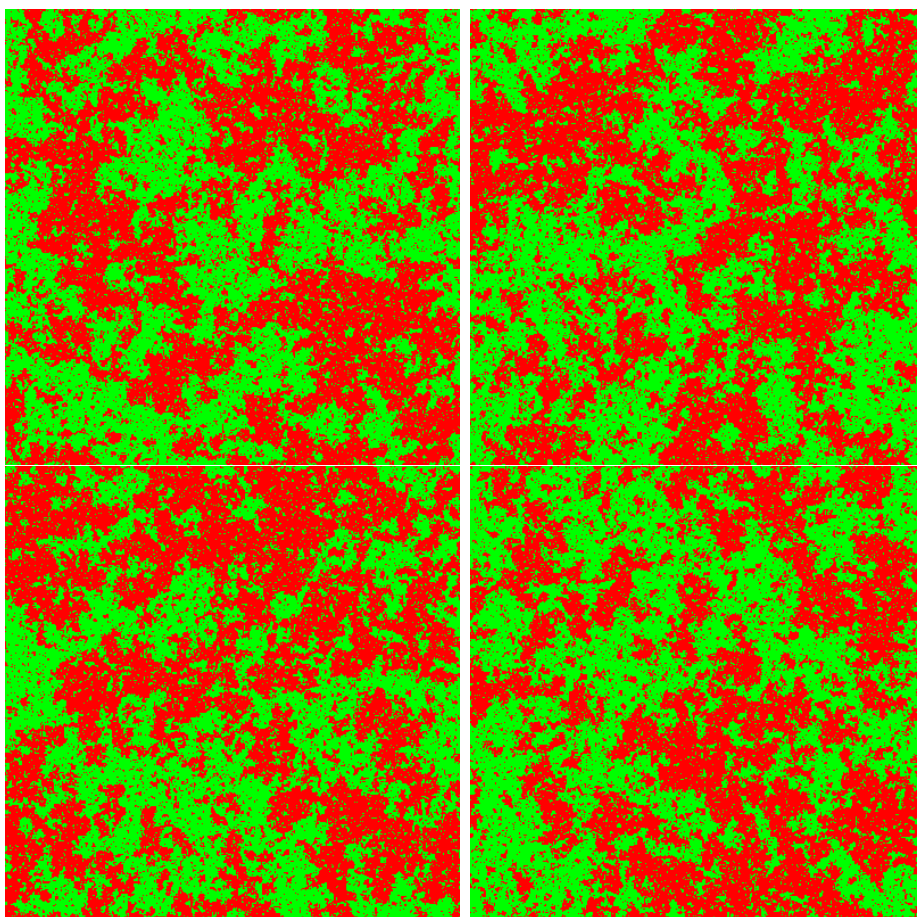
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
6	7	8	9	10	6	7	8	9	10	6	7	8	9	10
11	12	13	14	15	11	12	13	14	15	11	12	13	14	15
16	17	18	19	20	16	17	18	19	20	16	17	18	19	20
21	22	23	24	25	21	22	23	24	25	21	22	23	24	25
1	2	3	4	5	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	1	2	3	4	5
6	7	8	9	10	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	6	7	8	9	10
11	12	13	14	15	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	11	12	13	14	15
16	17	18	19	20	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	16	17	18	19	20
21	22	23	24	25	<del>21</del>	<del>22</del>	<del>23</del>	<del>24</del>	<del>25</del>	<del>21</del>	<del>22</del>	<del>23</del>	<del>24</del>	<del>25</del>
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
6	7	8	9	10	6	7	8	9	10	6	7	8	9	10
11	12	13	14	15	11	12	13	14	15	11	12	13	14	15
16	17	18	19	20	16	17	18	19	20	16	17	18	19	20
21	22	23	24	25	21	22	23	24	25	21	22	23	24	25



Σχήμα 13.34: Οριζόντια κίνηση στο τετραγωνικό πλέγμα με περιοδικές συνοριακές συνθήκες και  $L = 5$ . Η τροχιά είναι κύκλος.







Σχήμα 13.36: Διατάξεις σπιν για το πρότυπο Ising με  $L = 400$ ,  $\beta = 0.4292$  μετά από 4000, 9000, 12000 και 45000 sweeps αντίστοιχα. Παρατηρούμε τη δημιουργία μεγάλων clusters από όμοια σπιν τα οποία είναι δύσκολο να καταστρέψουμε και να δημιουργήσουμε αλλού με έναν single-flip αλγόριθμο, όπως ο Metropolis. Για το λόγο αυτό, οι χρόνοι αυτοσυσχετισμού είναι μεγάλοι.

# ΚΕΦΑΛΑΙΟ 14

## Κρίσιμοι Εκθέτες

### 14.1 Εισαγωγή

Ήδη αναφέραμε σε προηγούμενα κεφάλαια πως ένα φυσικό σύστημα το οποίο εμφανίζει μία συνεχή μετάβαση φάσης κατά την οποία, καθώς  $\beta \rightarrow \beta_c$  ή, ισοδύναμα, καθώς η αδιάστατη ή ανηγμένη θερμοκρασία<sup>1</sup>

$$t \equiv \frac{\beta_c - \beta}{\beta_c} \rightarrow 0, \quad (14.1)$$

το μήκος συσχετισμού στο θερμοδυναμικό όριο  $\xi \equiv \xi(\beta, L = \infty)$  απειρίζεται σύμφωνα με τη σχέση

$$\xi \sim |t|^{-\nu} \quad (\nu = 1 \text{ για 2d-Ising}), \quad (14.2)$$

βρίσκεται σε μία κλάση παγκοσμιότητας (universality class) η οποία χαρακτηρίζεται από κρίσιμους εκθέτες. Οι κρίσιμοι εκθέτες περιγράφουν την κύρια μη αναλυτική συμπεριφορά των φυσικών ποσοτήτων που παρουσιάζεται ασυμπτωτικά στο θερμοδυναμικό όριο<sup>2</sup>  $L \rightarrow \infty$ , όταν  $t \rightarrow 0$ . Αυτό που χαρακτηρίζει μία κλάση παγκοσμιότητας είναι ότι, ενώ αποτελείται από συστήματα που δίνονται από διαφορετικές Χαμιλτονιανές, τα συστήματα αυτά έχουν τους ίδιους κρίσιμους εκθέτες. Έτσι, αν στο πρότυπο Ising αλλάξουμε την αλληλεπίδραση πλησιέστερων γειτόνων προσθέτοντας όρους από πιο απομακρυσμένες πλεγματικές θέσεις (πάντα όμως σε απόσταση  $\ll \xi$ ) ή αν αλλάξουμε την τοπολογία του πλέγματος από τετραγωνικό σε τριγωνικό, εξαγωνικό κλπ, θα παραμείνουμε στην ίδια κλάση παγκοσμιότητας. Έτσι, μπορούμε να μελετήσουμε ένα πολύπλοκο φυσικό σύστημα στην περιοχή της μετάβασης

---

<sup>1</sup>Συνήθως στη βιβλιογραφία ορίζεται  $t = (T - T_c)/T_c$ , αλλά για  $t \ll 1$ , οι δύο ορισμοί είναι σχεδόν ισοδύναμοι (διαφέρουν κατά ένα όρο  $\sim t^2$ ).

<sup>2</sup>Προσοχή: πρώτα  $L \rightarrow \infty$  και μετά  $t \rightarrow 0$ .

φάσης χρησιμοποιώντας το απλούστερο δυνατόν μοντέλο που ανήκει στην ίδια κλάση παγκοσμιότητας. Για παράδειγμα, η μετάβαση φάσης υγρό/ατμός κοντά στο τρικρίσιμο σημείο ανήκει στην ίδια κλάση παγκοσμιότητας με το πρότυπο Ising στο επίπεδο.

Στο πρότυπο Ising είδαμε στο Κεφάλαιο 12 τους κρίσιμους εκθέτες

$$\chi \sim |t|^{-\gamma}, \quad \gamma = 7/4, \quad (14.3)$$

$$c \sim |t|^{-\alpha}, \quad \alpha = 0 \quad \text{και} \quad (14.4)$$

$$\langle m \rangle \sim |t|^\beta \quad t < 0, \quad \beta = 1/8. \quad (14.5)$$

Η παραπάνω συμπεριφορά παρουσιάζεται, μόνο όταν  $L \rightarrow \infty$ . Για πεπερασμένο πλέγμα όλες οι ποσότητες είναι αναλυτικές, αφού υπολογίζονται από την αναλυτική συνάρτηση επιμερισμού  $Z$  που δίνεται από την (13.4). Στο πεπερασμένο πλέγμα όσο  $1 \ll \xi \ll L$ , το μοντέλο συμπεριφέρεται (προσεγγιστικά) όπως το άπειρο σύστημα ενώ καθώς  $\beta \approx \beta_c$  και το  $\xi \sim L$ , παρουσιάζονται φαινόμενα επίδρασης του πεπερασμένου μεγέθους (finite size effects). Οι διακυμάνσεις  $\chi, c$  στο πεπερασμένο πλέγμα παρουσιάζουν μέγιστο για μια ψευδοκρίσιμη θερμοκρασία  $\beta_c(L)$  η οποία φυσικά<sup>3</sup>

$$\lim_{L \rightarrow \infty} \beta_c(L) = \beta_c. \quad (14.6)$$

Θυμίζουμε στον αναγνώστη ότι για το πρότυπο Ising στο τετραγωνικό πλέγμα που δίνεται από την (13.14) έχουμε  $\beta_c = \log(1 + \sqrt{2})/2$ .

Για  $\beta = \beta_c(L)$ , λόγω της (14.2), έχουμε  $\xi(t, L) \sim L \Rightarrow |t| = |(\beta_c - \beta_c(L))/\beta_c| \sim L^{-1/\nu}$  οπότε οι σχέσεις (14.3)–(14.5) γίνονται

$$\chi \sim L^{\gamma/\nu}, \quad (14.7)$$

$$c \sim L^{\alpha/\nu} \quad \text{και} \quad (14.8)$$

$$m \sim L^{-\beta/\nu}. \quad (14.9)$$

Τονίζουμε ότι οι παραπάνω σχέσεις αφορούν όλη την ψευδοκρίσιμη περιοχή, οπότε στην πράξη μπορούμε να πάρουμε είτε τις τιμές των ποσοτήτων, όταν  $\beta = \beta_c(L)$  είτε όταν  $\beta = \beta_c$ <sup>4</sup>. Στις επόμενες παραγράφους θα

<sup>3</sup>Κάθε φυσική ποσότητα μπορεί να έχει (ελαφρά) διαφορετική ψευδοκρίσιμη θερμοκρασία οπότε μπορεί να γράφουμε  $\beta_c^x(L), \beta_c^e(L)$ .

<sup>4</sup>Στο όριο  $L \rightarrow \infty$  η διαφορά δεν έχει σημασία. Η ταχύτητα σύγκλισης στις ασυμπτωτικές αυτές σχέσεις μπορεί να διαφέρει.

δείξουμε πώς να υπολογίσουμε τους κρίσιμους εκθέτες από τις σχέσεις (14.3)–(14.5) και (14.7)–(14.9).

## 14.2 Κρίσιμη Επιβράδυνση

Η μελέτη των κρίσιμων εκθετών απαιτεί προσομοιώσεις ακριβείας, άρα και υψηλής στατιστικής. Ο αλγόριθμος Metropolis που αναπτύχθηκε στο προηγούμενο κεφάλαιο πάσχει από κρίσιμη επιβράδυνση (critical slowing down) στην κρίσιμη περιοχή αλλαγής φάσης και είναι αδύνατον να μελετήσουμε αρκετά μεγάλα πλέγματα για την εξαγωγή αποτελεσμάτων ακριβείας. Στο κεφάλαιο αυτό, θα αναλύσουμε τους λόγους εμφάνισης και τις δυσκολίες που παρουσιάζει το φαινόμενο αυτό στις προσομοιώσεις Monte Carlo. Θα παρουσιάσουμε τον αλγόριθμο του Wolff που σχεδόν εξαλείφει το φαινόμενο. Αλγόριθμοι σαν και αυτόν βασίζονται στη μερική κατανόηση της δυναμικής των μοντέλων που μελετούμε και έχουν ειδική εφαρμογή σε αντίθεση με τον αλγόριθμο Metropolis που εφαρμόζεται σε όλα τα συστήματα στατιστικής φυσικής και έχει γενική εφαρμογή.

Όπως συζητήσαμε στην ενότητα 13.5, η προσομοίωση του πρότυπου Ising στην κρίσιμη περιοχή αλλαγής φάσης με τον αλγόριθμο Metropolis παρουσιάζει μεγάλη αύξηση του χρόνου αυτοσυσχετισμού που δίνεται από τη σχέση βάθμισης

$$\tau \sim \xi^z. \quad (14.10)$$

Στην ψευδοκρίσιμη περιοχή το μήκος συσχετισμού γίνεται  $\xi \sim L$  και παίρνουμε τη σχέση (13.39)  $\tau \sim L^z$ . Όταν  $z > 0$ , έχουμε το φαινόμενο της κρίσιμης επιβράδυνσης στην προσομοίωση.

Εδώ αξίζει να αναφέρουμε ότι το φαινόμενο αυτό είναι που ουσιαστικά περιορίζει τη δυνατότητα που έχουμε να προσομοιώσουμε συστήματα με πολύ μεγάλο  $L$  από την άποψη του διαθέσιμου  $t_{\text{CPU}}$ <sup>5</sup>. Όταν μετράμε μια τοπική ποσότητα, όπως η μαγνήτιση ανά πλεγματοειδή θέση  $\langle m \rangle$ , η αύξηση του  $L$  αυξάνει τον χρόνο<sup>6</sup>  $t_{\text{CPU}} \sim L^d$ , αλλά ταυτόχρονα οι μετρήσεις που παίρνουμε για την  $\langle m \rangle$  αυξάνουν με τον ίδιο ρυθμό. Άρα, αν σε κάθε sweep του πλέγματος παίρναμε μία στατιστικά ανεξάρτητη διάταξη των σπιν, τότε δε θα είχαμε επιπλέον κόστος για μια δεδομένης ακρίβειας μέτρηση, αφού με την αύξηση του  $L$  το αυξημένο

<sup>5</sup>Φυσικά, το μέγεθος της μνήμης είναι ο άλλος σημαντικός παράγοντας που θα βάλει φραγμό στις προσομοιώσιμες τιμές του  $L$ .

<sup>6</sup> $d = 2$  στην περίπτωση του πρότυπου Ising στο επίπεδο, αλλά το επιχείρημα ισχύει και για περισσότερες διαστάσεις  $d > 2$ .

κόστος παραγωγής της διάταξης θα αντισταθμιζόταν από τα περισσότερα σπιν πάνω στα οποία θα μετρήσουμε την  $\langle m \rangle$  και θα είχαμε για τον χρόνο συλλογής δεδομένης στατιστικής για τη μαγνήτιση ανά πλεγματοειδή θέση  $t_{\text{CPU}}^{(m)} \sim L^0 = 1$ . Οπότε θα μας συνέφερε η μέτρηση στο μεγαλύτερο δυνατόν  $L$ , ώστε να ελαχιστοποιήσουμε τα φαινόμενα επίδρασης του πεπερασμένου μεγέθους του πλέγματος.

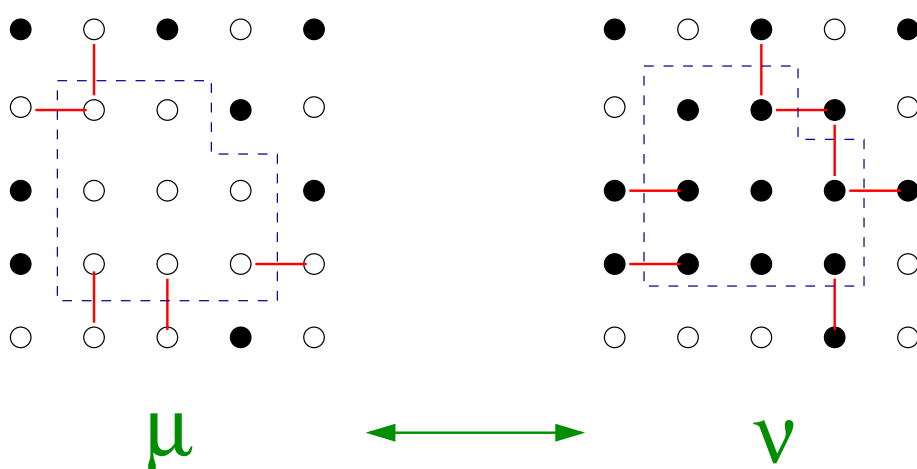
Η παρουσία κρίσιμης επιβράδυνσης, όμως, προσθέτει το κόστος παραγωγής ανεξάρτητων μετρήσεων και παίρνουμε  $t_{\text{CPU}}^{(m)} \sim L^z$  με αποτέλεσμα για κάποια τιμή του  $L$  να γίνεται απαγορευτικά ακριβό να προσομοιώσουμε μεγαλύτερο σύστημα. Στη συγκεκριμένη περίπτωση που μελετάμε, το πρότυπο Ising στο επίπεδο με τον αλγόριθμο Metropolis,  $z \approx 2.17$  και το πρόβλημα αυτό παρουσιάζεται πολύ γρήγορα. Είναι ιδιαίτερα ενδιαφέρον να αναζητηθούν αλγόριθμοι που να ελαχιστοποιούν το φαινόμενο και στην περίπτωση του πρότυπου Ising, αλλά και άλλων μοντέλων με σπιν η λύση είναι αρκετά απλή. Όπως θα δούμε όμως, η λύση προκύπτει από τη βαθύτερη κατανόηση της δυναμικής που προκαλεί το φαινόμενο της κρίσιμης επιβράδυνσης και αφορά μόνο τα πρότυπα εκείνα με παρόμοια δυναμική δομή<sup>7</sup>.

Ο λόγος που παρουσιάζεται το φαινόμενο της κρίσιμης επιβράδυνσης είναι η αύξηση του μήκους συσχετισμού  $\xi$  στην κρίσιμη περιοχή. Καθώς πλησιάζουμε την κρίσιμη θερμοκρασία  $\beta \rightarrow \beta_c^-$  από τη φάση αταξίας των σπιν, δημιουργούνται στο πλέγμα μεγάλες περιοχές όπου τα σπιν δείχνουν προς την ίδια κατεύθυνση. Η επικράτηση των περιοχών αυτών είναι αυτή που τελικά θα οδηγήσει το σύστημα στη φάση τάξης με μη μηδενική μαγνήτιση ανά πλεγματοειδή θέση στο θερμοδυναμικό όριο. Ο αλγόριθμος Metropolis επιχειρεί την αλλαγή των σπιν ένα-ένα, κάτι το οποίο είναι δύσκολο ειδικά για σπιν μέσα στις περιοχές αυτές. Για ένα σπιν με τέσσερις γείτονες που έχουν σπιν στην ίδια κατεύθυνση με αυτό, η αλλαγή γίνεται δεκτή με πιθανότητα  $\sim e^{-8\beta_c} \approx 0.029$ , η οποία είναι πολύ μικρή. Τα περισσότερα σπιν στα οποία η αλλαγή γίνεται δεκτή βρίσκονται στα όρια των περιοχών αυτών και η αλλαγή από μία διάταξη σε μια άλλη όπου οι περιοχές αυτές βρίσκονται σε ανεξάρτητες θέσεις και έχουν ανεξάρτητα σχήματα απαιτεί μεγάλο χρόνο: Μια περιοχή δημιουργείται/καταστρέφεται (στοχαστικά φυσικά) μετακινώντας το σύνορό της.

<sup>7</sup>Ενώ, αντίθετα, ο αλγόριθμος Metropolis έχει γενική εφαρμογή στα περισσότερα συστήματα της στατιστικής φυσικής.

## 14.3 Ο Αλγόριθμος του Wolff

Για τη βελτίωση της επίδρασης του φαινομένου της κρίσιμης επιβράδυνσης είναι ανάγκη να επινοηθούν αλγόριθμοι όπου θα μπορεί κανείς σε ένα βήμα να αλλάξει την τιμή των σπιν σε περιοχές συγκρίσιμες με τις μεγάλες περιοχές όμοιων σπιν. Οι περιοχές αυτές από κατάλληλα επιλεγμένα όμοια σπιν ονομάζονται τα clusters του αλγόριθμου. Το πρώτο βήμα έγινε με τον αλγόριθμο Swendsen-Wang [62], ενώ αργότερα παρουσιάστηκε και ο αλγόριθμος του Uli Wolff [63].



Σχήμα 14.1: Οι δύο διατάξεις των σπιν που προκύπτουν από την αλλαγή των σπιν ενός Wolff cluster. Φαίνονται οι δεσμοί που δημιουργούνται/καταστρέφονται κατά τη μετάβαση οι οποίοι βρίσκονται στο σύνορο του cluster.

Από την ιδέα που παρουσιάσαμε προκύπτει ότι η διαδικασία κατασκευής των clusters πρέπει να είναι στοχαστική και να εξαρτάται από τη θερμοκρασία. Σε ψηλές θερμοκρασίες  $\beta \ll \beta_c$  πρέπει να ευνοείται η κατασκευή μικρών clusters με μέγεθος ίσο με μερικές πλεγματικές σταθερές, ενώ σε χαμηλές θερμοκρασίες  $\beta \gg \beta_c$  να ευνοείται η κατασκευή μεγάλων clusters με μέγεθος  $\sim L$ .

Η βασική ιδέα στον αλγόριθμο του Wolff είναι να επιλέξουμε έναν σπιν-“σπόρο” (γεννήτορα, seed) και να κατασκευάσουμε ένα cluster γύρω από αυτόν. Σε κάθε βήμα προσθέτουμε μέλη στο cluster με πιθανότητα  $P_{\text{add}} = P_{\text{add}}(\beta)$ . Με κατάλληλα επιλεγμένη  $P_{\text{add}}(\beta)$  το cluster θα αναπτύσσεται με τον αναμενόμενο τρόπο και θα ικανοποιείται η συνθήκη λεπτομερούς ισορροπίας (12.59). Η σχέση φαίνεται γραφικά στο σχήμα 14.1. Στην κατάσταση  $\mu$  το cluster περιγράφεται από τη διακεκομμένη γραμμή. Η νέα κατάσταση  $\nu$  είναι αυτή όπου όλα τα σπιν του

cluster αντιστρέφονται, ενώ όλα τα σπιν στις πλεγματικές θέσεις εκτός του cluster μένουν τα ίδια.

Η κατάλληλη επιλογή της  $P_{\text{add}}$  θα μας δώσει τη σχέση (12.60)

$$\frac{P(\mu \rightarrow \nu)}{P(\nu \rightarrow \mu)} = e^{-\beta(E_\nu - E_\mu)}, \quad (14.11)$$

και η συζήτηση που θα ακολουθήσει βασίζεται στην πολύ απλή απόδειξη που βρίσκεται στο βιβλίο των Newman και Barkema [4]. Η κρίσιμη παρατήρηση είναι πως η μεταβολή της ενέργειας στο δεξί μέλος της (14.11) οφείλεται στο σπάσιμο/δημιουργία δεσμών στο σύνορο του cluster, αφού στο εσωτερικό του δεν μεταβάλλονται οι δεσμοί από όμοια σπιν, ενώ εκτός του cluster δεν επέρχεται καμιά μεταβολή. Αυτό μπορείτε να το δείτε στο απλό παράδειγμα του σχήματος 14.1. Με κατάλληλη επιλογή πιθανότητας επιλογής  $g(\mu \rightarrow \nu)$  της κατάστασης  $\nu$  και του λόγου αποδοχής  $A(\mu \rightarrow \nu)$ , έτσι ώστε

$$P(\mu \rightarrow \nu) = g(\mu \rightarrow \nu) A(\mu \rightarrow \nu), \quad (14.12)$$

θα πετύχουμε να ισχύει η (14.11) και να μεγιστοποιηθούν οι λόγοι αποδοχής. Μάλιστα, στην περίπτωση μας, θα βρούμε ότι  $A(\mu \rightarrow \nu) = 1$ !

Η πιθανότητα επιλογής  $g(\mu \rightarrow \nu)$  είναι η πιθανότητα να κατασκευάσουμε το συγκεκριμένο cluster και σπάει σε 3 όρους από ανεξάρτητες επιλογές:

$$g(\mu \rightarrow \nu) = p_{\text{seed}} \times p_{\text{yes}}^{\text{int}} \times p_{\text{no}}^{\text{border}} \quad (14.13)$$

Ο πρώτος όρος είναι η πιθανότητα να ξεκινήσουμε το cluster από το συγκεκριμένο seed. Επιλέγοντας με ομοιόμορφη πιθανότητα μια πλεγματική θέση έχουμε

$$p_{\text{seed}} = \frac{1}{N}. \quad (14.14)$$

Στη συνέχεια, το cluster αναπτύσσεται γύρω από τον γεννήτορά του. Ο δεύτερος όρος  $p_{\text{yes}}^{\text{int}}$  είναι η πιθανότητα να ενσωματώσουμε όλα τα μέλη του cluster. Η πιθανότητα αυτή είναι πολύπλοκη σχέση του μεγέθους και σχήματος του cluster, αλλά ευτυχώς δεν είναι σημαντική στον υπολογισμό μας. Ο λόγος είναι ότι κατά τη μετάβαση  $\nu \rightarrow \mu$  ο αντίστοιχος όρος είναι ακριβώς ο ίδιος, αφού τα δύο cluster είναι ακριβώς τα ίδια (έχουν μόνο αντίθετα σπιν)!

$$p_{\text{yes}}^{\text{int}}(\mu \rightarrow \nu) = p_{\text{yes}}^{\text{int}}(\nu \rightarrow \mu) \equiv C_{\mu\nu}. \quad (14.15)$$

Ο τρίτος όρος είναι ο πιο ενδιαφέρων. Το cluster σταματάει την ανάπτυξη του όταν στις θέσεις πάνω στο σύνορο του πούμε “όχι” στην ενσωμάτωση των γειτονικών όμοιων σπιν (τα αντίθετα σπιν, προφανώς,



δεν ενσωματώνονται). Αν  $P_{\text{add}}$  είναι η πιθανότητα να πούμε “ναι” στην ενσωμάτωση, η πιθανότητα να πούμε “όχι” είναι  $1 - P_{\text{add}}$ . Ας υποθέσουμε πως τα γειτονικά με το cluster όμοια σπιν στην κατάσταση  $\mu$  είναι  $m$ , ενώ στην κατάσταση  $\nu$  είναι  $n$ . Λ.χ. στο σχήμα 14.1  $m = 5$  και  $n = 7$ . Άρα, η πιθανότητα να σταματήσουμε το cluster της κατάστασης  $\mu$  στο σύνορό που φαίνεται στο σχήμα είναι να πούμε “όχι”  $m$  φορές, κάτι που γίνεται με πιθανότητα  $(1 - P_{\text{add}})^m$ :

$$p_{\text{no}}^{\text{border}}(\mu \rightarrow \nu) = (1 - P_{\text{add}})^m. \quad (14.16)$$

Ανάλογα, το cluster στην κατάσταση  $\nu$  σταματάει στο ίδιο σύνορο με πιθανότητα

$$p_{\text{no}}^{\text{border}}(\nu \rightarrow \mu) = (1 - P_{\text{add}})^n. \quad (14.17)$$

Τελικά, παίρνουμε

$$\frac{P(\mu \rightarrow \nu)}{P(\nu \rightarrow \mu)} = \frac{\frac{1}{N} C_{\mu\nu} (1 - P_{\text{add}})^m A(\mu \rightarrow \nu)}{\frac{1}{N} C_{\mu\nu} (1 - P_{\text{add}})^n A(\nu \rightarrow \mu)} = e^{-\beta(E_\nu - E_\mu)}. \quad (14.18)$$

Το δεξί μέλος της παραπάνω εξίσωσης εξαρτάται πάλι μόνο από τους δεσμούς στο σύνορο του cluster. Οι εσωτερικοί δεσμοί δε συνεισφέρουν, οπότε η μεταβολή στην ενέργεια οφείλεται μόνο στη δημιουργία/καταστροφή των δεσμών στο σύνορο. Κάθε δεσμός που δημιουργείται στο σύνορο κατά τη μετάβαση  $\mu \rightarrow \nu$  μειώνει την ενέργεια κατά 2, ενώ κάθε δεσμός που σπάει την αυξάνει κατά 2:

$$E_\nu - E_\mu = (-2n) - (-2m) = 2(m - n), \quad (14.19)$$

οπότε παίρνουμε

$$(1 - P_{\text{add}})^{m-n} \frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = e^{-2\beta(m-n)} \Rightarrow \frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = [e^{2\beta}(1 - P_{\text{add}})]^{n-m}. \quad (14.20)$$

Από την παραπάνω σχέση βλέπουμε ότι αν επιλέξουμε

$$1 - P_{\text{add}} = e^{-2\beta} \Rightarrow P_{\text{add}} = 1 - e^{-2\beta}, \quad (14.21)$$

μπορούμε επίσης να επιλέξουμε

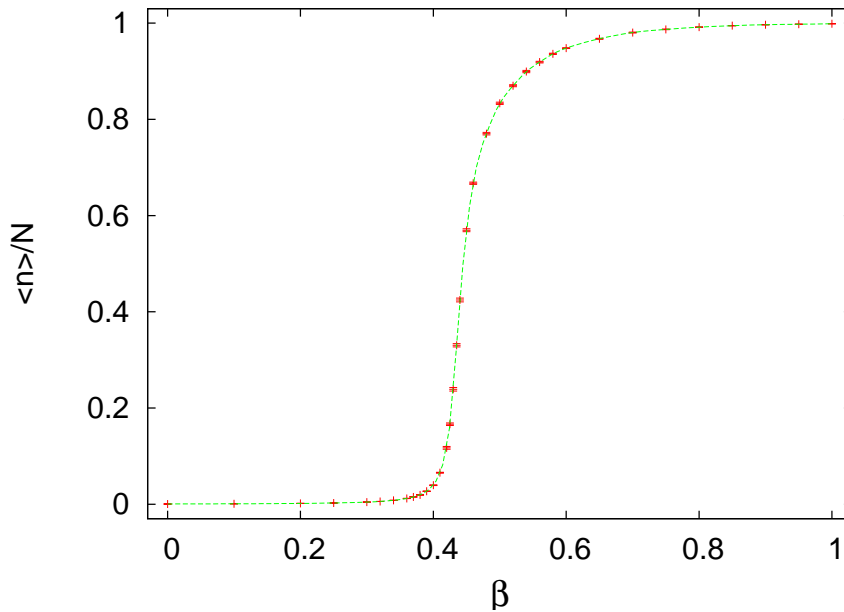
$$A(\mu \rightarrow \nu) = A(\nu \rightarrow \mu) = 1! \quad (14.22)$$

Άρα, μπορούμε να έχουμε τη συνθήκη (14.11) να ισχύει φτιάχνοντας το cluster με την  $P_{\text{add}}$  της (14.21), κάνοντας πάντα τη νέα κατάσταση που προκύπτει δεκτή στο επόμενο βήμα του Μόντε Κάρλο.

Συνοψίζοντας, ένα βήμα του αλγόριθμου Wolff γίνεται ως εξής:

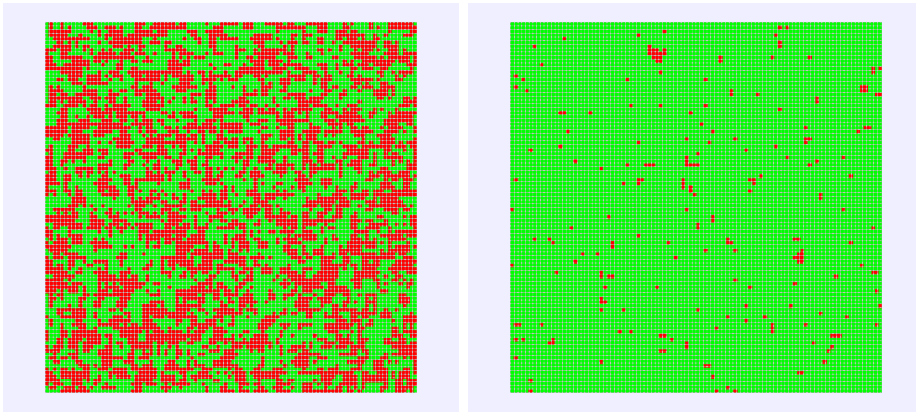
1. Επιλέγουμε με ομοιόμορφη πιθανότητα  $p_{\text{seed}} = \frac{1}{N}$  μία πλεγματική θέση που θα είναι ο γεννήτορας (seed) του cluster. Είναι το πρώτο νέο μέλος του cluster.
2. Επαγωγικά: Για κάθε νέο μέλος του cluster επισκεπτόμαστε τους πλησιέστερους γείτονές του που δεν ανήκουν ήδη στο cluster. Αν το σπιν τους είναι ίδιο με αυτό των μελών του cluster, τους προσθέτουμε στα “νέα μέλη” με πιθανότητα  $P_{\text{add}}$ . Το αρχικό σπιν είναι πια “παλιό μέλος”.
3. Όταν τελειώσουν τα “νέα μέλη”, η διαδικασία ανάπτυξης του cluster σταματάει.
4. Αλλάζουμε την τιμή του σπιν κάθε μέλους του cluster.

Ο αλγόριθμος είναι εργοδικός, αφού από μια οποιαδήποτε κατάσταση μπορούμε να πάμε σε μία άλλη με μια σειρά από clusters μεγέθους 1 (ισοδύναμο με single-flip). Η πιθανότητα  $P_{\text{add}}$  εξαρτάται από τη θερ-



Σχήμα 14.2: Η εξάρτηση του μεγέθους των Wolff clusters από τη θερμοκρασία. Φαίνεται το μέσο μέγεθος ενός Wolff cluster σε σχέση με το μέγεθος του πλέγματος. Σε μεγάλες θερμοκρασίες  $\beta \ll \beta_c$  αυτό είναι  $\sim 1/N$ , ενώ όταν η θερμοκρασία γίνεται πολύ μικρή  $\beta \gg \beta_c$  γίνεται  $\sim 1$ . Τα δεδομένα είναι για το πρότυπο Ising στο τετραγωνικό πλέγμα για  $L = 40$ .

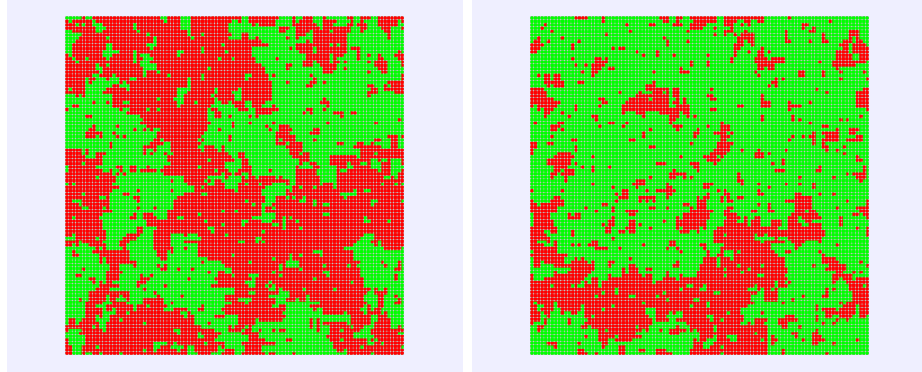
μοκρασία  $\beta$  και είναι μικρή για μεγάλες θερμοκρασίες ( $\beta \ll \beta_c$ ) και σχεδόν 1, όταν η θερμοκρασία τείνει στο μηδέν ( $\beta \gg \beta_c$ ). Άρα, στην πρώτη περίπτωση ο αλγόριθμος φτιάχνει πολύ μικρά clusters (μεγέθους 1 για  $\beta = 0$ ), ενώ μεγάλα στη δεύτερη. Άρα, στις μεγάλες θερμοκρασίες έχουμε σχεδόν τυχαία spin flips όπως και στον αλγόριθμο Metropolis, ενώ στις μικρές θερμοκρασίες (μεγάλο  $\beta$ ) ο αλγόριθμος έχει μεγάλη πιθανότητα να αλλάξει το cluster που επικρατεί στο πλέγμα. Αυτό φαίνεται καθαρά στο σχήμα 14.2 όπου υπολογίζουμε το ποσοστό του πλέγματος που καλύπτεται από το τυπικό Wolff cluster  $\langle n \rangle / N$  ως συνάρτηση της θερμοκρασίας. Για μικρά  $\beta$ ,  $\langle n \rangle / N \rightarrow 1/N$ , ενώ για μεγάλα  $\langle n \rangle / N \rightarrow 1$ .



Σχήμα 14.3: Μία τυπική διάταξη των σπιν στη φάση αταξίας (αριστερά,  $\beta = 0.25$ ) και τάξης (δεξιά,  $\beta = 0.5556$ ) για το πρότυπο Ising στο τετραγωνικό πλέγμα για  $L = 100$ .

Στο σχήμα 14.3 βλέπουμε τυπικές διατάξεις των σπιν σε μικρές και μεγάλες θερμοκρασίες. Για μικρά  $\beta$ , ο αλγόριθμος Wolff επιλέγει τυχαία πλεγματική θέση, φτιάχνει (συνήθως) ένα μικρό cluster και του αλλάζει την τιμή των σπιν. Ο αλγόριθμος Metropolis επιλέγει τυχαία πλεγματική θέση και με μεγάλη πιθανότητα της αλλάζει το σπιν. Το αποτέλεσμα είναι η (σχεδόν) τυχαία κατανομή των σπιν όπως αναμένεται στη φάση αταξίας. Για μεγάλα  $\beta$ , οι διατάξεις των σπιν “παγώνουν” με μικρές θερμικές διακυμάνσεις σκορπισμένες στο πλέγμα. Ο αλγόριθμός του Wolff τυπικά επιλέγει για seed μία πλεγματική θέση στο cluster που επικρατεί στο πλέγμα και το νέο cluster είναι με μεγάλη πιθανότητα σχεδόν όλο το προηγούμενο. Με την αλλαγή της τιμής των σπιν, όλες οι θερμικές διακυμάνσεις των αντίθετων σπιν ενσωματώνονται στο νέο cluster. Νέες θερμικές διακυμάνσεις προκύπτουν από τα (λίγα και

σκόρπια) spin που δεν ενσωματώθηκαν στο νέο cluster. Ο αλγόριθμος Metropolis επιλέγει τυχαίες θέσεις στο πλέγμα και αλλάζει με μικρή πιθανότητα τα spin μέσα στο μεγάλο cluster, ενώ εξαφανίζει με πιθανότητα 1 τις θερμικές διακυμάνσεις. Οι δύο αλγόριθμοι λειτουργούν με όμοια αποτελεσματικότητα.



Σχήμα 14.4: Δύο τυπικές διατάξεις των spin στην (ψευδο)κρίσιμη περιοχή ( $\beta = 0.4348$ ) για το πρότυπο Ising στο τετραγωνικό πλέγμα για  $L = 100$ . Οι δύο διατάξεις “απέχουν” 5000 sweeps χρησιμοποιώντας τον αλγόριθμο Metropolis.

Στο σχήμα 14.3 βλέπουμε τυπικές διατάξεις των spin στην ψευδοκρίσιμη περιοχή. Αυτές κυριαρχούνται από cluster μεγάλου μεγέθους με τυχαίο σχήμα, μέγεθος και κατανομή θέσης στο πλέγμα. Ο αλγόριθμος Wolff φτιάχνει με μεγάλη πιθανότητα μεγάλα clusters, με αποτέλεσμα μεγάλα cluster να καταστρέφονται και να δημιουργούνται σε χρόνο ανάλογο με το μέγεθος των cluster (στο σχήμα 14.2 βλέπουμε ότι  $\langle n \rangle / N \approx 0.5$ ). Αντιθέτως, ο αλγόριθμος Metropolis αλλάζει τα cluster κυρίως μεταβάλλοντας το σχήμα τους στο σύνορο, με αποτέλεσμα να χρειάζονται πολλά sweeps για την καταστροφή μιας διάταξης από spin clusters και τη δημιουργία μιας νέας και στατιστικά ανεξάρτητης από την προηγούμενη. Αναμένεται λοιπόν δραστική μείωση του χρόνου αυτοσυσχετισμού στην ψευδοκρίσιμη περιοχή με τη χρήση του αλγόριθμου του Wolff.

Η μέση τιμή του μεγέθους των Wolff clusters είναι μια δυναμική ποσότητα. Για να το δούμε αυτό καθαρά θα δείξουμε ότι στη φάση αταξίας ( $\beta < \beta_c$ ) ισχύει

$$\chi = \beta \langle n \rangle. \quad (14.23)$$

Παρουσιάζουμε την απόδειξη από τους Newmann και Barkema [4]: Αντί να κατασκευάσουμε ένα Wolff cluster, επισκεπτόμαστε κάθε ζευγάρι από πλεγματικές θέσεις και αν έχουν το ίδιο spin, τότε δημιουρ-

γούμε μεταξύ τους έναν δεσμό με πιθανότητα  $P_{\text{add}} = 1 - e^{-2\beta}$ . Όταν τελειώσει η διαδικασία, το πλέγμα έχει χωριστεί σε  $N_c$  Wolff<sup>8</sup> clusters κάθε ένα από τα οποία έχει  $n_i$  μέλη και σπιν  $S_i$ . Επιλέγουμε τυχαία μία πλεγματική θέση και αλλάζουμε το σπιν του cluster στο οποίο ανήκει. Καταστρέφουμε τους δεσμούς και επαναλαμβάνουμε τη διαδικασία<sup>9</sup>. Η συνολική μαγνήτιση τότε είναι:

$$M = \sum_{i=1}^{N_c} S_i n_i, \quad (14.24)$$

και

$$\langle M^2 \rangle = \left\langle \left( \sum_{i=1}^{N_c} S_i n_i \right) \left( \sum_{j=1}^{N_c} S_j n_j \right) \right\rangle = \left\langle \sum_{i \neq j} S_i S_j n_i n_j \right\rangle + \left\langle \sum_i S_i^2 n_i^2 \right\rangle. \quad (14.25)$$

Οι τιμές  $S_i = \pm 1$  είναι ισοπίθανες λόγω της συμμετρίας του μοντέλου, άρα ο πρώτος όρος είναι μηδέν. Επειδή  $S_i^2 = 1$ , παίρνουμε τελικά

$$\langle m^2 \rangle = \frac{1}{N^2} \langle M^2 \rangle = \frac{1}{N^2} \left\langle \sum_i n_i^2 \right\rangle. \quad (14.26)$$

Στον αλγόριθμο Wolff η κατασκευή ενός cluster ισοδυναμεί με την επιλογή ενός από τα cluster που φτιάξαμε παραπάνω. Σύμφωνα με τα παραπάνω (τυχαία επιλογή πλεγματικής θέσης και αλλαγή του σπιν στο cluster που ανήκει), η πιθανότητα επιλογής του cluster  $i$  είναι

$$p_i = \frac{n_i}{N}, \quad (14.27)$$

άρα, η μέση τιμή των Wolff clusters θα είναι

$$\langle n \rangle = \left\langle \sum_i p_i n_i \right\rangle = \left\langle \sum_i \frac{n_i}{N} n_i \right\rangle = N \langle m^2 \rangle. \quad (14.28)$$

Χρησιμοποιώντας τη Σχέση (14.26) και το γεγονός ότι για  $\beta < \beta_c$  έχουμε  $\langle m \rangle = 0$ <sup>10</sup>, παίρνουμε τελικά

$$\chi = \beta N (\langle m^2 \rangle - \langle m \rangle^2) = \beta \langle n \rangle. \quad (14.29)$$

<sup>8</sup>Αυτά είναι πραγματικά Wolff clusters, αφού οι δεσμοί ενεργοποιήθηκαν με τη σωστή πιθανότητα. Υπάρχουν φυσικά και απομονωμένες πλεγματικές θέσεις, χωρίς ενεργοποιημένους δεσμούς. Αυτές αποτελούν Wolff clusters μεγέθους 1.

<sup>9</sup>Αυτό είναι ισοδύναμο ως προς το αποτέλεσμα με τον αλγόριθμο Wolff, φυσικά όχι με τον πιο ... αποδοτικό τρόπο.

<sup>10</sup>Αυτό ισχύει στο θερμοδυναμικό όριο, για πεπερασμένο πλέγμα οι δύο ποσότητες θα διαφέρουν κατά τον παράγοντα  $\beta N \langle m \rangle^2 > 0$ , η διαφορά αυτή, όμως, εξαφανίζεται, καθώς το μέγεθος του πλέγματος απειρίζεται.

## 14.4 Σχεδιασμός Κώδικα

Στην παράγραφο αυτή θα σχεδιάσουμε την υλοποίηση του αλγόριθμου Wolff όπως αυτός παρουσιάστηκε στη σελίδα 609. Για να αναπτύξουμε ένα cluster γύρω από το seed που επιλέγεται, θα χρειαστεί να χρησιμοποιηθεί μια βοηθητική θέση στη μνήμη (buffer) από την οποία θα ανασύρουμε τα νέα μέλη του cluster για να εξετάσουμε αν θα προσθέσουμε στο cluster τους πλησιέστερους γείτονές τους. Κάθε πλεγματική θέση που θα προσθέτουμε θα αποθηκεύεται στο buffer για να εξεταστεί με τη σειρά του αργότερα.

Υπάρχουν δύο τέτοιες δομές δεδομένων που αρκούν για τη δουλειά που θέλουμε να κάνουμε. Η πρώτη είναι το stack (ή LIFO: last in – first out) και η δεύτερη το queue (ή FIFO: first in – first out). Και οι δύο είναι arrays στα οποία η διαφορά είναι το πώς ανασύρουμε τα δεδομένα από αυτά. Αυτό γίνεται ακριβώς όπως λέει και το όνομά τους. Στο stack, όταν ανασύρουμε ένα στοιχείο, παίρνουμε αυτό που είχαμε βάλει τελευταίο. Στο queue, όταν ανασύρουμε ένα στοιχείο, παίρνουμε αυτό που είχαμε βάλει πρώτο.

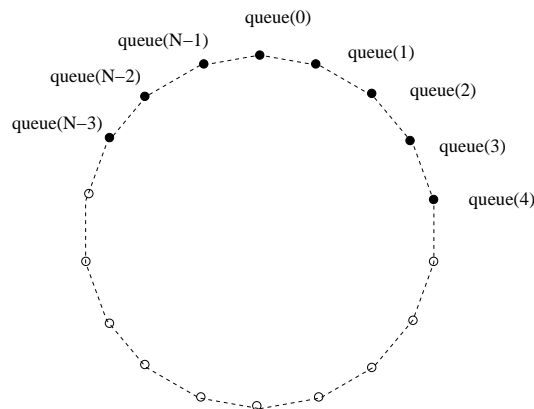
Το stack υλοποιείται από ένα μονοδιάστατο array  $stack(0:N-1)$  στο οποίο “σπρώχνουμε” (push) μια νέα τιμή και “αντλούμε” (pop) μια τιμή που θέλουμε να χρησιμοποιήσουμε. Για τη χρήση αυτή χρειαζόμαστε έναν ακέραιο  $m$  ίσο με τα ενεργά στοιχεία στο stack. Το στοιχείο που θα αντλήσουμε ανά πάσα στιγμή είναι το  $stack(m-1)$ . Για να σπρώξουμε ένα στοιχείο  $e$  στο stack ακολουθούμε τα εξής βήματα:

1. Ελέγχουμε αν το stack έχει άδειες θέσεις ( $m < N$ ).
2. Θέτουμε  $stack(m) = e$ .
3. Αυξάνουμε το  $m$  κατά 1.

Αντίθετα, για να τραβήξουμε μια τιμή και να την αποθηκεύσουμε στη μεταβλητή  $e$ , κάνουμε τα εξής:

1. Ελέγχουμε αν το stack περιέχει στοιχεία ( $m > 0$ ).
2. Μειώνουμε το  $m$  κατά 1.
3. Θέτουμε  $e = stack(m)$ .

Η υλοποίηση του queue γίνεται διαφορετικά. Η τοπολογία της διάταξης των στοιχείων είναι κυκλική αντί για γραμμική, όπως φαίνεται στο σχήμα 14.5. Χρησιμοποιούμε ένα array  $queue(0:N-1)$  και δύο ακέραιους  $m$ ,  $n$  να δείχνουν την αρχή και τέλος του buffer. Η αρχή των δεδομένων



Σχήμα 14.5: Η τοπολογία των δεδομένων στο queue. Στο εικονιζόμενο array έχουμε 8 στοιχεία αποθηκευμένα στο  $\text{queue}(N-3) \dots \text{queue}(4)$ . Έχουμε  $m=5$ ,  $n=N-3$ ,  $m-n = 8 \bmod N$ . Ένα στοιχείο προστίθεται στο  $\text{queue}(m)=\text{queue}(5)$  και ένα στοιχείο ανασύρεται διαβάζοντας το  $\text{queue}(n)=\text{queue}(N-3)$ .

είναι το στοιχείο  $\text{queue}(m-1)$ , ενώ το τέλος το  $\text{queue}(n)$ . Όταν το queue είναι άδειο  $m=n$  ενώ το ίδιο ισχύει και όταν είναι γεμάτο. Άρα, θα χρειαζόμαστε και μια μεταβλητή flag στην οποία αρχικά θα δώσουμε την τιμή 0 (άδειο queue)<sup>11</sup>. Ο αριθμός  $(m-n) \bmod N$  είναι ο αριθμός των αποθηκευμένων στοιχείων<sup>12</sup>. Όταν το queue περιέχει δεδομένα θα παίρνει την τιμή 1. Άρα, για να προσθέσουμε ένα στοιχείο  $e$  στο queue ακολουθούμε τα εξής βήματα:

1. Ελέγχουμε αν το stack είναι γεμάτο ( $m=n$  και  $\text{flag}=1$ ).
2. Θέτουμε  $\text{flag}=1$ .
3. Θέτουμε  $\text{queue}(m) = e$ .
4. Αυξάνουμε το  $m$  κατά  $1 \bmod N$ .

Για να τραβήξουμε μια τιμή και να την αποθηκεύσουμε στη μεταβλητή  $e$  κάνουμε τα εξής:

1. Ελέγχουμε αν το stack είναι άδειο ( $m=n$  και  $\text{flag}=0$ ).
2. Θέτουμε  $e = \text{queue}(n)$ .
3. Αυξάνουμε το  $n$  κατά  $1 \bmod N$ .

<sup>11</sup> Αν επιλέξουμε να αποθηκεύουμε το πολύ  $N-1$  στοιχεία στο  $\text{queue}(N)$ , τότε ο αλγόριθμος απλοποιείται (άσκηση).

<sup>12</sup> Εκτός αν  $m=n$ , οπότε είναι 0 ή  $N$  ανάλογα με την τιμή της flag.

4. Αν  $m=n$  θέτουμε  $\text{flag}=0$ .

Συνοψίζοντας, η αλγοριθμική διαδικασία κατασκευής ενός Wolff cluster στην περίπτωση του πρότυπου Ising είναι η εξής:

1. Επιλέγουμε ένα seed από το πλέγμα με πιθανότητα  $1/N$ .
2. Ελέγχουμε τους πλησιέστερους γείτονές του. Αν έχουν το ίδιο σπιν, τους προσθέτουμε στο cluster με πιθανότητα  $P_{\text{add}} = 1 - e^{-2\beta}$ . Τα σπιν που προσθέτουμε (“νέα μέλη”) αποθηκεύονται στο array  $\text{stack}(0:N-1)$  σύμφωνα με τα παραπάνω.
3. Ανασύρουμε μία πλεγματική θέση από το  $\text{stack}(0:N-1)$ . Αν είναι άδειο, προχωρούμε στο επόμενο βήμα. Ελέγχουμε τους πλησιέστερους γείτονές της. Αν δεν είναι ήδη στο cluster και έχουν το ίδιο σπιν, τους προσθέτουμε στο cluster με πιθανότητα  $P_{\text{add}}$ .
4. Σημειώνουμε το μέγεθος του cluster και αλλάζουμε το σπιν του.

Η επιλογή  $\text{stack}$  ή  $\text{queue}$  δεν έχει ουσιαστική διαφορά. Τα αποτελέσματα είναι ίδια και δεν υπάρχουν σημαντικές διαφορές στην ταχύτητα. Η διαφορά είναι μόνο στον τρόπο που αναπτύσσονται τα clusters (για  $\text{stack}$  το cluster αναπτύσσεται σπειροειδώς γύρω από το seed, ενώ για  $\text{queue}$  το cluster αναπτύσσεται πρώτα σε μία διεύθυνση και μετά στην άλλη). Ο προσεκτικός προγραμματιστής θα δοκιμάσει και τις δύο, ώστε να ελέγξει τα αποτελέσματά του για λάθη στον κώδικα. Πιο σημαντικός λόγος είναι να ελέγξει την ποιότητα της γεννήτριας τυχαίων αριθμών, αφού ο αλγόριθμος του Wolff είναι ιδιαίτερα ευαίσθητος στις ατέλειές τους.

#### 14.4.1 Ο Κώδικας

Στην παράγραφο αυτή θα παρουσιάσουμε και θα αναλύσουμε τον κώδικα που θα χρησιμοποιήσουμε στην προσομοίωση του πρότυπου Ising με τον αλγόριθμο Wolff.

Η καρδιά του αλγόριθμου βρίσκεται στην υπορουτίνα `wolff()` την οποία αποθηκεύουμε στο αρχείο `wolff.f90`. Κάθε κλήση της από το κυρίως πρόγραμμα κατασκευάζει ένα Wolff cluster, του αλλάζει το σπιν και σημειώνει το μέγεθός του.

Στο πρώτο μέρος της υπορουτίνας δημιουργούμε το buffer τύπου  $\text{stack}$  `stack(0:N-1)` το οποίο θα χρησιμοποιήσουμε στην κατασκευή του cluster. Αυτό το κάνουμε δυναμικά με τη χρήση της `ALLOCATE` και είμαστε προσεκτικοί στο τέλος να ελευθερώσουμε τη ζητούμενη μνήμη με



την DEALLOCATE (αλλιώς θα έχουμε “memory leak” και η μνήμη που ζητά το πρόγραμμα θα αυξηθεί ανεξέλεγκτα).

```
ALLOCATE(stack(0:N-1),STAT=chk)
if(chk>0) call locerr('allocation failure for stack in wolff')
....
DEALLOCATE(stack)!free memory of stack
```

Παρατηρούμε πως αν το σύστημα αδυνατεί να μας δώσει τη ζητούμενη μνήμη, τότε θέτει  $chk>0$  και η `locerr()` σταματάει το πρόγραμμα.

Στη συνέχεια, αρχικοποιούμε το cluster με την επιλογή του seed.

```
call ranlux(r,1)
cseed = INT(N*r)+1
stack(0) = cseed
nstack = 1 !the stack has 1 member, the seed
sold = s(cseed)
snew = -s(cseed) !the new spin value of the cluster
s(cseed) = snew !we flip all new members of cluster
ncluster = 1 !size of cluster=1
```

`cseed` είναι το seed και τοποθετείται αμέσως στο cluster [`stack(0) = cseed`]. Η μεταβλητή `nstack` είναι ο αριθμός των στοιχείων του stack και αρχικά τίθεται ίση με 1. Η `ncluster` μετράει τον αριθμό των πλεγματικών θέσεων στο cluster και τίθεται αρχικά ίση με 1. `sold=s(cseed)` είναι η παλιά τιμή του σπιν του cluster και `snew=-sold` η νέα. Η τιμή του σπιν του νέου μέλους του cluster αλλάζεται αμέσως (`s(cseed)=snew`)! Αυτό βοηθάει στην πιο αποτελεσματική εφαρμογή του αλγόριθμου. Ελέγχοντας αν το σπιν ενός πλησιέστερου γείτονα είναι ίσο με `sold`, ελέγχει ταυτόχρονα αν το σπιν του είναι ίδιο με του cluster και αν ήδη έχει ήδη μπει στο cluster από προηγούμενο έλεγχο!

Ο βρόχος επανάληψης πάνω στα νέα μέλη του cluster συνοψίζεται παρακάτω:

```
do while(nstack > 0)
!pull a site off the stack:
  nstack = nstack - 1; scluster = stack(nstack)
!check its four neighbours:
!-----scluster + XNN:
  nn = scluster + XNN; if(nn > N) nn = nn - N
  if(s(nn) == sold)then
    call ranlux(r,1)
    if(r<padd)then
      stack(nstack)=nn; nstack = nstack + 1
```

```

        s(nn)          =snew
        ncluster       =ncluster+1
    endif
endif
! ... check other 3 nearest neighbors ...
enddo

```

Ο βρόχος `do while(nstack > 0)` εκτελείται όσο `nstack>0`, δηλ. όσο το `stack` είναι γεμάτο και έχουμε νέα μέλη στο `cluster`. Η μεταβλητή `scluster` είναι η τρέχουσα πλεγματική θέση που τραβάμε από το `stack` για να κοιτάξουμε τους πλησιέστερους γείτονές της. Η γραμμή `nn = scluster + XNN`; `if(nn > N) nn = nn - N` επιλέγει τον πλησιέστερο γείτονα προς τα δεξιά και τον αποθηκεύει στη μεταβλητή `nn`. Αν το σπιν `s(nn)` του `nn` είναι ίσο με `sold`, τότε ο γείτονας αυτός έχει σπιν ίδιο με αυτό του `cluster` και δεν έχει ήδη μπει στο `cluster` (γιατί τότε θα είχαμε αλλάξει το σπιν του). Η μεταβλητή `padd` είναι ίση με  $P_{add}$  (αυτό τίθεται στην `init`) και αν `r<padd` (το οποίο συμβαίνει με πιθανότητα  $P_{add}$ ), προσθέτουμε τον `nn` στο `cluster`: Τον προσθέτουμε στο `stack`, αλλάζουμε το σπιν [`s(nn)=snew`] και αυξάνουμε το μέγεθος του `cluster` κατά 1. Επαναλαμβάνουμε ακριβώς τα ίδια και για τους υπόλοιπους πλησιέστερους γείτονες. Ολόκληρος ο κώδικας παρατίθεται παρακάτω για τη διευκόλυνση του αναγνώστη:

```

subroutine wolff
  use global_data
  implicit none
  integer          :: cseed,nstack,sold,snew,scluster,nn,chk
  integer          :: ncluster
  real(8)          :: r
  integer,allocatable :: stack(:)
!allocate stack memory:
  ALLOCATE(stack(0:N-1),STAT=chk)
  if(chk>0) call locerr('allocation failure for stack in wolff')
!choose a seed for the cluster, put it on the stack and flip it
  call ranlux(r,1)
  cseed    = INT(N*r)+1
  stack(0) = cseed
  nstack   = 1           !the stack has 1 member, the seed
  sold     = s(cseed)
  snew     = -s(cseed)   !the new spin value of the cluster
  s(cseed) = snew        !we flip all new members of cluster
  ncluster = 1           !size of cluster=1
!start the loop on spins in the stack:
  do while(nstack > 0)
!pull a site off the stack:

```

```

    nstack = nstack - 1; scluster = stack(nstack)
!check its four neighbours:
!-----scluster + XNN:
    nn      = scluster + XNN; if(nn > N) nn = nn - N
    if(s(nn) == sold)then
        call ranlux(r,1)
        if(r<padd)then
            stack(nstack)=nn; nstack = nstack + 1
            s(nn)          =snew
            ncluster       =ncluster+1
        endif
    endif
!-----scluster - XNN:
    nn      = scluster - XNN; if(nn < 1) nn = nn + N
    if(s(nn) == sold)then
        call ranlux(r,1)
        if(r<padd)then
            stack(nstack)=nn; nstack = nstack + 1
            s(nn)          =snew
            ncluster       =ncluster+1
        endif
    endif
!-----scluster + YNN:
    nn      = scluster + YNN; if(nn > N) nn = nn - N
    if(s(nn) == sold)then
        call ranlux(r,1)
        if(r<padd)then
            stack(nstack)=nn; nstack = nstack + 1
            s(nn)          =snew
            ncluster       =ncluster+1
        endif
    endif
!-----scluster - YNN:
    nn      = scluster - YNN; if(nn < 1) nn = nn + N
    if(s(nn) == sold)then
        call ranlux(r,1)
        if(r<padd)then
            stack(nstack)=nn; nstack = nstack + 1
            s(nn)          =snew
            ncluster       =ncluster+1
        endif
    endif
enddo !do while(nstack > 0)
print '(A,I14)', '#clu ', ncluster
!
DEALLOCATE(stack)!free memory of stack
!
end subroutine wolff

```

Για να συνδέσουμε τη συνάρτηση με το υπόλοιπο πρόγραμμα, ώστε να έχουμε κατασκευή ενός cluster ανά “sweep”<sup>13</sup>, μεταβάλλουμε τη main() ως εξής:

```
!===== main.f90 =====
program Ising2D
  use global_data
  implicit none
  integer :: isweep

  call init
  do isweep = 1, nsweep
    if(algorithm .eq. 1)then
      call wolff
    else
      call met
    endif
    call measure
  end do
  call endsim
end program Ising2D
```

Ορίσαμε τη (global) μεταβλητή algorithm, ώστε ο χρήστης να μπορεί να ελέγχει τον αλγόριθμο που θα χρησιμοποιήσει (Wolff ή Metropolis)<sup>14</sup>. Μένει να οριστεί η (global) μεταβλητή padd  $\equiv P_{\text{add}} = 1 - e^{-2\beta}$ , κάτι το οποίο κάνουμε στην init(). Στο αρχείο global\_data.f90 προσθέτουμε τις γραμμές

```
real(8)          :: padd
integer          :: algorithm
```

στο init.c

```
algorithm=0          !default is metropolis,1 is wolff
padd = 1.0D0 - exp(-2.0D0*beta)
```

ενώ μεταβάλλουμε το αρχείο options.f90, ώστε να παίρνει ένα διακόπτη -w όπου θα θέτει algorithm=1, ώστε το πρόγραμμα να λειτουργεί τον αλγόριθμο του Wolff:

<sup>13</sup>Προσοχή, ένα Metropolis sweep δεν είναι το ίδιο με ένα cluster update. Το μέσο μέγεθος κάθε cluster μεταβάλλεται με τη θερμοκρασία και είναι πολύ μικρό για μεγάλες θερμοκρασίες.

<sup>14</sup>Μεταβάλλετε την παραπάνω συνάρτηση, ώστε η wolff() να καλείται μέχρι φτιάξει clusters με συνολικό μέγεθος τουλάχιστον N.

```

.....
select case(getopt("-hL:b:s:S:n:r:uw"))
case( 'w' )
    algorithm = 1
.....

```

Προσθέτουμε τη σχετική πληροφορία στο μήνυμα βοήθειας usage και simmessage και ... είμαστε έτοιμοι! Για τη μεταγλώττιση χρησιμοποιούμε το Makefile

```

FC      = gfortran
OBJS    = global_data.o getopt.o main.o init.o met.o wolff.o \
          measure.o end.o options.o ranlux.o
FFLAGS  = -O2

is: $(OBJS)
    $(FC) $(FFLAGS) $^ -o $@

$(OBJS): global_data.f90
options.o: getopt.f90
%.o: %.f90
    $(FC) $(FFLAGS) -c -o $@ $<

```

και με τις εντολές

```

> make
> ./is -h
Usage: ./is [options]
    -L: Lattice length (N=L*L)
    -b: beta (options beta overrides the one in config)
    -s: start (0 cold, 1 hot, 2 old config.)
    -S: seed (options seed overrides the one in config)
    -n: number of sweeps and measurements of E and M
    -w: use wolff algorithm for the updates
    .....
> ./is -L 20 -b 0.44 -s 1 -S 34235322 -n 5000 -w > outL20b0.44

```

μεταγλωττίζουμε, υπενθυμίζουμε στον εαυτό μας τον τρόπο χρήσης του εκτελέσιμου αρχείου is και κάνουμε ένα δοκιμαστικό τρέξιμο για  $L = 40$ ,  $\beta = 0.44$  φτιάχνοντας 5000 clusters ξεκινώντας από μία “θερμή” διάταξη των σπιν αποθηκεύοντας τα δεδομένα στο αρχείο outL20b0.44.

## 14.5 Συλλογή Δεδομένων

Για να πάρουμε τα αποτελέσματά μας, επιλέγουμε αρχικά το μέγεθος ενός πλέγματος που θέλουμε να προσομοιώσουμε και στη συνέχεια, τις θερμοκρασίες που θα προσομοιώσουμε. Η δουλειά γίνεται γρήγορα βαρετή, επαναλαμβανόμενη, κουραστική και τότε είναι που υπεισέρχονται απρόσκλητοι επισκέπτες τα ... ζουζούνια! Η τεμπελιά στην περίπτωση αυτή είναι αρετή και αξίζει να μάθουμε μερικές τεχνικές που κάνουν τη ζωή μας ευκολότερη, αλλά και τα αποτελέσματά μας πιο αξιόπιστα (συνήθως...). Στη βοήθειά μας έρχεται ο φλοιός (Shell) και τα εργαλεία του. Ο φλοιός μας επιτρέπει να εκτελέσουμε μια σειρά εντολών με το ίδιο ακριβώς συντακτικό που θα χρησιμοποιούσαμε στη γραμμή εντολών γράφοντάς τις σε ένα απλό αρχείο κειμένου. Ένα απλό παράδειγμα είναι η σειρά εντολών που γράφουμε στο αρχείο run1:

```
# ##### run1 #####
./is -L 20 -b 0.10 -s 1 -n 5000 -w -S 3423 > outL20b0.10
./is -L 20 -b 0.20 -s 2 -n 5000 -w > outL20b0.20
./is -L 20 -b 0.30 -s 2 -n 5000 -w > outL20b0.30
./is -L 20 -b 0.40 -s 2 -n 5000 -w > outL20b0.40
./is -L 20 -b 0.42 -s 2 -n 5000 -w > outL20b0.42
./is -L 20 -b 0.44 -s 2 -n 5000 -w > outL20b0.44
./is -L 20 -b 0.46 -s 2 -n 5000 -w > outL20b0.46
./is -L 20 -b 0.48 -s 2 -n 5000 -w > outL20b0.48
./is -L 20 -b 0.50 -s 2 -n 5000 -w > outL20b0.50
./is -L 20 -b 0.60 -s 2 -n 5000 -w > outL20b0.60
./is -L 20 -b 0.70 -s 2 -n 5000 -w > outL20b0.70
```

Η πρώτη γραμμή που αρχίζει από τον χαρακτήρα # είναι σχόλιο και θα αγνοηθεί από τον φλοιό. Η δεύτερη ξεκινάει από την αρχή μια προσομοίωση από “καυτή” διάταξη σπιν (-s 1) για πλέγμα με  $L=20$  (-L 20) και θερμοκρασία  $\beta = 0.10$  (-b 0.10). Αρχικό seed καθορίζεται ο αριθμός 3423 (-S 3423) και κάνουμε μετρήσεις πάνω σε 5000 Wolff clusters (-n 5000 -w). Τα αποτελέσματα ανακατευθύνονται από το stdout στο αρχείο outL20b0.10 (> outL20b0.10).

Οι επόμενες δέκα γραμμές συνεχίζουν την προσομοίωση στις θερμοκρασίες  $\beta = 0.20 - 0.70$ . Κάθε προσομοίωση συνεχίζει από τη διάταξη σπιν που αποθήκευσε το πρόγραμμα στο αρχείο conf στην προηγούμενη προσομοίωση.

Μόλις φτιάξαμε ένα (πρωτόγονο είναι αλήθεια) σενάριο φλοιού (shell script)! Για να τρέξουν οι εντολές αυτές πρέπει να κάνουμε το αρχείο εκτελέσιμο, δίνοντάς του (την πρώτη φορά μόνο) άδεια πρόσβασης x

```
> chmod a+x run1
```

και στη συνέχεια, να το εκτελέσουμε καλώντας το

```
> ./run1
```

Δεν τα πήγαμε και άσχημα... Αλλά μπορούμε και καλύτερα! Αντί να βάλουμε μία γραμμή για κάθε προσομοίωση, μπορούμε να χρησιμοποιήσουμε τις δυνατότητες προγραμματισμού του φλοιού. Ας δούμε πώς. Στο αρχείο run2 γράφουμε:

```
#!/bin/tcsh -f
# ##### run2 #####
set L      = 20
set betas  = (0.10 0.20 0.30 0.40 0.42 0.44 0.46 0.48 0.50\
              0.60 0.70)
set start  = "-s 1 -S 3423"
set nsweeps = 5000

foreach beta ($betas)
  echo "L= $L beta= $beta"
  ./is -L $L -b $beta -n $nsweeps -w $start > outL${L}b${beta}
  set start = "-s 2"
end
```

Η πρώτη γραμμή καλεί τον φλοιό tcsh να ερμηνεύσει το σενάριο. Αυτό δε χρειαζόταν στο run1, γιατί οποιοσδήποτε φλοιός μπορούσε να ερμηνεύσει τις εντολές που είχαμε γράψει. Εδώ όμως βάζουμε συντακτικό που είναι κατανοητό μόνο από τον συγκεκριμένο φλοιό. Το ζεύγος χαρακτήρων #! πρέπει να είναι οι δύο πρώτοι χαρακτήρες του αρχείου.

Η δεύτερη γραμμή είναι σχόλιο.

Η τρίτη γραμμή ορίζει μία μεταβλητή φλοιού με το όνομα L. Η τιμή της μεταβλητής είναι "20" (string, όχι αριθμός ακέραιος) και έχουμε πρόσβαση σε αυτή βάζοντας ένα \$ μπροστά από το όνομά της. Δηλ. όπου γράφουμε στο σενάριο \$L (ή \${L}), ο φλοιός θα αντικαθιστά το string 20. Για παράδειγμα όταν γράφουμε στο σενάριο outL\${L}b, ο φλοιός θα φτιάχνει το string outL20b.

Η τέταρτη γραμμή ορίζει ένα array με το όνομα betas. Πρόσβαση στα στοιχεία του array παρέχεται από το συντακτικό \$betas[αριθμός], όπου "αριθμός" είναι το στοιχείο του array αρχίζοντας από το 1. Στο παραπάνω παράδειγμα \$betas[1] = 0.10, \$betas[2] = 0.20, ..., \$betas[11] = 0.70. Ειδικά η μεταβλητή \$#betas = 11 είναι ο αριθμός

των στοιχείων του array. Όταν γράφουμε \$betas, ο φλοιός αντικαθιστά όλες τις τιμές του array<sup>15</sup>.

Η πέμπτη γραμμή ορίζει τη μεταβλητή start να έχει την τιμή \$start ίση με το string "-s 1 -S 3423". Τα εισαγωγικά μπήκαν επειδή οι λέξεις χωρίζονται με κενά και ορίζουν έτσι επακριβώς το string. Οτιδήποτε γράφουμε από τον χαρακτήρα # και μετά είναι σχόλιο και ο φλοιός το αγνοεί.

Η εντολή foreach είναι ένας τρόπος να τρέξουμε έναν βρόχο στον φλοιό tcsh. Οι επαναλαμβανόμενες εντολές αρχίζουν από την επόμενη γραμμή και τελειώνουν στη γραμμή με την εντολή end. Οι επαναλήψεις γίνονται μία για κάθε τιμή του array που βάζουμε ανάμεσα στις παρενθέσεις. Εδώ θα γίνει μία φορά για κάθε τιμή της μεταβλητής \$betas. Κάθε φορά που γίνεται μια επανάληψη, η μεταβλητή βρόχου, το όνομα της οποίας μπαίνει αμέσως μετά τη λέξη foreach αντικαθίσταται από το επόμενο στοιχείο του array. Άρα, ο βρόχος που γράψαμε θα εκτελεστεί μία φορά για κάθε τιμή του betas και η μεταβλητή beta θα παίρνει διαδοχικά τις τιμές 0.10, 0.20, ... , 0.70. Οι επόμενες τρεις γραμμές είναι οι εντολές που θα εκτελεστούν έντεκα φορές. Η echo "αντηχεί" τα ορίσματα της και μας πληροφορεί για την τρέχουσα τιμή των παραμέτρων της προσομοίωσης που τρέχει (πολύ χρήσιμο, ειδικά αν οι προσομοιώσεις θέλουν αρκετό χρόνο). Η ./is τρέχει το πρόγραμμα, κάθε φορά με διαφορετική τιμή για τη beta. Προσέξτε πώς το αρχείο στο οποίο ανακατευθύνουμε τα δεδομένα αλλάζει όνομα, κάθε φορά που αλλάζει η τιμή της beta. Άρα, στο τέλος θα έχουμε τα δεδομένα μας στα αρχεία outL20b0.10, outL20b0.20, ... , outL20b0.70. Η τρίτη εντολή αλλάζει τον τρόπο που αρχίζει η προσομοίωση, από θερμή διάταξη σπιν στο να διαβάσει τη διάταξη από το αρχείο conf. Την πρώτη φορά που τρέχει ο βρόχος η τιμή της start είναι "-s 1 -S 3423" (θερμή διάταξη, seed ίσο με 3423), ενώ στις επόμενες η τιμή της είναι "-s 2" (old configuration).

Όταν λοιπόν θέλουμε να τρέξουμε για μια άλλη τιμή του L, δεν έχουμε παρά να αλλάξουμε την τιμή της αντίστοιχης μεταβλητής και να τρέξουμε πάλι το ίδιο script. Ή μήπως όχι... Χτύπησε πάλι η τεμπελιά:

```
#!/bin/tcsh -f

set Ls      = (10 20 40)
set betas   = (0.10 0.20 0.30 0.40 0.42 0.44 0.46 0.48 0.50 \
               0.60 0.70)
set nsweeps = 5000
```

<sup>15</sup> Δοκιμάστε την εντολή: echo \$betas[3] \$#betas \$betas.



```
foreach L ($Ls )
  set start = "-s 1 -S 3423"
  foreach beta ($betas)
    echo "L= $L beta= $beta"
    ./is -L $L -b $beta -n $nsweeps -w $start > outL${L}b${beta}
    set start = "-s 2"
  end
end
```

Δεν έχουμε παρά να προσθέσουμε έναν βρόχο για κάθε τιμή του  $L$  ορίζοντας τώρα τη μεταβλητή-array  $Ls$  και βάζοντας όσα  $L$  τραβάει η ψυχή μας... Προσέξτε μόνο πού μετακινήσαμε τον ορισμό της μεταβλητής  $start$  (γιατί;).

## 14.6 Ανάλυση Δεδομένων

Ο πρώτος έλεγχος στα δεδομένα μας είναι να εξετάσουμε γραφικά την εξέλιξή τους στον Μόντε Κάρλο χρόνο (time histories). Από εκεί θα κάνουμε μια πρώτη διάγνωση για τυχόν προβλήματα και λάθη και θα αποκτήσουμε μία αίσθηση για τους χρόνους εύρεσης της κατάστασης θερμικής ισορροπίας (thermalization) και των χρόνων αυτοσυσχετισμού. Από τα αρχεία που παράγαμε στην προηγούμενη παράγραφο μπορούμε να το καταφέρουμε εύκολα με τη βοήθεια του `gnuplot`. Για παράδειγμα, οι εντολές μέσα από το `gnuplot`<sup>16</sup>:

```
gnuplot> plot "<grep -v '#' outL40b0.44" u 1 \
with lines title "E"
gnuplot> plot "<grep -v '#' outL40b0.44" u (abs($2)) \
with lines title "|M|"
gnuplot> plot "<awk '/#clu/{ print $2}' outL40b0.44" u 1 \
with lines title "n"
```

μας δίνουν με απλό τρόπο τα time histories της ενέργειας, της απόλυτης τιμής της μαγνήτισης και του μεγέθους των cluster που κατασκευάζονται από τον αλγόριθμο του Wolff για  $L = 40$ ,  $\beta = 0.44$ .

Για να υπολογίσουμε τις τιμές της μέσης ενέργειας ανά link  $\langle e \rangle = \frac{1}{2N} \langle E \rangle$  και της (απόλυτης τιμής της) μαγνήτισης ανά πλεγματική θέση  $\langle m \rangle = \frac{1}{N} \langle M \rangle$  μπορούμε να χρησιμοποιήσουμε το πρόγραμμα για jack-knife που περιγράψαμε στο Παράρτημα 13.8. Αυτό θα το βρείτε και

<sup>16</sup>Σας θυμίζουμε πως η εντολή `plot` στο `gnuplot` δέχεται αντί για το όνομα ενός αρχείου δεδομένων, το `stdout` μιας εντολής `command` με το συντακτικό `plot "<command"`.

στα Tools στο συνοδευτικό λογισμικό στο αρχείο jack.f90. Το μεταγλωττίζουμε στο εκτελέσιμο αρχείο jack και αντιγράφουμε το αρχείο στον κατάλογο που εργαζόμαστε. Η μέση τιμή  $\langle e \rangle$  μπορεί τότε να υπολογιστεί με την εντολή

```
> grep -v # outL40b0.44 \
| awk -v L=40 'NR>500{print $1/(2*L*L)}' \
| ./jack
```

όπου με την awk ορίσαμε, χρησιμοποιώντας το option -v, τη μεταβλητή L=40 και τυπώσαμε την πρώτη στήλη διαιρεμένη με  $2N = 2L^2$ . Με τη συνθήκη NR>500 τυπώνουμε αφού περάσουν οι πρώτες 500 γραμμές, απορρίπτοντας τις πρώτες μετρήσεις για να είμαστε σίγουροι για την εύρεση της θερμικής ισορροπίας. Το αποτέλεσμα τυπώνεται στο stdout ως εξής:

```
# NDAT = 4500 data. JACK = 10 groups
# <o>, chi= (<o^2>-<o>^2)
# <o> +/- err          chi +/- err
-0.71091166666 0.0024162628283 0.0015719190590 7.819205433e-05
```

Οι γραμμές που αρχίζουν με # είναι σχόλια του προγράμματος και μας εξηγούν τα αποτελέσματα. Οι δύο πρώτοι αριθμοί είναι η  $\langle e \rangle$  και το σφάλμα της, ενώ ο τρίτος και ο τέταρτος οι διακυμάνσεις της ενέργειας  $\langle e^2 \rangle - \langle e \rangle^2$  και το σφάλμα της. Πολλαπλασιάζοντας την τελευταία με  $\beta^2 N$ , παίρνουμε την ειδική θερμότητα  $c$  και το σφάλμα της σύμφωνα με την εξίσωση (13.32). Μπορούμε να τυπώσουμε τις πληροφορίες αυτές βολικά για να τις αποθηκεύσουμε σε ένα αρχείο με την εντολή:

```
> set L = 40; set b = 0.44 ; \
grep -v # outL${L}b${b} | \
awk -v L=$L 'NR>500{print $1/(2*L*L)}' | \
./jack | grep -v # | \
awk -v L=$L -v b=$b \
'{print "e",L,b,$1,$2,b*b*L*L*$3,b*b*L*L*$4}'
```

Γιατί το κάναμε αυτό; Όχι λόγω ... μαζοχισμού! Οι παραπάνω εντολές δίνονται σε μία γραμμή στη γραμμή εντολών (φυσικά εμείς τη σπάμε για να φαίνεται στο κείμενο με τη γνωστή σύμβαση της \). Ανακαλώντας τις και αλλάζοντας μόνο την τιμή της L ή/και της b, παίρνουμε χωρίς καινούργιο κόπο τα αποτελέσματα για διαφορετική τιμή του L ή/και  $\beta$ . Επίσης, το shell script που θα φτιάξουμε σε λίγο θα μας φαίνεται λιγότερο ακατανόητο... Το αποτέλεσμα είναι

```
e 40 0.42 -0.619523333 0.00189807 0.311391 0.0228302
```

που δίνει  $\langle e \rangle = -0.6195(19)$  και  $c = 0.311(23)$ .

Η εντολή για τον υπολογισμό της μαγνήτισης δεν παρουσιάζει τώρα ιδιαίτερη δυσκολία. Το μόνο που έχουμε να κάνουμε είναι να υπολογίσουμε την απόλυτη τιμή της δεύτερης στήλης του αρχείου των δεδομένων για τις γραμμές που δεν αρχίζουν από #

```
> set L = 40 ; set b = 0.42 ; \
grep -v # outL${L}b${b} | \
awk -v L=$L 'NR>500{m=($2>0)?$2:-$2; print m/(L*L)}' | \
./jack | grep -v # | \
awk -v L=$L -v b=$b \
'{ print "m",L,b,$1,$2,b*L*L*$3,b*L*L*$4 }'
```

Η απόλυτη τιμή υπολογίζεται από την έκφραση  $(\$2>0)?\$2:-\$2$ , αποθηκεύεται στη μεταβλητή  $m$  και τυπώνεται αφού διαιρεθεί με  $N = L^2$ . Το αποτέλεσμα είναι

```
m 40 0.44 0.6250527778 0.00900370 21.8345 1.39975
```

που δίνει  $\langle m \rangle = 0.6251(90)$  και  $\chi = 21.8(14)$ .

Παρόμοια δίνουμε την εντολή για τον υπολογισμό του  $\langle n \rangle / N$ :

```
> set L = 40 ; set b = 0.44 ; \
grep '#clu' outL${L}b${b} | \
awk -v L=$L 'NR>500{ print $2/(L*L)}' | \
./jack | grep -v # | \
awk -v L=$L -v b=$b '{ print "n",L,b,$1,$2 }'
```

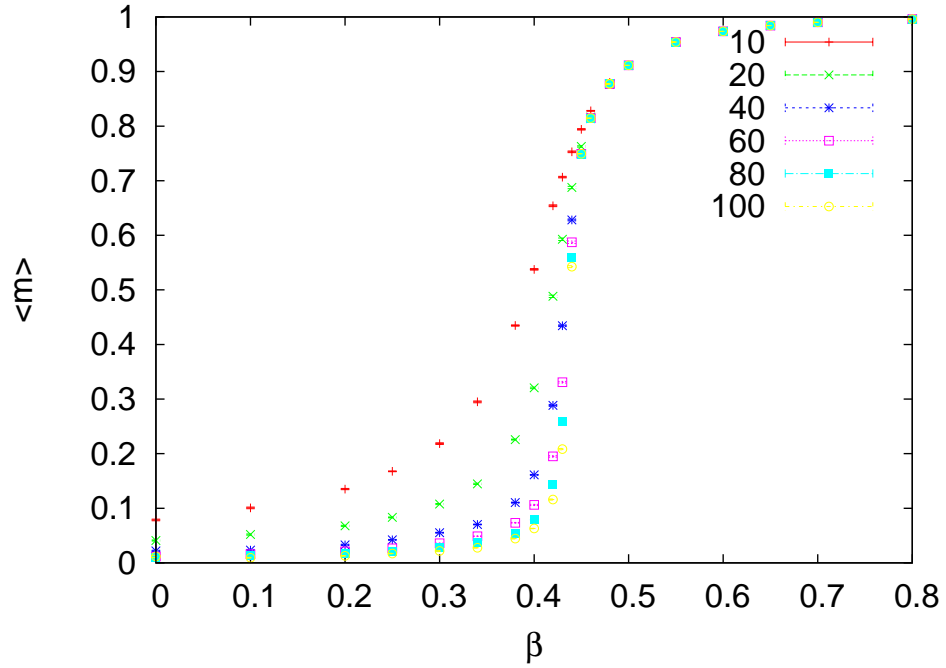
Το αποτέλεσμα είναι

```
n 40 0.44 0.4257476389 0.01302602
```

που δίνει  $\langle n \rangle / N = 0.426(13)$ .

Όσα είπαμε παραπάνω τα γράφουμε όλα μαζί στο script run3:

```
#!/bin/tcsh -f
set Ls      = (10 20 40 60 80 100)
set betas   = (0.00 0.10 0.20 0.25 0.30 0.34 0.38 \
               0.40 0.42 0.43 0.44 0.45 0.46 0.48 \
               0.48 0.50 0.55 0.60 0.65 0.70 0.80 )
```



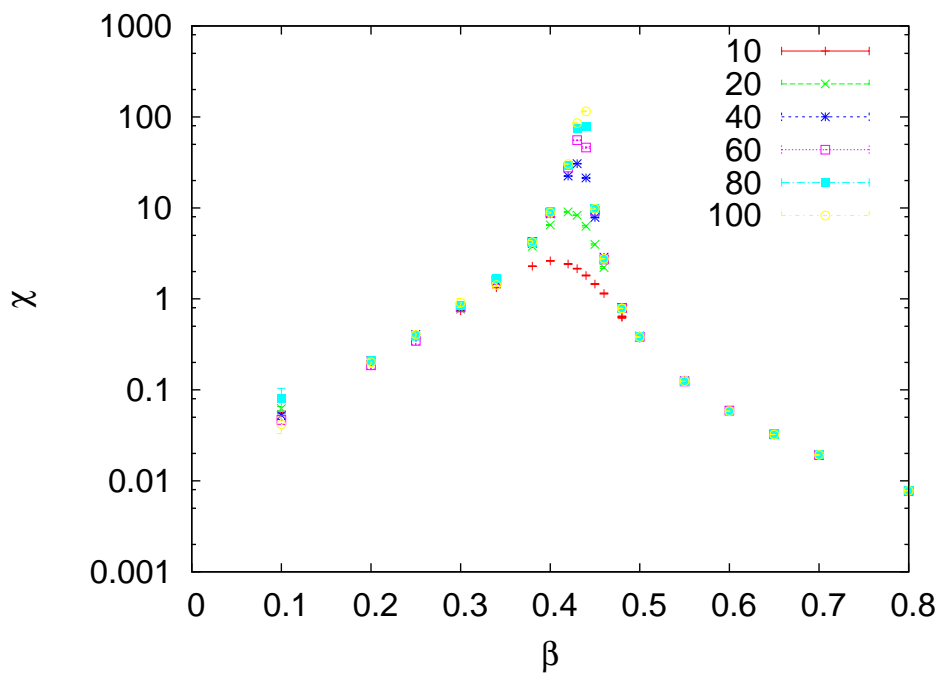
Σχήμα 14.6: Τα αποτελέσματα από τις προσομοιώσεις που γίνονται από το σενάριο φλοιού run3. Στο σχήμα δείχνεται η  $\langle m \rangle$  και φαίνεται καθαρά η μείωση της μαγνήτισης σε ψηλές θερμοκρασίες  $\beta \ll \beta_c$  ως  $1/L$ .

```

set nsweeps = 100000

foreach L ($Ls )
  set start = "-s 1 -S 3423"
  foreach beta ($betas)
    ./is -L $L -b $beta -n $nsweeps -w $start > outL${L}b${beta}
    set start = "-s 2"
    # Calculate <math>\langle e \rangle = \langle E \rangle / (2N)</math> and <math>c = \beta^2 N (\langle e^2 \rangle - \langle e \rangle^2)</math>:
    grep -v '#' outL${L}b${beta} | \
    awk -v L=$L 'NR>500{print $1/(2*L*L)}' | \
    ./jack | grep -v '#' | \
    awk -v L=$L -v b=$beta \
    '{print "e",L,b,$1,$2,b*b*L*L*$3,b*b*L*L*$4}'
    # Calculate <math>\langle m \rangle = \langle |M| \rangle / N</math> and <math>\chi = \beta^2 N (\langle m^2 \rangle - \langle m \rangle^2)</math>
    grep -v '#' outL${L}b${beta} | \
    awk -v L=$L 'NR>500{m=($2>0)?$2:-$2;print m/(L*L)}' | \
    ./jack | grep -v '#' | \
    awk -v L=$L -v b=$beta \
    '{print "m",L,b,$1,$2,b*L*L*$3,b*L*L*$4}'
  end
end

```



Σχήμα 14.7: Τα αποτελέσματα από τις προσομοιώσεις που γίνονται από το σενάριο φλοιού run3. Στο σχήμα δείχνεται η μαγνητική επιδεκτικότητα  $\chi$ . Μακριά από την κρίσιμη περιοχή είναι σχεδόν ανεξάρτητη από το μέγεθος του πλέγματος, ενώ στην ψευδοκρίσιμη περιοχή έχουμε γρήγορη αύξηση, όπως αναμένεται από την εξίσωση (13.10).

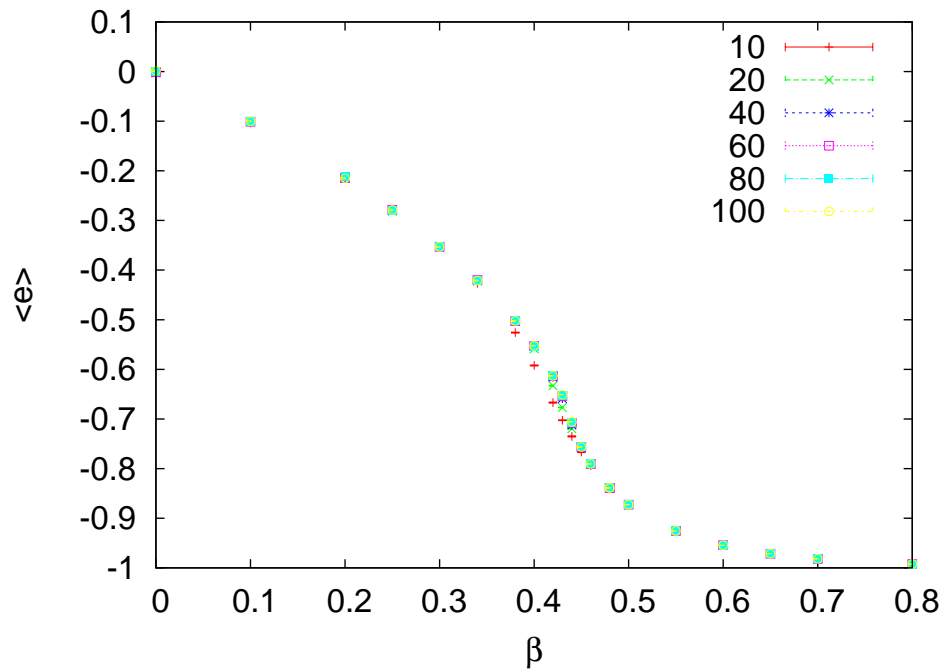
Το τρέχουμε με την εντολή

```
> ./run3 > out &
```

και αφού τελειώσει μπορούμε να δούμε τα αποτελέσματα με το gnuplot<sup>17</sup>:

```
set xlabel "beta"
set ylabel "<m>"
plot "<grep '^m 10 ' out" u 3:4:5 with errorbars title " 10"
replot "<grep '^m 20 ' out" u 3:4:5 with errorbars title " 20"
replot "<grep '^m 40 ' out" u 3:4:5 with errorbars title " 40"
```

<sup>17</sup>Πρόκειται για εντολές που δίνουμε μέσα στο gnuplot, παρόλο που δεν ακολουθούμε τη συνηθισμένη σύμβαση να προτάσσουμε το prompt gnuplot>.



Σχήμα 14.8: Τα αποτελέσματα από τις προσομοιώσεις που γίνονται από το σενάριο φλοιού run3. Στο σχήμα δείχνεται η  $\langle \Phi \rangle$ .

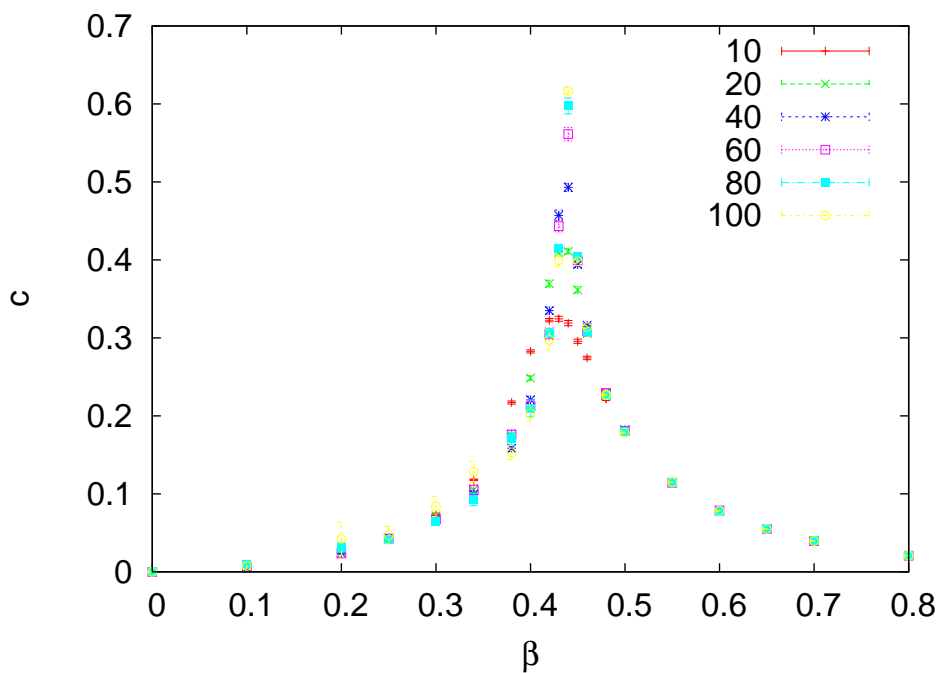
```
replot "<grep '^m 60 ' out" u 3:4:5 with errorbars title " 60"
replot "<grep '^m 80 ' out" u 3:4:5 with errorbars title " 80"
replot "<grep '^m 100 ' out" u 3:4:5 with errorbars title "100"
```

Οι παραπάνω εντολές δίνουν τη γραφική παράσταση της μαγνήτισης.

```
set ylabel "chi"
set log y
plot "<grep '^m 10 ' out" u 3:6:7 with errorbars title " 10"
replot "<grep '^m 20 ' out" u 3:6:7 with errorbars title " 20"
replot "<grep '^m 40 ' out" u 3:6:7 with errorbars title " 40"
replot "<grep '^m 60 ' out" u 3:6:7 with errorbars title " 60"
replot "<grep '^m 80 ' out" u 3:6:7 with errorbars title " 80"
replot "<grep '^m 100 ' out" u 3:6:7 with errorbars title "100"
```

Οι παραπάνω εντολές δίνουν τη γραφική παράσταση της μαγνητικής επιδεκτικότητας.

```
set ylabel "<math>\langle e \rangle</math>"
plot "<grep '^e 10 ' out" u 3:4:5 with errorbars title " 10"
```

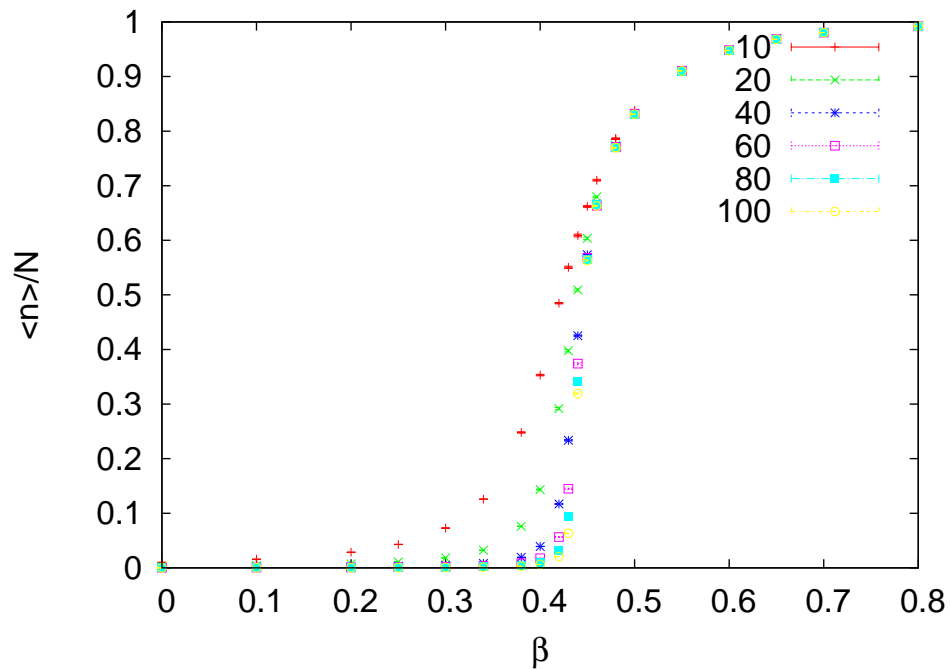


Σχήμα 14.9: Τα αποτελέσματα από τις προσομοιώσεις που γίνονται από το σενάριο φλοιού run3. Στο σχήμα δείχνεται η ειδική θερμότητα  $c$ . Μακριά από την κρίσιμη περιοχή είναι σχεδόν ανεξάρτητη από το μέγεθος του πλέγματος, ενώ στην ψευδοκρίσιμη περιοχή έχουμε αύξηση, όπως αναμένεται από την εξίσωση (13.8).

```
replot "<grep '^e 20 ' out" u 3:4:5 with errorbars title " 20"
replot "<grep '^e 40 ' out" u 3:4:5 with errorbars title " 40"
replot "<grep '^e 60 ' out" u 3:4:5 with errorbars title " 60"
replot "<grep '^e 80 ' out" u 3:4:5 with errorbars title " 80"
replot "<grep '^e 100 ' out" u 3:4:5 with errorbars title "100"
```

Οι παραπάνω εντολές δίνουν τη γραφική παράσταση της ενέργειας.

```
set ylabel "c"
plot "<grep '^e 10 ' out" u 3:6:7 with errorbars title " 10"
replot "<grep '^e 20 ' out" u 3:6:7 with errorbars title " 20"
replot "<grep '^e 40 ' out" u 3:6:7 with errorbars title " 40"
replot "<grep '^e 60 ' out" u 3:6:7 with errorbars title " 60"
replot "<grep '^e 80 ' out" u 3:6:7 with errorbars title " 80"
replot "<grep '^e 100 ' out" u 3:6:7 with errorbars title "100"
```



Σχήμα 14.10: Τα αποτελέσματα από τις προσομοιώσεις που γίνονται από το σενάριο φλοιού run3. Δίνεται η γραφική παράσταση του μέσου μεγέθους των Wolff clusters σαν ποσοστό του συνολικού πλέγματος  $\langle n \rangle / N$ .

Οι παραπάνω εντολές δίνουν τη γραφική παράσταση της ειδικής θερμότητας.

```
set ylabel "<n>/N"
plot "<grep '^n 10 ' out" u 3:4:5 with errorbars title " 10"
replot "<grep '^n 20 ' out" u 3:4:5 with errorbars title " 20"
replot "<grep '^n 40 ' out" u 3:4:5 with errorbars title " 40"
replot "<grep '^n 60 ' out" u 3:4:5 with errorbars title " 60"
replot "<grep '^n 80 ' out" u 3:4:5 with errorbars title " 80"
replot "<grep '^n 100 ' out" u 3:4:5 with errorbars title "100"
```

Οι παραπάνω εντολές δίνουν τη γραφική παράσταση της  $\langle n \rangle / N$ .

## 14.7 Χρόνοι Αυτοσυσχετισμού

Στην περίπτωση του αλγόριθμου του Metropolis ορίσαμε τη “μονάδα του χρόνου” στην κατασκευή του δείγματος στην προσομοίωση Μόντε



Κάρλο το “sweep”. Ένα sweep ορίζεται, όταν έχουμε  $N$  απόπειρες αλλαγής των σπιν του πλέγματος.

Στην περίπτωση του αλγόριθμου του Wolff έχουμε την επιπλέον πολυπλοκότητα που εισάγει το μεταβλητό μέγεθος των Wolff clusters με τη θερμοκρασία. Άρα, αν ορίσουμε ένα βήμα του αλγόριθμου να είναι η κατασκευή ενός cluster, τότε δεν έχουμε έναν καλό ορισμό του sweep. Για πολύ χαμηλές θερμοκρασίες  $\beta \gg \beta_c$ , η κατασκευή ενός cluster είναι σχεδόν ισοδύναμη με μία αλλαγή σπιν ανά πλεγματική θέση. Αν κρατήσουμε αυτό να είναι η μονάδα μέτρησης του χρόνου στην προσομοίωση, τότε ένα τέτοιο sweep θα είναι ίσο με

$$(1 \text{ sweep}) = \frac{N}{\langle n \rangle} (\text{Wolff cluster updates}) \quad (14.30)$$

Για μεγάλες θερμοκρασίες  $\beta \ll \beta_c$  χρειαζόμαστε περίπου  $N$  Wolff clusters για να έχουμε ένα sweep του πλέγματος. Ανάλογα θα ορίσουμε και τους χρόνους αυτοσυσχετισμού. Για μια φυσική ποσότητα  $O$  ο χρόνος αυτοσυσχετισμού σε μονάδες κατασκευής Wolff cluster θα συμβολίζεται με  $\tau_O^W$ , ενώ ο χρόνος αυτοσυσχετισμού  $\tau_O$  που θα χρησιμοποιήσουμε για τον έλεγχο της απόδοσης του αλγόριθμου θα είναι σε μονάδες sweeps και θα ισχύει

$$\tau_O = \tau_O^W \frac{\langle n \rangle}{N}. \quad (14.31)$$

Αρχικά πραγματοποιούμε προσομοιώσεις για  $L = 10, 20, 40, 60, 80$  και 100 στη θερμοκρασία  $\beta = 0.4407$ . Κατασκευάζουμε  $5 \times 10^6$  clusters. Τα αποτελέσματα αποθηκεύονται σε αρχεία με όνομα `outL${L}b0.4407`. Στη συνέχεια, πραγματοποιούμε προσομοιώσεις με τον αλγόριθμο Metropolis και πραγματοποιούμε  $10 \times 10^6$  sweeps. Τα αποτελέσματα αποθηκεύονται σε αρχεία με το όνομα `outL${L}b0.4407met`. Το παρακάτω σενάριο φλοιού κάνει τα πράγματα πιο ... ξεκούραστα.

```
#!/bin/tcsh -f
set Ls      = (10 20 40 60 80 100)
set beta    = 0.4407
set nsweeps = 5000000
set start   = "-s 1 -S 3423"
# Wolff cluster algorithm:
foreach L ($Ls)
./is -L $L -b $beta -n $nsweeps $start -w> outL${L}b${beta}
# Mean cluster size <n>/N
grep '#clu' outL${L}b${beta} | \
awk -v L=$L 'NR>10000{print $2/(L*L)}' | \
./jack -d $nsweeps | grep -v '#'
```

```

awk -v L=$L -v b=$beta '{ print "n",L,b,$1,$2 }'
end
# Metropolis algorithm
set nsweeps = 10000000
foreach L ($Ls)
  ./is -L $L -b $beta -n $nsweeps $start > outL${L}b${beta}met
end

```

Μεταγλωττίζουμε το αρχείο autoc.f90 από τον κατάλογο Tools και αντιγράφουμε το εκτελέσιμο αρχείο autoc στον κατάλογο που δουλεύουμε και με το παρακάτω σενάριο φλοιού μετράμε τις συναρτήσεις αυτοσυσχετισμού της μαγνήτισης  $\rho_m(t)$

```

#!/bin/tcsh -f
set Ls = (10 20 40 60 80 100)
set b = 0.4407
# Wolff
set tmax = 1000
set ndata = 5000000
foreach L ($Ls)
  set f = outL${L}b${b}
  grep -v '#' $f | \
  awk -v L=$L \
  'BEGIN{N=L*L}NR>100000{ print ($2>0)?($2/N):(-$2/N) }' | \
  ./autoc -t $tmax -n $ndata> $f.rhom
end
# Metropolis
set tmax = 8000
set ndata = 10000000
foreach L ($Ls)
  set f = outL${L}b${b}met
  grep -v '#' $f | \
  awk -v L=$L \
  'BEGIN{N=L*L}NR>100000{ print ($2>0)?($2/N):(-$2/N) }' | \
  ./autoc -t $tmax -n $ndata> $f.rhom
end

```

Ο μέγιστος χρόνος στη μέτρηση της  $\rho_m(t)$  είναι φυσικά πολύ μεγαλύτερος για τον αλγόριθμο Metropolis (μεταβλητή \$tmax). Επίσης, πετάμε τις πρώτες 100000 μετρήσεις. Τα αποτελέσματα τα βρίσκουμε σε αρχεία των οποίων οι καταλήξεις είναι .rhom. Στη συνέχεια, προσαρμόζουμε την  $\rho_m(t)$  στη συνάρτηση (13.56) με τρεις χρόνους αυτοσυσχετισμού όπως περιγράφεται στο Παράρτημα 13.7. Τα αποτελέσματά μας δίνονται στον πίνακα 14.1<sup>18</sup>

<sup>18</sup>Παρατηρούμε μια διαφορά στις μετρήσεις μας για τον χρόνο αυτοσυσχετισμού για τον αλγόριθμο Metropolis σε σχέση με τις αριθμητικές τιμές που παρουσιάσαμε

$L$	$\tau_m^W$	$\langle n \rangle / N$	$\tau_m$	$\tau_{m,\text{Metropolis}}$
10	2.18(2)	0.6124(2)	1.33(1)	16.1(1)
20	3.48(5)	0.5159(1)	1.80(3)	70.7(4)
40	5.10(6)	0.4342(2)	2.21(3)	330(6)
60	6.12(6)	0.3927(2)	2.40(2)	795(5)
80	7.33(7)	0.3653(3)	2.68(3)	1740(150)
100	8.36(6)	0.3457(1)	2.89(2)	2660(170)

Πίνακας 14.1: Οι χρόνοι αυτοσυσχετισμού της μαγνήτισης που προκύπτουν από τις προσομοιώσεις και την ανάλυση που περιγράφεται στο κείμενο. Στη 2η στήλη είναι οι χρόνοι αυτοσυσχετισμού  $\tau_m^W$  του αλγόριθμου Wolff, με μονάδα “χρόνου” την κατασκευή ενός Wolff cluster. Η 3η στήλη έχει το μέσο μέγεθος των Wolff clusters σαν ποσοστό του μεγέθους του πλέγματος  $\langle n \rangle / N$ . Η 4η στήλη έχει το  $\tau_m = \tau_m^W \langle n \rangle / N$ , τον χρόνο αυτοσυσχετισμού της μαγνήτισης σε ένα Wolff sweep σύμφωνα με την εξίσωση (14.31). Και τέλος, η 5η στήλη έχει τους χρόνους αυτοσυσχετισμού της μαγνήτισης για τον αλγόριθμο Metropolis για σύγκριση.

Από τη σχέση (14.10) αναμένεται να έχουμε  $\tau_m \sim L^z$  όπου  $z$  ο δυναμικός κρίσιμος εκθέτης. Ο  $z$  υπολογίζεται εύκολα σε μία συνεδρία του gnuplot

```
gnuplot> tau(x) = c*x**z
gnuplot> fit tau(x) "autoc.dat" u 1:2:3 via c,z
gnuplot> plot "autoc.dat" u 1:2:3 w e t 'W steps ', tau(x)
gnuplot> fit tau(x) "autoc.dat" u 1:6:7 via c,z
gnuplot> plot "autoc.dat" u 1:6:7 w e t 'W sweeps ', tau(x)
gnuplot> fit tau(x) "autoc.dat" u 1:8:9 via c,z
gnuplot> plot "autoc.dat" u 1:8:9 w e t "Metropolis", tau(x)
```

όπου υπολογίζουμε το  $z$  για τον αλγόριθμο Wolff σε Wolff steps, Wolff sweeps και για τον αλγόριθμο Metropolis σε Metropolis sweeps. Τα αποτελέσματα είναι

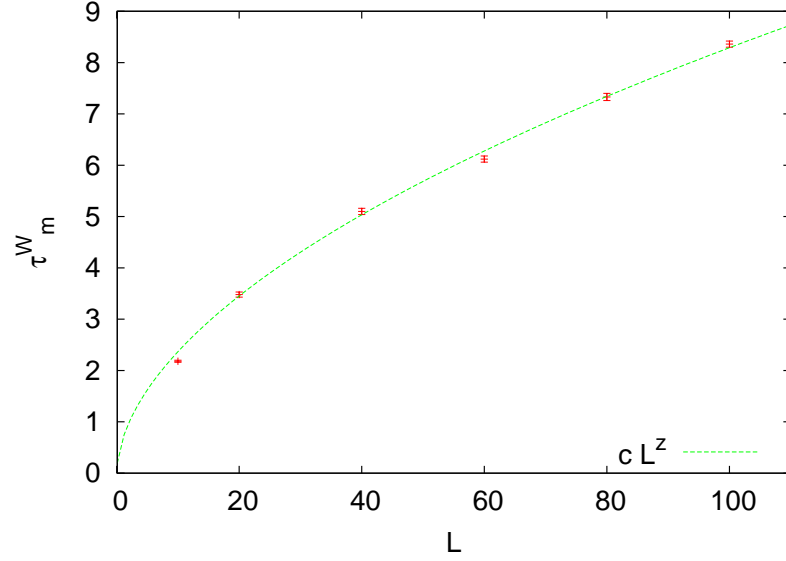
$$\tau_m^W \sim L^z, \quad z = 0.54 \pm 0.02 \quad (14.32)$$

$$\tau_m \sim L^z, \quad z = 0.29 \pm 0.02 \quad (14.33)$$

$$\tau_{m,\text{Metropolis}} \sim L^z, \quad z = 2.21 \pm 0.02 \quad (14.34)$$

Οι αντίστοιχες γραφικές παραστάσεις δίνονται στα σχήματα 14.11-

στο Παράρτημα 13.7. Η διαφορά είναι ο πενταπλασιασμός της στατιστικής μας και δείχνει πως το πραγματικό σφάλμα στον προσδιορισμό των  $\tau$  εμπεριέχει συστηματικά σφάλματα που δεν έχουμε λάβει υπόψη μας.



Σχήμα 14.11: Οι χρόνοι αυτοσυσχετισμού  $\tau_m^W$  της μαγνήτισης για τον αλγόριθμο του Wolff για  $\beta = 0.4407$ . Η μονάδα του χρόνου είναι ένα Wolff cluster. Ο δυναμικός κρίσιμος εκθέτης βρίσκεται με την προσρμογή της συνάρτησης  $cL^{z^W}$  από όπου προκύπτει  $z^W = 0.54(2)$ .

14.13. Οι τιμές που αναμένονται από τη βιβλιογραφία είναι  $0.50(1)$ ,  $0.25(1)$  και  $2.167(1)$  [4, 59, 66]. Καλύτερα αποτελέσματα μπορούμε να πάρουμε αυξάνοντας τη στατιστική και το μέγεθος του πλέγματος, κάτι που το αφήνουμε ως άσκηση στον αναγνώστη.

Αξίζει τον κόπο να σημειώσουμε τη σχέση μεταξύ των δυναμικών εκθετών των εξισώσεων (14.32) και (14.33). Από την εξίσωση (14.29)  $\chi = \beta \langle n \rangle$  και από την εξίσωση (13.10)  $\chi \sim |t|^{-\gamma}$ , (13.6)  $\xi \sim |t|^{-\nu}$  και στην ψευδοκρίσιμη περιοχή  $\xi \sim L$  παίρνουμε

$$\tau_m = \tau_m^W \frac{\langle n \rangle}{L^2} \sim L^{z^W} \frac{L^{\gamma/\nu}}{L^2} = L^{z^W + \gamma/\nu - 2} \quad (14.35)$$

όπου υποθέσαμε  $\tau_m^W \sim L^{z^W}$ ,  $z^W \equiv 0.54(2)$  και  $\tau_m \sim L^z$ . Οπότε παίρνουμε

$$z = z^W + \frac{\gamma}{\nu} - 2. \quad (14.36)$$

Χρησιμοποιώντας τις τιμές (13.12)  $\gamma = 7/4$ ,  $\nu = 1$  παίρνουμε

$$z = z^W - \frac{1}{4}, \quad (14.37)$$

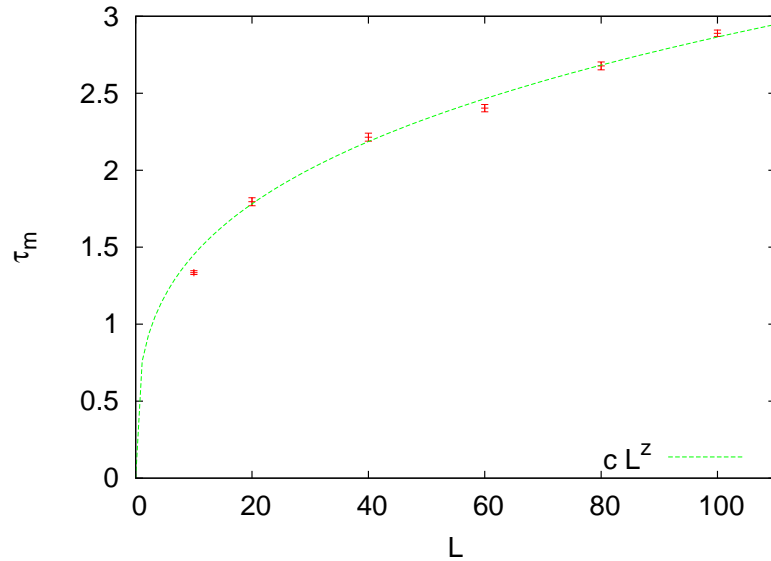
που ισχύει για τις τιμές που υπολογίσαμε, μέσα στα όρια του σφάλματος, καθώς και για τις τιμές της βιβλιογραφίας.

$L$	$\gamma(t < 0)$	$\gamma(t > 0)$
40	1.7598(44)	1.730(17)
60	1.7455(24)	1.691(14)
80	1.7409(21)	1.737(12)
100	1.7420(24)	1.7226(75)
120	1.7390(15)	1.7725(69)
140	1.7390(23)	1.7354(72)
160	1.7387(10)	1.746(17)
200	1.7380(11)	1.759(15)
500	1.7335(8)	1.7485(83)

Πίνακας 14.2: Υπολογισμός του κρίσιμου εκθέτη  $\gamma$  από την προσαρμογή των δεδομένων των Σχημάτων 14.14 και 14.15. Στην πρώτη στήλη προσαρμόζονται τα δεδομένα για  $\beta > \beta_c(t < 0)$  και στη δεύτερη τα δεδομένα για  $\beta < \beta_c(t > 0)$ . Στις παρενθέσεις δίνονται τα στατιστικά σφάλματα των προσαρμογών και όχι τα συστηματικά που προέρχονται από την επιλογή του διαστήματος της προσαρμογής. Η αναμενόμενη τιμή είναι  $\gamma = 7/4$ .

$L$	$\beta(t < 0)$	$\beta_+(t > 0)$
40	0.1101(7)	0.1122(29)
60	0.1129(5)	0.1102(19)
80	0.1147(5)	0.1118(21)
100	0.1175(3)	0.1170(11)
120	0.1167(4)	0.1172(16)
140	0.1190(2)	0.1187(19)
160	0.1191(4)	0.1134(20)
200	0.1205(10)	0.1138(24)
500	0.1221(2)	0.1294(50)

Πίνακας 14.3: Υπολογισμός του κρίσιμου εκθέτη  $\beta$  από την προσαρμογή των δεδομένων των Σχημάτων 14.16. Στην πρώτη στήλη προσαρμόζονται τα δεδομένα για  $\beta > \beta_c(t < 0)$  και στη δεύτερη τα δεδομένα για  $\beta < \beta_c(t > 0)$ . Στις παρενθέσεις δίνονται τα στατιστικά σφάλματα των προσαρμογών και όχι τα συστηματικά που προέρχονται από την επιλογή του διαστήματος της προσαρμογής. Η αναμενόμενη τιμή είναι  $\beta = \beta_+ = 1/8$ .



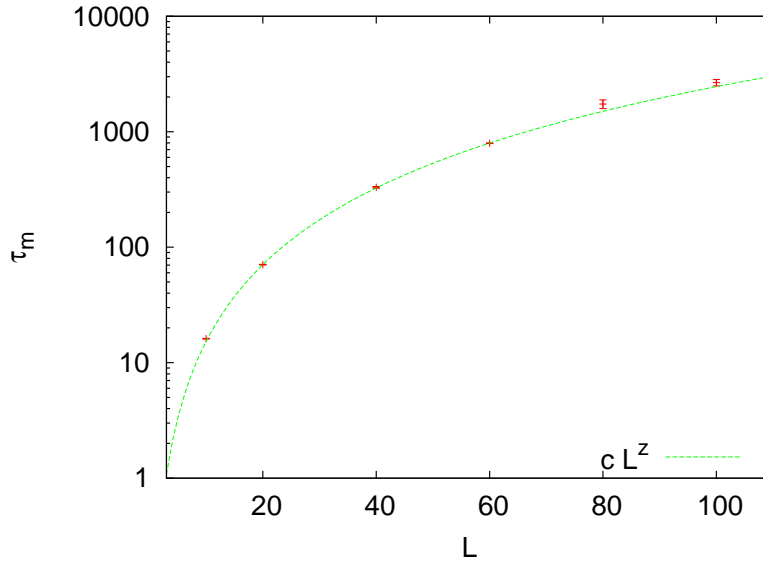
Σχήμα 14.12: Οι χρόνοι αυτοσυσχετισμού  $\tau_m$  της μαγνήτισης για τον αλγόριθμο του Wolff για  $\beta = 0.4407$ . Η μονάδα του χρόνου είναι ένα Wolff sweep. Ο δυναμικός κρίσιμος εκθέτης βρίσκεται με την προσαρμογή της συνάρτησης  $cL^z$  από όπου προκύπτει  $z = 0.29(2)$ .

## 14.8 Βάθμιση Θερμοκρασίας

Στην παράγραφο αυτή θα ελέγξουμε τον βαθμό με τον οποίο οι σχέσεις (14.3)–(14.5) μπορούν να χρησιμοποιηθούν για τον υπολογισμό των κρίσιμων εκθετών  $\gamma$ ,  $\alpha$  και  $\beta$ . Το συμπέρασμα που θα βγάλουμε είναι πως οι σχέσεις αυτές δεν είναι η καλύτερη επιλογή για τον εν λόγω υπολογισμό και ότι θα πρέπει να αποφεύγονται, αν είναι δυνατόν<sup>19</sup>. Για να δούμε καθαρή βάθμιση, θα πρέπει να προσομοιώσουμε για πολύ μικρά  $t \ll 1$  και για αρκετά μεγάλο  $L$  λόγω των μεγάλων φαινομένων επίδρασης πεπερασμένου μεγέθους (finite size effects). Επίσης, υπάρχουν παραδείγματα που οι τιμές που παίρνει κανείς εξαρτώνται ισχυρά από το διάστημα προσαρμογής των δεδομένων με αποτέλεσμα να έχουμε μεγάλα συστηματικά σφάλματα από τη διαδικασία προσαρμογής (fitting) ή ακόμα και λάθος αποτελέσματα<sup>20</sup>.

<sup>19</sup>Παρατηρήστε πως στο συγκεκριμένο μοντέλο η κρίσιμη θερμοκρασία μας είναι επακριβώς γνωστή. Όταν δεν την γνωρίζουμε, τότε υπεισέρχονται επιπλέον συστηματικά σφάλματα που δυσχεραίνουν τον υπολογισμό. Για τον αριθμητικό υπολογισμό της κρίσιμης θερμοκρασίας θα μιλήσουμε σε επόμενη παράγραφο.

<sup>20</sup>Στο σύγγραμμα [4] αναφέρεται πως στο random field Ising model παρουσιάζεται ψευδοβάθμιση για ένα διάστημα του  $t$ , ενώ για πολύ μικρότερο  $t$  έχουμε crossover σε



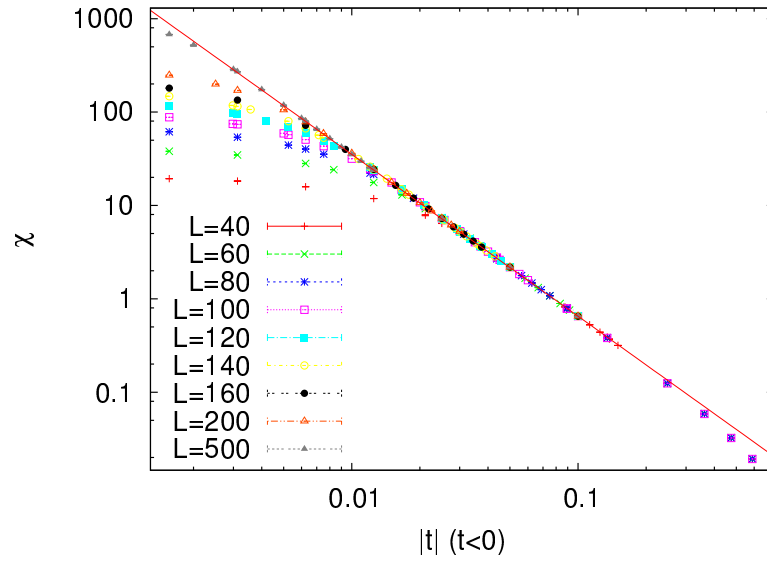
Σχήμα 14.13: Οι χρόνοι αυτοσυσχετισμού  $\tau_{m,\text{Metropolis}}$  της μαγνήτισης για τον αλγόριθμο Metropolis για  $\beta = 0.4407$ . Η μονάδα του χρόνου είναι ένα Metropolis sweep. Ο δυναμικός κρίσιμος εκθέτης βρίσκεται με την προσαρμογή της συνάρτησης  $cL^z$  από όπου προκύπτει  $z = 2.21(2)$ .

Για τον λόγο αυτό προσομοιώσαμε το πρότυπο Ising για  $L = 40, 60, 80, 100, 120, 140, 160, 200$  και  $500$ . Διαλέξαμε θερμοκρασίες που αντιστοιχούν σε αρκετά μικρά  $t$  γύρω από την κρίσιμη θερμοκρασία. Στο συνοδευτικό λογισμικό θα βρείτε τα shell scripts που χρησιμοποιήθηκαν για τον λόγο αυτό.

Αρχικά υπολογίζουμε τον εκθέτη  $\gamma$  από τη σχέση (14.3). Προσαρμόζουμε στην κατάλληλη περιοχή του  $|t|$  την τιμή της  $\chi(t)$  για κάθε τιμή του  $L$  στη συνάρτηση  $a|t|^{-\gamma}$  που έχει δύο ανεξάρτητες παραμέτρους  $\gamma, a$ . Η επιλογή του διαστήματος γίνεται με τον εντοπισμό γραμμικής συμπεριφοράς της  $\chi(t)$  σε διάγραμμα με λογαριθμική κλίμακα και στους δύο άξονες<sup>21</sup>. Παρατηρούμε ότι για μεγάλο  $|t|$  έχουμε απόκλιση από τη γραμμική συμπεριφορά, ενώ για πολύ μικρό  $|t|$  παρουσιάζονται σφάλματα από τα φαινόμενα επίδρασης πεπερασμένου μεγέθους, όταν  $\xi \approx L$ . Καθώς το  $L$  μεγαλώνει, τα φαινόμενα επίδρασης πεπερασμένου μεγέθους μειώνονται και τα δεδομένα πλησιάζουν κοντύτερα προς την ασυμπτωτική συμπεριφορά  $|t|^{-\gamma}$  για ολοένα και μικρότερο  $|t|$ . Τα αποτελέσματα

διαφορετική βάθμιση που δίνει το σωστό εκθέτη. Δείτε επίσης τα [67], [68].

<sup>21</sup>Η προσαρμογή μπορεί να γίνει και με γραμμική προσαρμογή σε ευθεία των σημείων  $(\log |t_i|, \log \chi(t_i))$  με απλή μέθοδο ελαχίστων τετραγώνων.

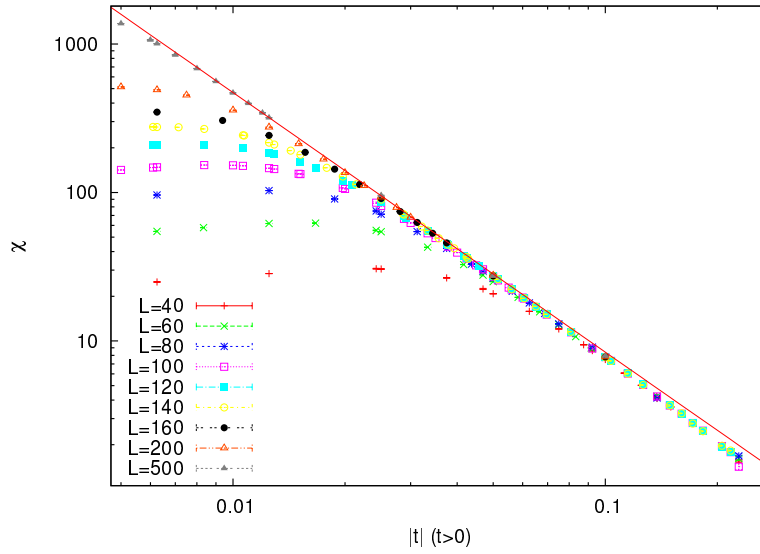


Σχήμα 14.14: Οι συναρτήσεις της μαγνητικής επιδεκτικότητας  $\chi(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάρθμιση σύμφωνα με την σχέση (14.3). Η ευθεία γραμμή είναι προσαρμογή στη σχέση αυτή για το μεγαλύτερο πλέγμα. Παρατηρήστε την ελάττωση των finite size effects καθώς το  $L$  μεγαλώνει και το διάστημα προσαρμογής επεκτείνεται σε ολοένα μικρότερα  $|t|$ . Τα δεδομένα είναι για θερμοκρασίες  $\beta > \beta_c(t < 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση τάξης.

είναι καθαρότερα για  $\beta > \beta_c(t < 0)$ , γιατί για  $t > 0$  οι διακυμάνσεις είναι ισχυρότερες κοντά στην ψευδοκρίσιμη θερμοκρασία  $\beta_c(L) < \beta_c$  και έχουμε μεγαλύτερα φαινόμενα επίδρασης πεπερασμένου μεγέθους.

Στον πίνακα 14.2 σημειώνουμε τα αποτελέσματα για τον εκθέτη  $\gamma$  για όλες τις τιμές του  $L$  που μετρήσαμε. Τα σφάλματα που σημειώνονται στον πίνακα είναι τα απλά στατιστικά σφάλματα της προσαρμογής που έδωσε το αναφερόμενο αποτέλεσμα και είναι πολύ μικρότερο από το συστηματικό σφάλμα της προσαρμογής. Το τελευταίο προέρχεται κυρίως από την εκτίμηση του διαστήματος θερμοκρασιών  $t$  που θα συμπεριλάβουμε στην προσαρμογή. Μία συστηματική αντιμετώπιση της εκτίμησης των σφαλμάτων αυτών είναι να αλλάζουμε το διάστημα αυτό και να συμπεριλάβουμε στις δυνατές τιμές αυτές που προκύπτουν από όλες τις αποδεκτές προσαρμογές στη συνάρτηση που επιλέξαμε. Καμιά φορά αυτό δίνει υπερεκτιμημένο σφάλμα και έγκειται στην “τέχνη” και εμπειρία μας να αποφασίσουμε τις τιμές που θα κρατήσουμε. Για παράδειγμα, από τα σχήματα 14.14 και 14.15 παρατηρούμε ότι το διάστημα της προσαρμογής γίνεται πιο ευδιάκριτο μελετώντας την  $\chi(t)$  για ολοένα και μεγαλύτερα  $L$ . Καθώς το  $L$  μεγαλώνει, τα σημεία πλησιάζουν





Σχήμα 14.15: Οι συναρτήσεις της μαγνητικής επιδεκτικότητας  $\chi(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάθμιση σύμφωνα με την σχέση (14.3). Η ευθεία γραμμή είναι προσαρμογή στη σχέση αυτή για το μεγαλύτερο πλέγμα. Παρατηρήστε την ελάττωση των finite size effects καθώς το  $L$  μεγαλώνει και το διάστημα προσαρμογής επεκτείνεται σε ολοένα μικρότερα  $|t|$ . Τα δεδομένα είναι για θερμοκρασίες  $\beta < \beta_c(t > 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση αταξίας. Τα finite size effects είναι μεγαλύτερα από την περίπτωση  $t < 0$  λόγω της επίδρασης των μεγάλων διακυμάνσεων στο ψευδοκρίσιμο σημείο  $\beta_c(L) < \beta_c$ .

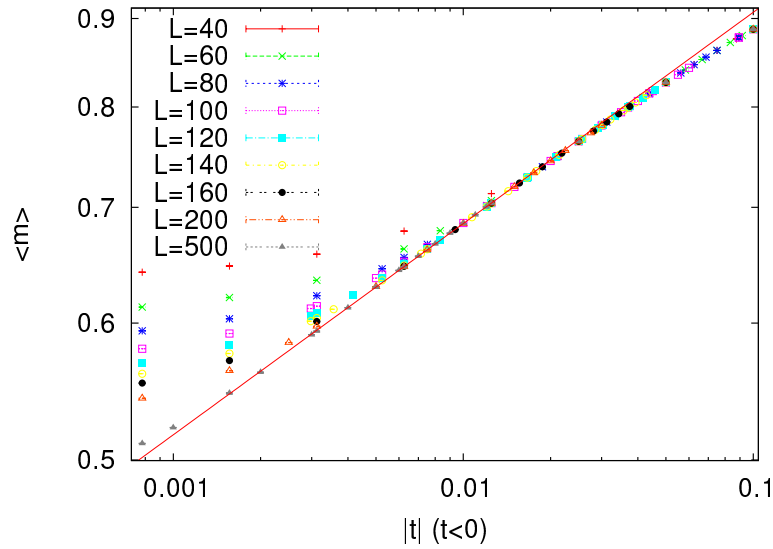
ζουν προς την ασυμπτωτική καμπύλη ολοένα και περισσότερο. Επίσης, οι λογικές τιμές για τις παραμέτρους που υπεισέρχονται στην προσαρμογή μπορούν να αποτελέσουν και αυτές ένα κριτήριο αποδοχής ή όχι του αποτελέσματος μιας προσαρμογής (λ.χ. να έχει η σταθερά  $a$  τιμές  $\sim 1$ ). Λαμβάνοντας υπόψη τα παραπάνω, μπορούμε να αναφέρουμε τα αποτελέσματα

$$\gamma = 1.74 \pm 0.02 \quad (t < 0), \quad (14.38)$$

$$\gamma = 1.73 \pm 0.04 \quad (t > 0). \quad (14.39)$$

Στη συνέχεια, υπολογίζουμε τον κρίσιμο εκθέτη  $\beta$  από την εξίσωση (14.5). Η σχέση αυτή ισχύει, καθώς προσεγγίζουμε το κρίσιμο σημείο από την ψυχρή περιοχή  $\beta > \beta_c$  ή  $t < 0$ . Στο θερμοδυναμικό όριο η μαγνήτιση είναι παντού μηδέν για κάθε  $\beta < \beta_c$ . Παρόλα αυτά, στο πεπερασμένο πλέγμα  $\langle m \rangle > 0$  και είναι λογικό να περιμένουμε μία σχέση βάθμισης της μορφής

$$\langle m \rangle \sim |t|^{\beta_+ - 1}, \quad t > 0. \quad (14.40)$$



Σχήμα 14.16: Οι συναρτήσεις της μαγνήτισης  $\langle m \rangle(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάρθμιση σύμφωνα με την σχέση (14.5). Η ευθεία γραμμή είναι προσαρμογή στη σχέση αυτή για το μεγαλύτερο πλέγμα. Παρατηρήστε την ελάττωση των finite size effects καθώς το  $L$  μεγαλώνει και το διάστημα προσαρμογής επεκτείνεται σε ολοένα μικρότερα  $|t|$ . Τα δεδομένα είναι για θερμοκρασίες  $\beta > \beta_c(t < 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση τάξης.

όπου ο ορισμός του  $\beta_+$  γίνεται έτσι, γιατί όπως θα δούμε

$$\beta_+ = \beta = 1/8. \quad (14.41)$$

Ακολουθούμε τη διαδικασία προσαρμογής των δεδομένων που περιγράψαμε παραπάνω. Τα αποτελέσματα για τους εκθέτες  $\beta$  και  $\beta_+$  καταγράφονται στον πίνακα 14.3. Λαμβάνοντας υπόψη τα συστηματικά σφάλματα που περιγράψαμε παραπάνω βρίσκουμε ότι

$$\beta = 0.121 \pm 0.003 \quad t < 0, \quad (14.42)$$

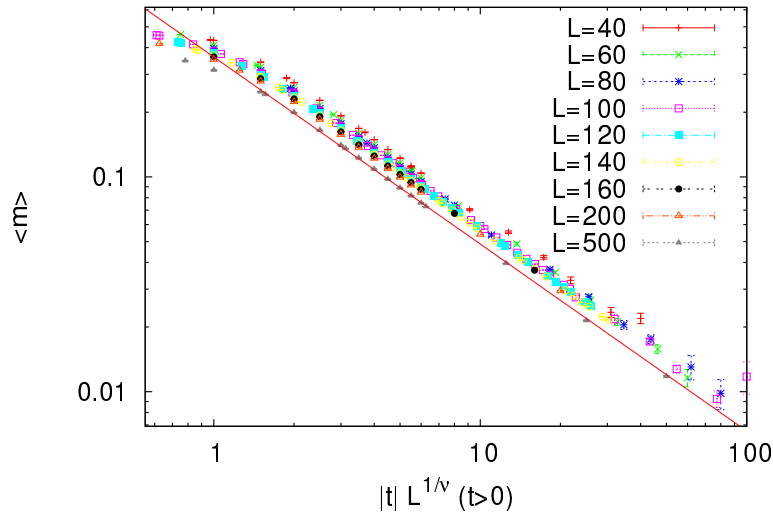
$$\beta_+ = 0.120 \pm 0.007 \quad t < 0, \quad (14.43)$$

που πρέπει να συγκριθεί με την αναμενόμενη τιμή  $\beta = \beta_+ = 1/8$ .

Η περίπτωση του εκθέτη  $\alpha$  χρειάζεται ιδιαίτερη προσοχή. Η τιμή που αναμένεται είναι  $\alpha = 0$ . Αυτό δε σημαίνει ότι  $c \sim$  σταθ αλλά ότι<sup>22</sup>

$$c \sim |\log |t||. \quad (14.44)$$

<sup>22</sup> Αυτό δεν αποκλείει η κύρια συμπεριφορά να είναι μια δύναμη του λογαρίθμου ή λογάριθμος του λογαρίθμου κλπ. Αυτό θα πρέπει να μελετηθεί με προσοχή.

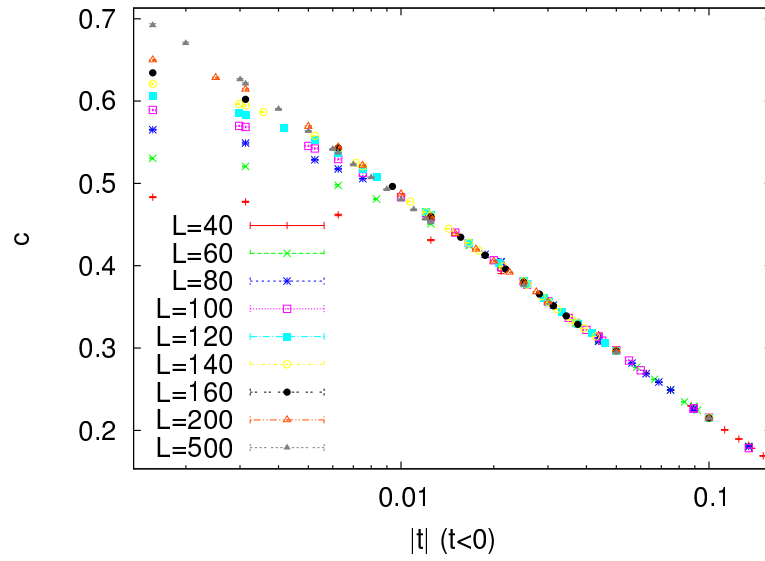


Σχήμα 14.17: Οι συναρτήσεις της μαγνήτισης  $\langle m \rangle(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάθμιση σύμφωνα με την σχέση (14.40). Η ευθεία γραμμή είναι προσαρμογή στη σχέση αυτή για το μεγαλύτερο πλέγμα. Παρατηρήστε την ελάττωση των finite size effects καθώς το  $L$  μεγαλώνει και το διάστημα προσαρμογής επεκτείνεται σε ολοένα μικρότερα  $|t|$ . Τα δεδομένα είναι για θερμοκρασίες  $\beta < \beta_c(t > 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση αταξίας.

Στην περίπτωση αυτή βρίσκουμε ότι τα δεδομένα προσαρμόζονται καλύτερα στην παραπάνω σχέση παρά σε μια σχέση δύναμης, όπως στην περίπτωση της μαγνήτισης και της μαγνητικής επιδεκτικότητας. Αυτό μπορεί να φανεί εποπτικά κάνοντας τη γραφική παράσταση σε λογαριθμική κλίμακα μόνο κατά τον άξονα  $|t|$  και να συγκρίνουμε με την ανάλογη γραφική παράσταση όπου και οι δύο άξονες είναι σε λογαριθμική κλίμακα. Βρίσκουμε ότι τα δεδομένα μας ακολουθούν μια ευθεία στην πρώτη περίπτωση ενώ στη δεύτερη όχι. Μια προσεκτική μελέτη θα συγκρίνει την ποιότητα των προσαρμογών στις δύο συναρτήσεις και θα επιλέξει το καλύτερο μοντέλο. Το αφήνουμε ως άσκηση στον αναγνώστη.

## 14.9 Βάθμιση Πεπερασμένου Μεγέθους

Στην παράγραφο αυτή θα υπολογίσουμε τους κρίσιμους εκθέτες με τη βοήθεια των σχέσεων (14.7)-(14.9), δηλ. με τη βάθμιση που παρουσιάζουν ασυμπτωτικά οι  $\chi(\beta = \beta_c, L)$ ,  $c(\beta = \beta_c, L)$  και  $\langle m \rangle(\beta = \beta_c, L)$ , καθώς το μέγεθος του συστήματος αυξάνει. Η διεθνής ορολογία για τη μέθοδο αυτή του υπολογισμού είναι “finite size scaling”.

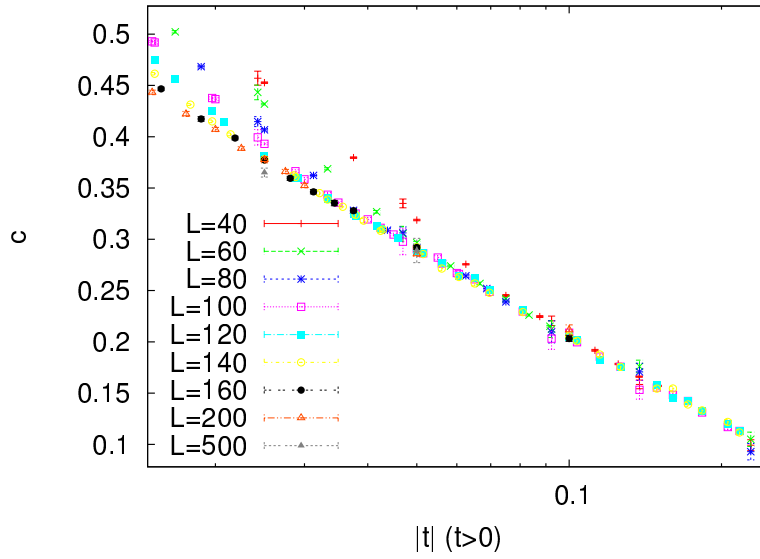


Σχήμα 14.18: Οι συναρτήσεις της ειδικής θερμότητας  $c(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάρθρωση σύμφωνα με την σχέση (14.44). Παρατηρήστε ότι μόνο ο άξονας  $|t|$  είναι σε λογαριθμική κλίμακα. Τα δεδομένα είναι για θερμοκρασίες  $\beta > \beta_c(t < 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση τάξης.

Η σχέση (14.7) δίνει τον εκθέτη  $\gamma/\nu$ . Για τον υπολογισμό του, χρησιμοποιούμε τις τιμές της μαγνητικής επιδεκτικότητας στη γνωστή θερμοκρασία  $\beta_c$  για διαφορετικές τιμές του  $L$ . Με προσαρμογή των τιμών  $\chi(\beta_c, L)$  στη συνάρτηση  $aL^g$  υπολογίζουμε τις σταθερές  $a$  και  $g$  και συγκρίνουμε την τελευταία με την αναμενόμενη τιμή  $\gamma/\nu = 7/4 = 1.75$ . Στη διαδικασία των προσαρμογών πρέπει να αποφασίσουμε ποιες τιμές του  $L$  θα χρησιμοποιήσουμε. Φυσικά, το πρώτο κριτήριο είναι οι προσαρμογές να δίνουν αποδεκτό  $\chi^2/\text{dof} \lesssim 1$  και το σφάλμα στις σταθερές  $g$  και  $a$  να είναι μικρό, αλλά παρατηρούμε ότι και ο περιορισμός αυτός

$L$	$\gamma/\nu$	$\beta/\nu$
40–100	1.754(1)	0.1253(1)
140–1000	1.740(2)	0.1239(3)
40–1000	1.749(1)	0.1246(1)

Πίνακας 14.4: Οι κρίσιμοι εκθέτες  $\gamma/\nu$  και  $\beta/\nu$  που δίνονται από τη μελέτη βάρθρωσης μεγέθους των σχέσεων (14.7) και (14.9). Στην πρώτη στήλη δίνονται τα διαστήματα στις τιμές του  $L$  που θεωρήθηκαν στην προσαρμογή των τιμών των  $\chi(\beta_c, L)$  και  $\langle m \rangle(\beta_c, L)$  σε συναρτήσεις της μορφής  $aL^g$  και δίπλα οι αντίστοιχες τιμές για (την απόλυτη τιμή) των εκθετών  $g$  που προκύπτουν.



Σχήμα 14.19: Οι συναρτήσεις της ειδικής θερμότητας  $c(t, L)$  στην περιοχή θερμοκρασιών που παρουσιάζεται βάθμιση σύμφωνα με την σχέση (14.44). Παρατηρήστε ότι μόνο ο άξονας  $|t|$  είναι σε λογαριθμική κλίμακα. Τα δεδομένα είναι για θερμοκρασίες  $\beta < \beta_c(t > 0)$  όπου το κρίσιμο σημείο προσεγγίζεται από τη φάση αταξίας.

δεν είναι αρκετός: Στον πίνακα 14.4 βλέπουμε μικρές μεταβολές στις τιμές του εκθέτη  $\gamma/\nu$  που παίρνουμε για διαφορετικά διαστήματα προσρμογής. Οι μεταβολές αυτές μας δίνουν ένα μέτρο του συστηματικού σφάλματος στον προσδιορισμό τους. Στο πρόβλημα 9 σας ζητείται να δοκιμάσετε τον υπολογισμό μόνοι σας. Στον πίνακα 14.4 δίνονται τα αποτελέσματα και στο σχήμα 14.20 η αντίστοιχη γραφική παράσταση. Το τελικό αποτέλεσμα, λαμβάνοντας υπόψη το συστηματικό σφάλμα, είναι:

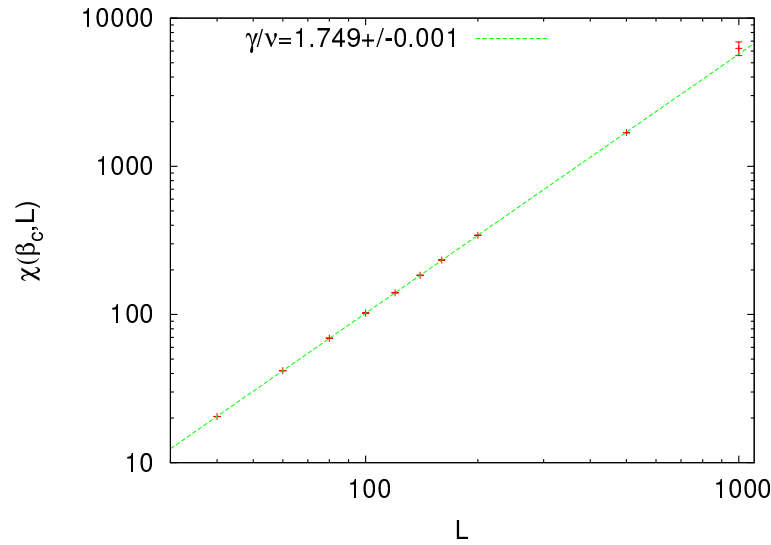
$$\frac{\gamma}{\nu} = 1.748 \pm 0.005. \quad (14.45)$$

Η σχέση (14.9) δίνει τον εκθέτη  $\beta/\nu$ . Για τον υπολογισμό του, ακολουθούμε ανάλογη διαδικασία για τις τιμές της μαγνήτισης  $\langle m \rangle(\beta_c, L)$  στην κρίσιμη θερμοκρασία. Το αποτέλεσμα είναι

$$\frac{\beta}{\nu} = 0.1245 \pm 0.0006. \quad (14.46)$$

Η σχέση (14.9) δίνει τον εκθέτη  $\alpha/\nu$ . Αλλά η αναμενόμενη τιμή  $\alpha = 0$  οδηγεί, σε αναλογία με τη σχέση (14.44), στη σχέση

$$c(\beta_c, L) \sim \log L. \quad (14.47)$$

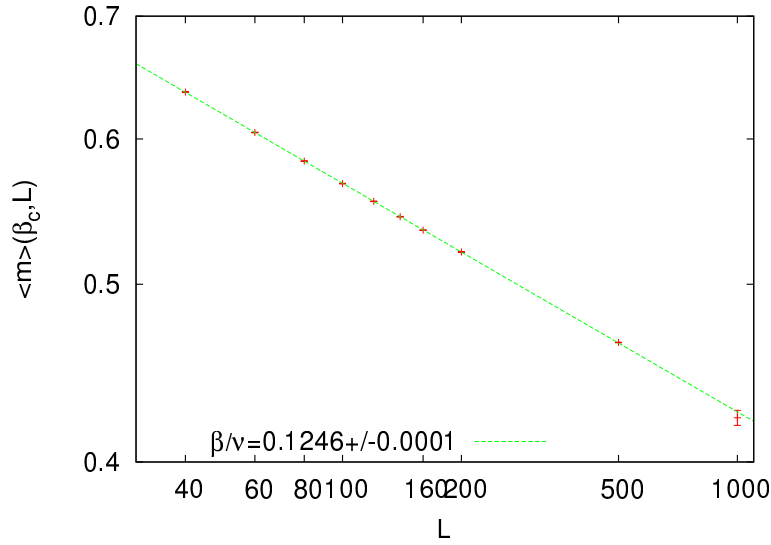


Σχήμα 14.20: Οι τιμές  $\chi(\beta_c, L)$  στην κρίσιμη θερμοκρασία για διαφορετικές τιμές του  $L$ . Οι κλίμακες στους άξονες είναι λογαριθμικές και η γραμμική σχέση των δεδομένων που προκύπτει, αποδεικνύει τη σχέση δύναμης  $\chi(\beta_c, L) = cL^g$ . Με προσαρμογή των δεδομένων στην συνάρτηση αυτή, ο υπολογισμός του εκθέτη  $g$  δίνει τον κρίσιμο εκθέτη  $\gamma/\nu$ , όπως προκύπτει από τη σχέση (14.7).

Η σχέση αυτή δείχνεται στο σχήμα 14.22. Ο κάθετος άξονας δεν είναι σε λογαριθμική κλίμακα, ενώ ο οριζόντιος είναι. Η γραμμική σχέση των δεδομένων στο σχήμα δείχνει ότι είναι συμβατά με την εξίσωση (14.47). Στο πρόβλημα 9 θα πρέπει να δείξετε αν η προσαρμογή των δεδομένων στο λογάριθμο είναι καλύτερη από την προσαρμογή σε μια συνάρτηση δύναμης της μορφής  $cL^a + b$  και να μελετήσετε τις δυσκολίες που προκύπτουν στη μελέτη αυτή. Βελτιώνοντας τα δεδομένα του πίνακα 14.8 με μεγαλύτερη στατιστική (ελάττωση στατιστικού σφάλματος) και μετρήσεις σε μεγαλύτερα  $L$ , τα αποτελέσματα μπορούν να γίνουν ακόμα καθαρότερα.

Παρατηρούμε ότι με τη μέθοδο βάθμισης πεπερασμένου μεγέθους (finite size scaling) οι τιμές που προκύπτουν για τους κρίσιμους εκθέτες προκύπτουν ευκολότερα από τη μελέτη της βάθμισης ως προς τη θερμοκρασία που μελετήσαμε στην παράγραφο 14.8. Οι τιμές που μετράμε ακολουθούν την ασυμπτωτική σχέση βάθμισης που δίνουν οι σχέσεις (14.7)–(14.9) με πολύ μικρότερη επίδραση των φαινομένων επίδρασης πεπερασμένου μεγέθους<sup>23</sup>. Μπορούμε με τη μελέτη πολύ μικρότερου συ-

<sup>23</sup>Θυμίζουμε στον αναγνώστη πως στους υπολογισμούς που δώσανε τις παραπάνω τιμές είχαμε μεγαλύτερη αστάθεια στις προσαρμογές ως προς την επιλογή διαστήμα-



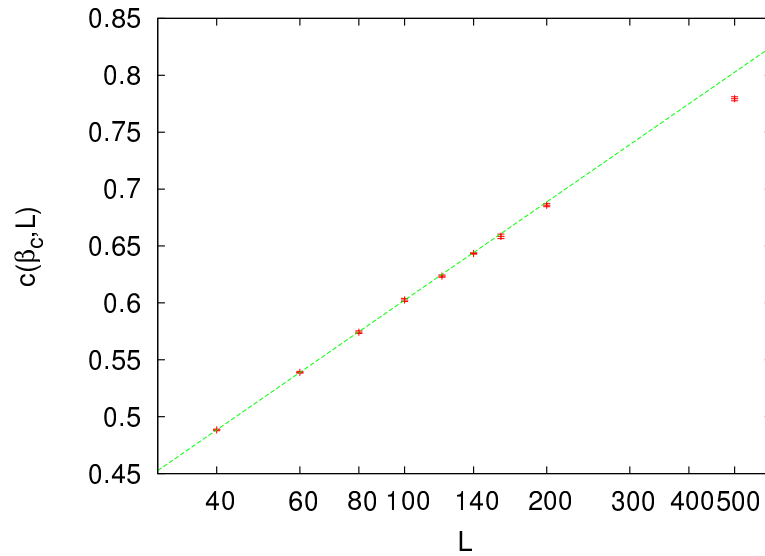
Σχήμα 14.21: Οι τιμές  $\langle m \rangle(\beta_c, L)$  στην κρίσιμη θερμοκρασία για διαφορετικές τιμές του  $L$ . Οι κλίμακες στους άξονες είναι λογαριθμικές και η γραμμική σχέση των δεδομένων που προκύπτει, αποδεικνύει τη σχέση δύναμης  $\langle m \rangle(\beta_c, L) = cL^g$ . Με προσαρμογή των δεδομένων στην συνάρτηση αυτή, ο υπολογισμός του εκθέτη  $g$  δίνει τον κρίσιμο εκθέτη  $\beta/\nu$ , όπως προκύπτει από τη σχέση (14.9).

στήματος να πάρουμε τιμές που έχουν μεγαλύτερη ακρίβεια και προκύπτουν καθαρότερα από αυτές που δώσαμε στις εξισώσεις (14.38), (14.39), (14.42), (14.43).

## 14.10 Προσδιορισμός της $\beta_c$

Στην παράγραφο 14.3 συζητήσαμε τη βάθμιση των φυσικών ποσοτήτων ως συνάρτηση της ανηγμένης θερμοκρασίας  $t$  και του μεγέθους του συστήματος  $L$ . Τη συζήτηση βοήθησε σημαντικά η γνώση της ακριβούς κρίσιμης θερμοκρασίας  $\beta_c = \log(1 + \sqrt{2})/2$  η οποία είναι γνωστή από τους αναλυτικούς υπολογισμούς και ειδικότερα από τη λύση του Onsager [55]. Δυστυχώς, στις περισσότερες ενδιαφέρουσες περιπτώσεις η κρίσιμη παράμετρος δεν είναι γνωστή. Στην περίπτωση αυτή η ανάλυση γίνεται δυσκολότερη, αφενός μεν γιατί η κρίσιμη θερμοκρασία πρέπει να υπολογιστεί και αφετέρου, διότι η πεπερασμένη ακρίβεια υπολογισμού της (άρα και στον υπολογισμό της  $t$ ) εισάγει σημαντικά συστη-

τος των τιμών των δεδομένων. Στην επόμενη παράγραφο θα δούμε ότι όταν η ακριβής τιμή της κρίσιμης θερμοκρασίας μας είναι άγνωστη, η υπεροχή του finite size scaling είναι ακόμα μεγαλύτερη.



Σχήμα 14.22: Οι τιμές  $c(\beta_c, L)$  στην κρίσιμη θερμοκρασία για διαφορετικές τιμές του  $L$ . Η κλίμακα στον οριζόντιο άξονα είναι λογαριθμική, ενώ στον κάθετο δεν είναι. Αυτό αποδεικνύει τη σχέση  $c(\beta_c, L) = c \log L$  το οποίο αποδεικνύεται και με την προσαρμογή των δεδομένων στην αντίστοιχη συνάρτηση. Αυτό είναι συμβατό με το γνωστό αποτέλεσμα  $\alpha = 0$  στη σχέση (14.8).

ματικά σφάλματα στον υπολογισμό των κρίσιμων εκθετών. Επίσης, η μη εκ των προτέρων γνώση της  $\beta_c$  κάνει την επιλογή των τιμών που θα επιλέξουμε στις σχέσεις βάθμισης (14.7)–(14.9) να πρέπει να προσδιοριστεί ακριβώς, ώστε να έχουμε γρήγορη σύγκλιση στην ασυμπτωτική συμπεριφορά και ελαχιστοποίηση των συστηματικών σφαλμάτων. Εδώ η κατανόηση της αιτίας της βάθμισης των φυσικών ποσοτήτων μπορεί να βοηθήσει. Ο λόγος που οδηγεί την ψευδοκρίσιμη συμπεριφορά ενός πεπερασμένου συστήματος που παρουσιάζει συνεχή μετάβαση φάσης στο θερμοδυναμικό όριο είναι η αύξηση του μήκους συσχετισμού  $\xi$  σε μέγεθος που συγκρίνεται με το μέγεθος του συστήματος  $L$ . Περαιτέρω αύξηση του  $\xi$  “πνίγεται” από το πεπερασμένο μέγεθος του πλέγματος και παρουσιάζονται τα φαινόμενα επίδρασης πεπερασμένου μεγέθους (finite size effects). Αυτά τα μεγάλα finite size effects προσδιορίζουν την ψευδοκρίσιμη περιοχή. Στην περιοχή αυτή, ποσότητες που απειρίζονται στο θερμοδυναμικό όριο (ή παρουσιάζουν μη αναλυτική συμπεριφορά, λ.χ. απότομη αλλαγή της τιμής της παραγώγου), έχουν πεπερασμένη, μέγιστη τιμή (ή πλησιάζουν προς τη μη αναλυτική συμπεριφορά). Αυτή, καθώς το μέγεθος του συστήματος μεγαλώνει, εκτείνεται σε μια όλο και πιο στενή περιοχή γύρω από την κρίσιμη θερμοκρασία. Οποιαδή-



$L$	$\beta_c(L)$	$\chi_{\max}$	$\beta'_c(L)$	$c_{\max}$
40	0.4308(4)	30.68(4)	0.437(1)	0.5000(20)
60	0.4342(2)	62.5(1)	0.4382(7)	0.5515(15)
80	0.4357(2)	103.5(1)	0.4388(5)	0.5865(12)
100	0.4368(1)	153.3(2)	0.4396(2)	0.6154(18)
120	0.4375(1)	210.9(2)	0.4396(4)	0.6373(20)
140	0.43793(13)	276.2(4)	0.4397(5)	0.6554(18)
160	0.4382(1)	349.0(5)	0.4398(4)	0.6718(25)
200	0.43870(7)	516.3(7)	0.4399(2)	0.6974(17)
500	0.43988(4)	2558(5)	0.44038(8)	0.7953(25)
1000	0.44028(4)	8544(10)	0.44054(8)	0.8542(36)

Πίνακας 14.5: Ο υπολογισμός των ψευδοκρίσιμων θερμοκρασιών  $\beta_c(L)$  και  $\beta'_c(L)$  που προκύπτουν από τον υπολογισμό της θέσης των μέγιστων της μαγνητικής επιδεκτικότητας  $\chi_{\max}$  και της ειδικής θερμότητας  $c_{\max}$  αντίστοιχα. Δίνονται και οι αντίστοιχες τιμές των μέγιστων.

ποτε τιμή και αν πάρουμε στην περιοχή αυτή με συστηματικό τρόπο<sup>24</sup> για αρκετά μεγάλο  $L$  θα πάρουμε τις ασυμπτωτικές συμπεριφορές που δίνονται από τις σχέσεις (14.7)–(14.9). Μία επιλογή που δίνει καλά αποτελέσματα είναι να υπολογίσουμε τις ποσότητες στην ψευδοκρίσιμη θερμοκρασία  $\beta_c(L)$ . Αυτή ορίζεται από την τιμή που παρουσιάζεται το μέγιστο των διακυμάνσεων της παραμέτρου τάξης που εδώ είναι η μαγνήτιση  $\langle m \rangle$

$$\chi(\beta_c(L), L) \equiv \chi_{\max}(L). \quad (14.48)$$

Στην περίπτωση αυτή, στο αριστερό μέλος των εξισώσεων (14.7)–(14.9) είναι οι τιμές κάθε φυσικής ποσότητας για  $\beta = \beta_c(L)$ .

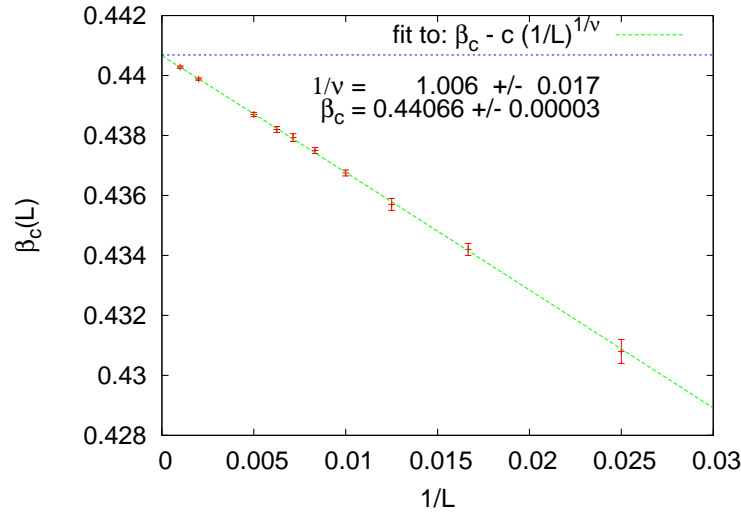
Ο ορισμός της  $\beta_c(L)$  μπορεί να μην είναι μοναδικός. Θα μπορούσε λ.χ. κανείς να χρησιμοποιήσει το μέγιστο της ειδικής θερμότητας

$$c(\beta'_c(L), L) \equiv c_{\max}(L). \quad (14.49)$$

που ορίζει μια διαφορετική  $\beta'_c(L)$ . Προφανώς  $\lim_{L \rightarrow \infty} \beta_c(L) = \lim_{L \rightarrow \infty} \beta'_c(L) = \beta_c$  οπότε και οι δύο επιλογές θα δώσουν τα ίδια αποτελέσματα για αρκετά μεγάλα  $L$ . Δεν συμβαίνει όμως το ίδιο γρήγορα και για τις δύο επιλογές, οπότε είναι συνήθως ευνοϊκή μία από αυτές (εδώ η  $\beta_c(L)$ ).

Το πρώτο βήμα είναι ο υπολογισμός του  $\beta_c$ . Όταν είμαστε στην ψευδοκρίσιμη περιοχή, έχουμε  $\xi \approx L$ , οπότε η βασική σχέση (14.2) δίνει

<sup>24</sup>Όπως λ.χ. κάναμε στην παράγραφο 14.3 όπου υπολογίσαμε κάθε ποσότητα ακριβώς στην  $\beta_c$ .

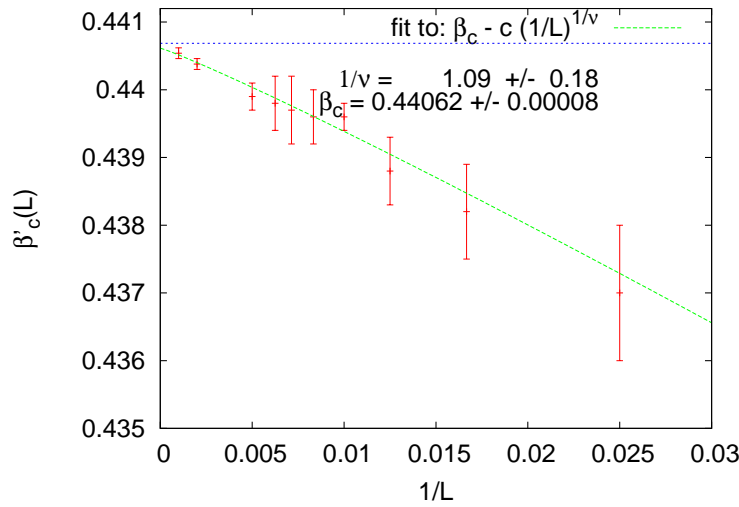


Σχήμα 14.23: Προσδιορισμός της κρίσιμης θερμοκρασίας  $\beta_c$  και του κρίσιμου εκθέτη  $\nu$  από τη σχέση (14.50). Χρησιμοποιώντας τις τιμές των ψευδοκρίσιμων θερμοκρασιών  $\beta_c(L)$  από τον πίνακα 14.5 προσαρμόζουμε τα δεδομένα σε μια συνάρτηση  $a - c(1/L)^b$ . Από τις τιμές που προκύπτουν για τις παραμέτρους  $a$ ,  $b$  και  $c$  υπολογίζουμε τα  $\beta_c = a$ ,  $1/\nu = b$ . Η οριζόντια γραμμή είναι η ακριβής τιμή  $\beta_c = \log(1 + \sqrt{2})/2 = 0.44069 \dots$

$$|t| = \left| \frac{\beta_c - \beta_c(L)}{\beta_c} \right| \sim \xi^{-\frac{1}{\nu}} \sim L^{-\frac{1}{\nu}} \Rightarrow \beta_c(L) = \beta_c - \frac{c}{L^{\frac{1}{\nu}}}. \quad (14.50)$$

Η διαδικασία είναι απλή, ως προς την αρχή της: Μετράμε αρχικά τη μαγνητική επιδεκτικότητα. Για κάθε  $L$  προσδιορίζουμε την ψευδοκρίσιμη περιοχή και υπολογίζουμε τη θέση  $\beta_c(L)$  και την τιμή του μέγιστου  $\chi_{\max}$ . Για να γίνει αυτό μπορεί να χρειαστεί να πάρουμε αρκετές μετρήσεις γύρω από τη  $\beta_c(L)$ . Ιδιαίτερη προσοχή δίνουμε στο να έχουμε αρκετές ανεξάρτητες μετρήσεις μετρώντας τον χρόνο αυτοσυσχετισμού (αυτός αυξάνει με το μέγεθος  $L$  σαν  $\tau \sim L^z$ ) και φυσικά να έχουμε φτάσει στην κατάσταση θερμικής ισορροπίας. Χρησιμοποιούμε τη σχέση (14.50) και προσαρμόζουμε τα δεδομένα μας σε μια συνάρτηση της μορφής  $a - c/L^b$  και από τις τιμές που προκύπτουν για τις σταθερές  $a$ ,  $b$  και  $c$  υπολογίζουμε τις  $\beta_c = a$ ,  $\nu = 1/b$ . Σε περίπτωση που γνωρίζουμε μια από τις σταθερές  $\beta_c$ ,  $\nu$  κρατάμε στην προσαρμογή τις αντίστοιχες παραμέτρους  $a$ ,  $b$  σταθερές και υπολογίζουμε την άλλη.

Τα αποτελέσματα παρουσιάζονται στο σχήμα 14.23 όπου γίνεται η γραφική παράσταση των δεδομένων του πίνακα 14.5. Το αποτέλεσμα



Σχήμα 14.24: Προσδιορισμός της κρίσιμης θερμοκρασίας  $\beta_c$  και του κρίσιμου εκθέτη  $\nu$  από τη σχέση (14.50). Χρησιμοποιώντας τις τιμές των ψευδοκρίσιμων θερμοκρασιών  $\beta'_c(L)$  από τον πίνακα 14.5 προσαρμόζουμε τα δεδομένα σε μια συνάρτηση  $a - c(1/L)^b$ . Από τις τιμές που προκύπτουν για τις παραμέτρους  $a$ ,  $b$  και  $c$  υπολογίζουμε τα  $\beta_c = a$ ,  $1/\nu = b$ . Η οριζόντια γραμμή είναι η ακριβής τιμή  $\beta_c = \log(1 + \sqrt{2})/2 = 0.44069 \dots$

είναι:

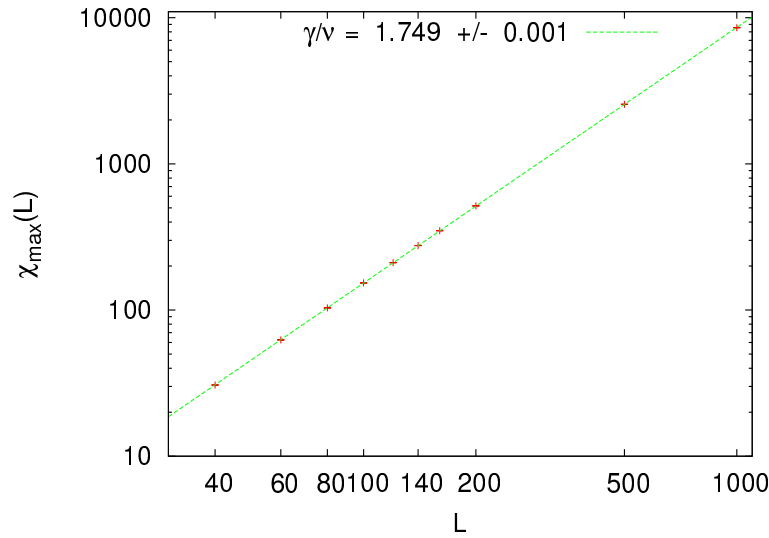
$$\begin{aligned}\beta_c &= 0.44066 \pm 0.00003 \\ \frac{1}{\nu} &= 1.006 \pm 0.017.\end{aligned}$$

Αυτό μπορεί να συγκριθεί με τις γνωστές τιμές  $\beta_c = \log(1 + \sqrt{2})/2 \approx 0.44069$  και  $1/\nu = 1$ .

Η διαδικασία επαναλαμβάνεται για τις ψευδοκρίσιμες θερμοκρασίες  $\beta'_c(L)$  όπου προκύπτουν από τα μέγιστα της ειδικής θερμότητας  $c_{\max}$ . Τα αποτελέσματα παρουσιάζονται στο σχήμα 14.24. Το αποτέλεσμα είναι:

$$\begin{aligned}\beta_c &= 0.44062 \pm 0.00008 \\ \frac{1}{\nu} &= 1.09 \pm 0.18.\end{aligned}$$

Παρατηρούμε από το σχήμα 14.24 και από τα παραπάνω αποτελέσματα ότι η μέτρηση από τη μελέτη της ειδικής θερμότητας δίνει αποτελέσματα συμβατά με τις εξισώσεις (14.51), αλλά με πολύ μικρότερη ακρίβεια και καθαρότητα. Τα μέγιστα της ειδικής θερμότητας είναι πιο ασαφή από αυτά της μαγνητικής επιδεκτικότητας.



Σχήμα 14.25: Υπολογισμός του κρίσιμου εκθέτη  $\gamma/\nu$  από τα μέγιστα της μαγνητικής επιδεκτικότητας, εφαρμόζοντας την ασυμπτωτική σχέση (14.9). Οι τιμές  $\chi_{\max}(L)$  λαμβάνονται από τον πίνακα (14.5) και προσαρμόζονται σε μια συνάρτηση της μορφής  $aL^b$ . Από το αποτέλεσμα της προσαρμογής παίρνουμε  $\gamma/\nu = 1.749(1)$ .

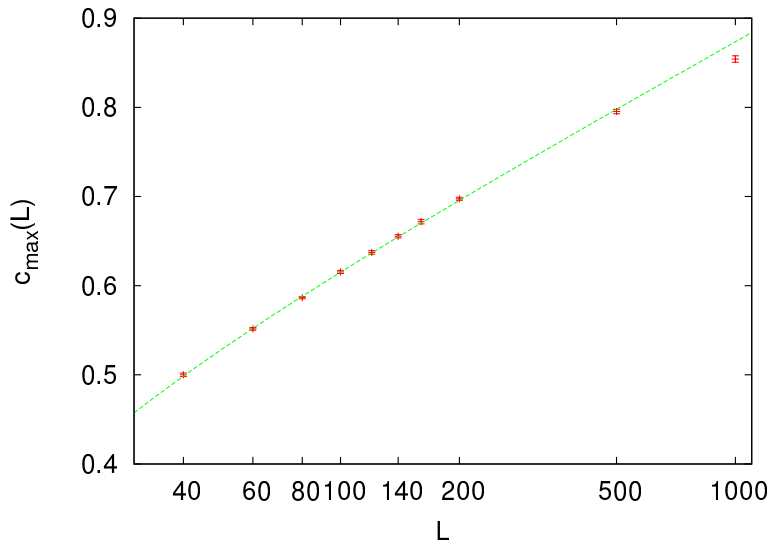
Από τα μέγιστα της μαγνητικής επιδεκτικότητας  $\chi_{\max}(L)$  υπολογίζουμε τον εκθέτη  $\gamma/\nu$ . Οι τιμές τους δίνονται στον πίνακα 14.5. Σύμφωνα με την ασυμπτωτική σχέση (14.9) προσαρμόζουμε τα δεδομένα σε μια συνάρτηση της μορφής  $aL^b$ , με  $a$  και  $b$  προσδιοριστέες παραμέτρους. Βρίσκουμε πολύ καλή βάρθμιση, άρα τα δεδομένα μας είναι μέσα στην ασυμπτωτική περιοχή. Το αποτέλεσμα είναι

$$\frac{\gamma}{\nu} = 1.749 \pm 0.001, \quad (14.51)$$

που είναι σε άριστη συμφωνία με την αναμενόμενη τιμή  $7/4$ .

Από τα μέγιστα της ειδικής θερμότητας υπολογίζουμε τον εκθέτη  $\alpha/\nu$ . Λόγω του ότι  $\alpha = 0$ , η μορφή της ασυμπτωτικής σχέσης αναμένεται να δίνεται από την (14.47). Βρίσκουμε ότι τα δεδομένα μας δεν προσαρμόζονται καλά σε μια απλή συνάρτηση της μορφής  $a \log L$  και είναι πιθανό να παίρνουμε συνεισφορές από όρους που μηδενίζονται στο θερμοδυναμικό όριο, αλλά που δίνουν πεπερασμένη και μη αμελητέα συνεισφορά για πεπερασμένο  $L$ . Πράγματι, προσαρμόζοντας τα δεδομένα σε μια συνάρτηση της μορφής  $a \log L + b - c/L$ , παίρνουμε πολύ καλή προσαρμογή των δεδομένων.<sup>25</sup> Η απόπειρα προσαρμογής σε

<sup>25</sup>Τα αποτελέσματά μας δικαιώνονται και από τους αναλυτικούς υπολογισμούς στο



Σχήμα 14.26: Υπολογισμός του κρίσιμου εκθέτη  $\alpha/\nu$  από τα μέγιστα της ειδικής θερμότητας, εφαρμόζοντας την ασυμπτωτική σχέση (14.8). Οι τιμές  $c_{\max}(L)$  λαμβάνονται από τον πίνακα (14.5) και προσαρμόζονται σε μια συνάρτηση της μορφής  $a \log L + b - c/L$ . Παίρνουμε  $a = 0.107(3)$ ,  $b = 0.13(1)$  και  $c = 1.2(3)$  με  $\chi^2/\text{dof} = 0.9$  για προσαρμογή των σημείων  $L = 40, \dots, 500$ . Αν αποπειραθούμε προσαρμογή κάτω από τις ίδιες συνθήκες σε συνάρτηση της μορφής  $aL^d + b - c/L$  παίρνουμε  $d = 0.004(97)$ , δηλ. εκθέτη συμβατό με την τιμή μηδέν και “περίεργες” τιμές για τις σταθερές  $a, b$ . Συμπεραίνουμε ότι τα δεδομένα μας είναι συμβατά με τον εκθέτη  $\alpha/\nu$  να είναι μηδέν.

συνάρτηση δύναμης της μορφής  $aL^d + b - c/L$  δεν είναι επιτυχής<sup>26</sup> και δίνει έτσι και αλλιώς εκθέτη συμβατό με τιμή μηδέν. Τα αποτελέσματα παρουσιάζονται στο σχήμα 14.26.

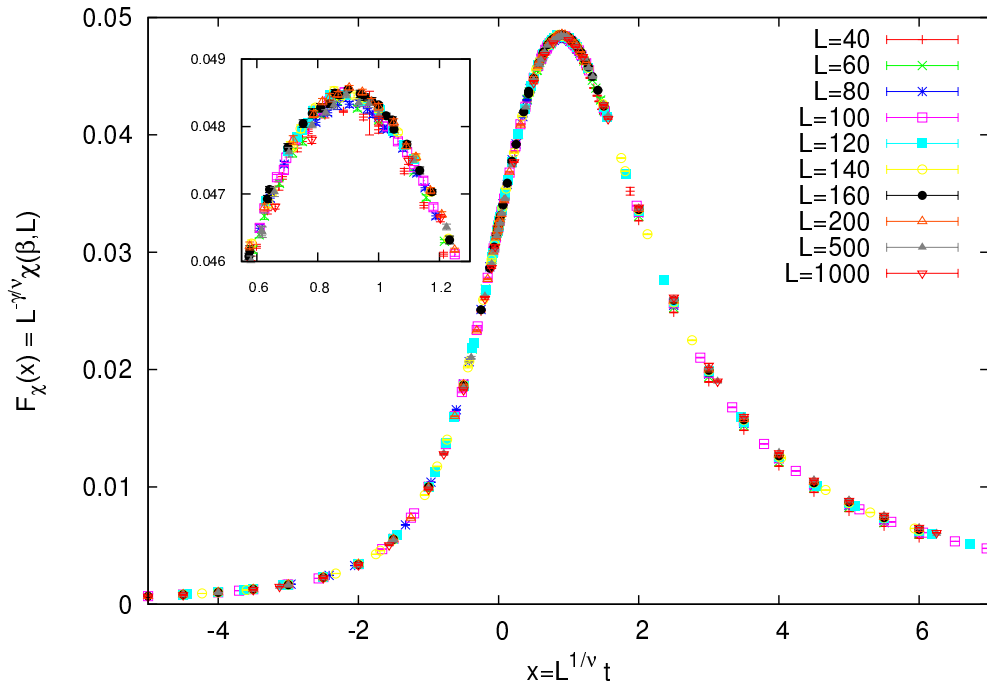
## 14.11 Μελέτη Βάθμισης με ... Κατάρρευση

Οι σχέσεις βάθμισης (14.3)–(14.9) προέρχονται από τη δυναμική εμφάνιση μιας μοναδικής κλίμακας μήκους που απειρίζεται στην κρίσιμη περιοχή, του μήκους συσχετισμού  $\xi$ .<sup>27</sup> Η εμφάνιση αυτής της κλίμακας που απειρίζεται, καθώς πλησιάζουμε το κρίσιμο σημείο σαν  $\xi \sim |t|^{-\nu}$ ,

[69] όπου αποδεικνύεται ότι οι διορθωτικοί όροι είναι ακέραιες δυνάμεις του  $1/L$ :  $c = a \log L + \sum_{k=0}^{\infty} c_k/L^k$ .

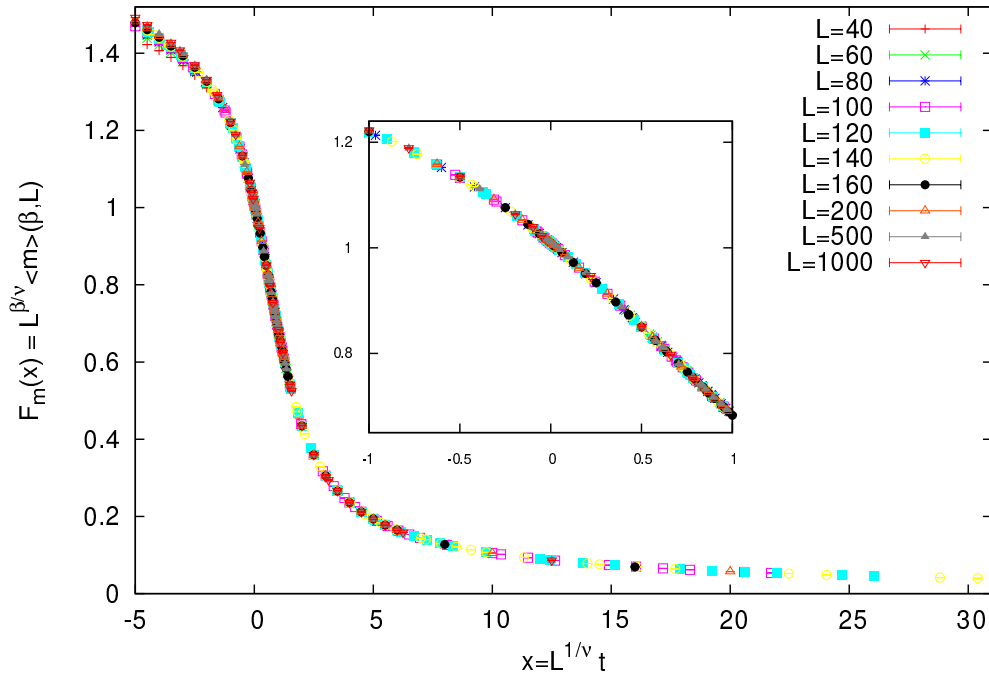
<sup>26</sup>Τα  $a$  και  $b$  δεν προσδιορίζονται μέσα στο σφάλμα, ουσιαστικά ο ένας όρος αναιρεί τον άλλο και μένει μόνο ο τρίτος, το  $c$ .

<sup>27</sup>Προσοχή: εδώ το  $\xi = \xi(t)$  είναι το μήκος συσχετισμού που έχει το άπειρο σύστημα σε θερμοκρασία  $t$  και όχι το μήκος συσχετισμού που μετράμε στο πεπερασμένο σύστημα.



Σχήμα 14.27: “Κατάρρευση” (collapse) των καμπύλων  $\chi(\beta, L)$  για διαφορετικές τιμές του  $L$  σύμφωνα με τη σχέση (14.59). Για το σχήμα έχουν χρησιμοποιηθεί οι γνωστές τιμές  $\beta_c = \ln(1 + \sqrt{2})/2$ ,  $\nu = 1$  και  $\gamma/\nu = 7/4$ .

οδηγεί στην παγκοσμιότητα της κρίσιμης συμπεριφοράς όλων των συστημάτων που ανήκουν στην ίδια κλάση παγκοσμιότητας. Αν πάρουμε για παράδειγμα τη μαγνητική επιδεκτικότητα  $\chi(\beta, L)$ , οι τιμές της εξαρτώνται ξεχωριστά από τη θερμοκρασία  $\beta$ , το μέγεθος του συστήματος  $L$  και φυσικά από τις λεπτομέρειες του συστήματος. Η εμφάνιση παγκόσμιας συμπεριφοράς για ολόκληρη την κλάση ισοδυναμίας που ανήκει το μοντέλο που μελετάμε μας οδηγεί στην υπόθεση ότι η μαγνητική επιδεκτικότητα στην περιοχή αυτή εξαρτάται μόνο από το μήκος συσχετισμού  $\xi$ . Καθώς στο πεπερασμένο σύστημα πλησιάζουμε την κρίσιμη θερμοκρασία, τα φαινόμενα κυριαρχούνται από την “κατάπνιξη” των διακυμάνσεων που προκύπτουν όταν  $\xi \sim L$ . Άρα, οι κλίμακες μήκους που καθορίζουν την κύρια συμπεριφορά των συναρτήσεων που περιγράφουν τη βαθμίσωση  $\chi \sim \xi^{\gamma/\nu}$  είναι οι  $\xi$  και  $L$ , άρα η αδιάστατη μεταβλητή  $L/\xi$  είναι η μοναδική μεταβλητή που υπεισέρχεται στις συναρτήσεις βαθμίσωσης. Για να πάρουμε την σχέση βαθμίσωσης  $\chi \sim \xi^{\gamma/\nu}$  αρκεί



Σχήμα 14.28: “Κατάρρευση” (collapse) των καμπύλων  $\langle m \rangle(\beta, L)$  για διαφορετικές τιμές του  $L$  σύμφωνα με τη σχέση (14.60). Για το σχήμα έχουν χρησιμοποιηθεί οι γνωστές τιμές  $\beta_c = \ln(1 + \sqrt{2})/2$ ,  $\nu = 1$  και  $\beta/\nu = 1/8$ .

να υποθέσουμε ότι κοντά στην κρίσιμη περιοχή<sup>28</sup>

$$\chi = \xi^{\gamma/\nu} F_\chi^{(0)}(L/\xi), \quad (14.52)$$

όπου  $F_\chi^{(0)}(z)$  είναι μια συνάρτηση μιας μεταβλητής με τις ιδιότητες

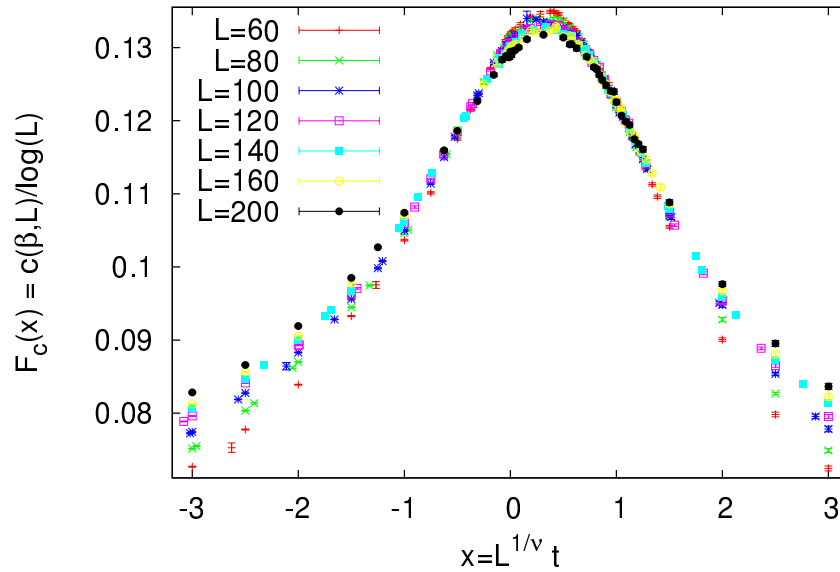
$$F_\chi^{(0)}(z) = \text{σταθ.} \quad z \gg 1, \quad (14.53)$$

και

$$F_\chi^{(0)}(z) \sim z^{\gamma/\nu} \quad z \rightarrow 0. \quad (14.54)$$

Πράγματι όταν  $1 \ll \xi \ll L$  ( $z \gg 1$ ), η μαγνητική επιδεκτικότητα παίρνει τιμές πολύ κοντά στις τιμές του άπειρου συστήματος (θερμοδυναμικό όριο) και από την (14.53) παίρνουμε  $\chi \sim \xi^{\gamma/\nu}$ . Καθώς  $\xi \sim L$ , εμφανίζονται τα φαινόμενα επίδρασης πεπερασμένου μεγέθους και από την (14.54) παίρνουμε  $\chi \sim \xi^{\gamma/\nu} (L/\xi)^{\gamma/\nu} = L^{\gamma/\nu}$ . Η τελευταία δεν είναι άλλη από τη σχέση (14.7) για τα μέγιστα της μαγνητικής επιδεκτικότητας στο

<sup>28</sup>Για περισσότερες λεπτομέρειες παραπέμπουμε στο Παράρτημα 14.13.



Σχήμα 14.29: “Κατάρρευση” (collapse) των καμπύλων  $c(\beta, L)$  για διαφορετικές τιμές του  $L$  σύμφωνα με τη σχέση (14.61). Για το σχήμα έχουν χρησιμοποιηθεί οι γνωστές τιμές  $\beta_c = \ln(1 + \sqrt{2})/2$  και  $\nu = 1$ .

πεπερασμένο σύστημα που μελετήσαμε στο σχήμα 14.25. Άρα, η συνάρτηση  $F_\chi^{(0)}(z)$  περιγράφει τον τρόπο που “αποκόπτεται” η μαγνητική επιδεκτικότητα από τα φαινόμενα επίδρασης πεπερασμένου μεγέθους.

Η συνάρτηση  $F_\chi^{(0)}(z)$  μπορεί να υπολογιστεί από τις μετρήσεις που κάνουμε στην προσομοίωση Μόντε Κάρλο. Επειδή το μήκος συσχέτισμού δεν υπολογίζεται άμεσα, αλλά υπεισέρχεται έμμεσα στις μετρήσεις μας, είναι πιο βολικό να ορίσουμε την αδιάστατη μεταβλητή βάθμισης να είναι η

$$x = L^{1/\nu} t, \quad (14.55)$$

όπου  $|x| \sim (L/\xi)^{1/\nu}$  αφού<sup>29</sup>  $\xi \sim |t|^{-\nu}$ . Ορίζοντας τότε  $F_\chi(x) \propto x^{-\gamma} F_\chi^{(0)}(x^\nu)$ , έτσι ώστε η σχέση (14.52) να γίνει

$$\chi = L^{\gamma/\nu} F_\chi(x) = L^{\gamma/\nu} F_\chi(L^{1/\nu} t). \quad (14.56)$$

Οι ασυμπτωτικές ιδιότητες της συνάρτησης βάθμισης  $F_\chi(x)$  προκύπτουν από τις σχέσεις (14.53) και (14.54). Για  $x = L^{1/\nu} t \gg 1$  για την  $F_\chi^{(0)}(x^\nu)$  ισχύει η (14.53) και παίρνουμε  $F_\chi^{(0)}(x^\nu) = \text{σταθ}$ . Από τον ορισμό  $F_\chi(x) = x^{-\gamma} F_\chi^{(0)}(x^\nu)$  παίρνουμε  $F_\chi(x) \sim x^{-\gamma} = (L/\xi)^{-\gamma/\nu}$  και επιβεβαιώνουμε την

<sup>29</sup> Αγνοούμε την απόλυτη τιμή στον ορισμό της  $x$ , ώστε να έχουμε ενιαίο φορμαλισμό για τις περιοχές πάνω και κάτω από την κρίσιμη θερμοκρασία



ιδιότητα βάθμισης της μαγνητικής επιδεκτικότητας στο θερμοδυναμικό όριο  $\chi \sim L^{\gamma/\nu} F_\chi(x) \sim L^{\gamma/\nu} (L/\xi)^{-\gamma/\nu} = \xi^{\gamma/\nu}$ . Άρα,

$$F_\chi(x) \sim x^{-\gamma} \quad x \gg 1. \quad (14.57)$$

Όταν  $x \rightarrow 0$ , ισχύει η (14.54) και  $F_\chi^{(0)}(x^\nu) \sim (x^\nu)^{\gamma/\nu} = x^\gamma$ . Τότε έχουμε  $F_\chi(x) \propto x^{-\gamma} F_\chi^{(0)}(x^\nu) \sim x^{-\gamma} x^\gamma = \text{σταθ.}$  Επιβεβαιώνουμε ότι όταν επικρατούν τα φαινόμενα επίδρασης πεπερασμένου μεγέθους ( $x \rightarrow 0$ ), παίρνουμε  $\chi = L^{\gamma/\nu} F_\chi(x) \sim L^{\gamma/\nu}$  όπως και παραπάνω. Άρα,

$$F_\chi(x) \sim \text{σταθ.} \quad |x| \ll 1. \quad (14.58)$$

Αντιστρέφοντας τη σχέση (14.56) μπορούμε να δούμε πώς είναι δυνατόν να υπολογιστεί η συνάρτηση βάθμισης από τις μετρήσεις της μαγνητικής επιδεκτικότητας

$$F_\chi(L^{1/\nu}t) = L^{-\gamma/\nu} \chi(\beta, L), \quad (14.59)$$

όπου η  $\chi(\beta, L)$  μετριέται σε θερμοκρασίες κοντά στην κρίσιμη περιοχή για διαφορετικές τιμές του  $L$ . Όταν η σχέση (14.59) ισχύει, τότε όλες οι μετρήσεις μας θα πέφτουν πάνω στην ίδια καμπύλη που δίνει τη συνάρτηση  $F_\chi(x)$  ανεξάρτητα από το μέγεθος του συστήματος  $L$ ! Φυσικά, μικρές αποκλίσεις αναμένονται λόγω φαινομένων επίδρασης πεπερασμένου μεγέθους, ιδιαίτερα για μικρά  $L$ . Αλλά όπως θα δούμε, η σύγκλιση γίνεται στην πράξη αρκετά γρήγορα.

Το μεγάλο κέρδος από την προσέγγιση αυτή στις μετρήσεις μας είναι ότι μπορούμε να χρησιμοποιήσουμε την παραπάνω σχέση για να προσδιορίσουμε την κρίσιμη θερμοκρασία  $\beta_c$ , τον εκθέτη  $\nu$ , καθώς και τον λόγο  $\gamma/\nu$  ταυτόχρονα! Παρατηρήστε ότι για να ισχύει η (14.59) πρέπει να προσδιορίσουμε την μεταβλητή  $x = L^{1/\nu}t$  για την οποία είναι αναγκαίο να γνωρίζουμε την  $\beta_c$  ( $t = (\beta_c - \beta)/\beta_c$ ) και τον εκθέτη  $\nu$ . Για τον υπολογισμό της  $F_\chi$  χρειάζεται να γνωρίζουμε τον εκθέτη  $\gamma/\nu$  που παρουσιάζεται στο δεξί μέλος της (14.59). Η σχέση (14.59) βρίσκεται να είναι πολύ ευαίσθητα εξαρτημένα από τις παραμέτρους  $\beta_c$ ,  $\nu$  και  $\gamma/\nu$  και έτσι έχουμε μια εξαιρετική και ακριβή μέθοδο για τον υπολογισμό τους.

Ο υπολογισμός γίνεται εύκολα. Θέτουμε δοκιμαστικές τιμές για τις παραμέτρους  $(\beta_c, \nu, \gamma/\nu)$ . Από τα δεδομένα μας χρησιμοποιούμε τις τιμές  $L$ ,  $\beta$ ,  $\beta_c$  και  $\nu$  για να υπολογίσουμε τη μεταβλητή βάθμισης  $x = L^{1/\nu}t = L^{1/\nu}(\beta_c - \beta)/\beta_c$ . Αυτή την αντιστοιχούμε στην τιμή της μαγνητικής επιδεκτικότητας  $\chi(\beta, L)$  από την οποία, μαζί με την τιμή του  $\gamma/\nu$ , υπολογίζουμε την  $F_\chi = \chi(\beta, L)/L^{\gamma/\nu}$ . Κάνουμε τη γραφική παράσταση

των σημείων  $(x_i, F_\chi(x_i))$  και επικεντρωνόμαστε στο διάστημα κοντά στην κρίσιμη περιοχή ( $t \approx 0$ ). Αλλάζουμε τις τιμές των παραμέτρων  $(\beta_c, \nu, \gamma/\nu)$  μέχρι να πετύχουμε βέλτιστη σύμπτωση των καμπύλων που προκύπτουν για τα διαφορετικά  $L$  που μελετάμε. Μόλις το πετύχουμε, οι τιμές των  $(\beta_c, \nu, \gamma/\nu)$  που αντιστοιχούν στη βέλτιστη σύμπτωση είναι και οι ζητούμενες.

Η “κατάρρευση” (collapse) των καμπύλων που προκύπτουν από τα σημεία  $(L_i^{1/\nu}(\beta_c - \beta_i)/\beta_c, L_i^{-\gamma/\nu}\chi(\beta_i, L_i))$  για διαφορετικά  $L$  είναι η αποδοτικότερη μέθοδος εκμετάλλευσης των ιδιοτήτων βάθμισης στην κρίσιμη περιοχή. Στο σχήμα 14.27 δείχνουμε τη συνάρτηση  $F_\chi(x)$  για τις γνωστές τιμές των παραμέτρων  $(\beta_c, \nu, \gamma/\nu) = (\ln(1 + \sqrt{2})/2, 1, 7/4)$ . Μικρές μεταβολές από τις τιμές αυτές κάνουν εμφανή την διαφορά αυτή. Μεταβάλλοντας μια από τις παραμέτρους και παρατηρώντας την αποστασιοποίηση των καμπύλων μεταξύ διαφορετικών  $L$ , μπορούμε να πάρουμε μια πρόχειρη εκτίμηση της ελάχιστης ακρίβειας της μεθόδου

$$\begin{aligned}\beta_c &= 0.44069 \pm 0.00001 \\ \nu &= 1.00 \pm 0.01 \\ \frac{\gamma}{\nu} &= 1.750 \pm 0.002,\end{aligned}$$

Παρατηρούμε ότι με αυτή την πρόχειρη μέθοδο πετύχαμε ακρίβεια συγκρίσιμη με αυτή των προηγούμενων (συστηματικών) υπολογισμών μας!

Ανάλογη διαδικασία μπορεί να ακολουθηθεί και για άλλες υπολογιζόμενες ποσότητες που παρουσιάζουν βάθμιση, όπως είναι η ειδική θερμότητα και η μαγνήτιση. Οι σχέσεις (14.8) και (14.9) γενικεύονται σύμφωνα με τα παραπάνω για τη μαγνήτιση<sup>30</sup>

$$\langle m \rangle(\beta, L) = L^{-\beta/\nu} F_m(L^{1/\nu} t), \quad (14.60)$$

και για την ειδική θερμότητα σε

$$c(\beta, L) = L^{\alpha/\nu} F_c(L^{1/\nu} t) = \log(L) F_c(L^{1/\nu} t), \quad (14.61)$$

αφού  $\alpha = 0$ . Τα αποτελέσματα παρουσιάζονται στα σχήματα 14.28 και 14.29 αντίστοιχα.

Για διευκόλυνση του αναγνώστη παραθέτουμε ένα πρόγραμμα gnuplot για την εύκολη κατασκευή σχημάτων, όπως αυτά που παρουσιάζονται στα σχήματα 14.27–14.29. Αν υποθέσουμε ότι τα δεδομένα μας είναι σε ένα αρχείο all τοποθετημένα ως εξής<sup>31</sup>:

<sup>30</sup>Προσοχή στο αριστερό μέλος  $\beta$  είναι η θερμοκρασία, ενώ στο δεξί ο εκθέτης που ορίστηκε στη σχέση βάθμισης (14.5).

<sup>31</sup>Το αρχείο μπορεί να βρεθεί στο συνοδευτικό λογισμικό στο αρχείο all.

```
# #####
# e L beta <e> +/- err c +/- err
# m L beta <m> +/- err chi +/- err
# n L beta <n>/N +/- err
# -----
....
e 1000 0.462721 -0.79839031 7.506e-07 0.290266 0.00027
m 1000 0.462721 0.82648701 1.384e-06 2.137 0.00179
....
```

όπου γραμμές που αρχίζουν με m έχουν τις μετρήσεις  $(L, \beta, \langle m \rangle, \delta \langle m \rangle, \chi, \delta \chi)$ , ενώ αυτές που αρχίζουν από e τις μετρήσεις  $(L, \beta, \langle e \rangle, \delta \langle e \rangle, c, \delta c)$ . Οι εντολές που πρέπει να προγραμματίσουμε γράφονται σε ένα αρχείο `scale_gamma.gpl`:

```
# Usage:
# Ls = "40 60 80 100 120 140 160 200 500 1000"
# bc = bcc; nu = 1 ; gnu = 1.75; load "scale_gamma.gpl";
# Ls: the values of L used in the collapse
# bc: the critical temperature used in the calculation of
#      t=(beta_c-beta)/beta_c
# nu: the exponent used in the calculation of x=L^{1/nu} t
# gnu: the exponent used in the calculation of
#       F_chi = L^{-gnu} chi(beta,L)

#the exact critical temperature (use bc=bcc is you wish):
bcc = 0.5*log(1.0+sqrt(2.0));
NLs = words(Ls); # The number of lattice sizes
LL(i) = word(Ls,i); # Returns the i_th lattice size
cplot(i) = sprintf("\
    <grep 'm %s ' all|\
    sort -k 3,3g|\
    awk -v L=%s -v bc=%f -v nu=%f -v gnu=%f \
    '{ print L^(1.0/nu)*(bc-$3)/bc,L^(-gnu)*$6,L^(-gnu)*$7 }'\
    ",LL(i),LL(i),bc,nu,gnu);

set macros
set term wxt enhanced

set title sprintf("b_c= %f nu= %f g/n= %f",bc,nu,gnu)
set xlabel "x=L^{1/nu} t"
set ylabel "F(x) = L^{-g/n} chi({/Symbol b},L)"

plot for[i=1:NLs] cplot(i) u 1:2:3 w e t sprintf("L=%s",LL(i))
```

Για τη χρήση του προγράμματος, ξεκινάμε το gnuplot και δίνουμε τις εντολές

```
gnuplot> Ls = "40 60 80 100 120 140 160 200 500 1000"
gnuplot> bc = 0.4406868; nu = 1 ; gnu = 1.75;
gnuplot> load "scale_gamma.gpl"
```

Οι δύο πρώτες εντολές καθορίζουν τις παραμέτρους του διαγράμματος. Η μεταβλητή  $Ls$  περιέχει τα μεγέθη του πλέγματος, την ... κατάρρευση των οποίων θέλουμε να μελετήσουμε, κάθε  $L$  χωρισμένο από το άλλο από ένα ή περισσότερα κενά. Η μεταβλητές  $bc$ ,  $nu$ ,  $gnu$  είναι οι παράμετροι  $\beta_c$ ,  $\nu$ ,  $\gamma/\nu$  που θα χρησιμοποιηθούν στη βάρθρωση (14.59). Η τρίτη εντολή καλεί το πρόγραμμά μας και φτιάχνει το διάγραμμα. Αν δεν έχουμε διαλέξει τις επιθυμητές παραμέτρους, τις επαναορίζουμε και... επαναλαμβάνουμε.

Για να καταλάβετε πώς λειτουργεί το πρόγραμμα, δείτε την τεκμηρίωση του `gnuplot`<sup>32</sup>. Θα σταθούμε μόνο στην κατασκευή του φίλτρου που μας δίνει τα σημεία του γραφήματος σωστά κανονικοποιημένα. Το `gnuplot` το βλέπει σαν μια ακολουθία χαρακτήρων (string) `cplot(i)` που είναι διαφορετική για κάθε  $L$  (ο δείκτης  $i$  αντιστοιχεί στην  $i$ -λέξη στη μεταβλητή  $Ls$ ). Το string αυτό τοποθετείται στην εντολή `plot` για κάθε  $i$  και έχει τη μορφή `"< grep ... L(-gnu)*$7}'"`. Οι εκάστοτε τιμές περνούν στο string αυτό μέσω της συνάρτησης `sprintf()` η οποία κάθε φορά καλείται με διαφορετικό  $i$ . Την κύρια δουλειά την κάνει η `awk` που πολλαπλασιάζει την κάθε μέτρηση (στήλες 6 και 7:  $\$6$ ,  $\$7$  είναι η  $\chi$  και το σφάλμα της  $\delta\chi$ ) με τον σωστό παράγοντα. Η συνιστώσα του φίλτρου με το `grep` επιλέγει τις γραμμές από το αρχείο `all` που έχουν τη μαγνητική επιδεκτικότητα για το δεδομένο μέγεθος πλέγματος  $L$ . Η συνιστώσα του φίλτρου με την εντολή `sort` ταξινομεί τα δεδομένα που παίρνει με σειρά αύξουσας θερμοκρασίας (στήλη 3:  $-k3, 3g$ ).

Το καίριο ερώτημα είναι πώς θα συστηματοποιηθεί η παραπάνω μέθοδος; Πώς κρίνουμε ότι μια κατάρρευση είναι καλή και πότε παύει να είναι; Πώς θα εκτιμήσουμε τα σφάλματα στις παραμέτρους που προσπαθούμε να προσδιορίσουμε; Για το τελευταίο ερώτημα προσπαθήσαμε να δώσουμε παραπάνω μια πρόχειρη μέθοδο για την εκτίμηση των σφαλμάτων. Μια άλλη μέθοδος είναι να εφαρμόσουμε μια μέθοδο τύπου `binning` ή `bootstrap`. Στην πρώτη περίπτωση χωρίζουμε τα δεδομένα μας σε  $n_b$  bins και εργαζόμαστε ανεξάρτητα σε κάθε ένα από αυτά. Κάθε bin θα μας δώσει μια βέλτιστη ομάδα παραμέτρων ( $\beta_c, \nu, \gamma/\nu$ ) την οποία θα χρησιμοποιήσουμε σαν μια ανεξάρτητη μέτρηση. Το σφάλμα τότε θα υπολογιστεί από τον γνωστό τύπο (13.42) (για  $n = n_b$ ).

<sup>32</sup> Από την online τεκμηρίωση, μπορείτε να χρησιμοποιήσετε τις εντολές `help word`, `help words`, `help macros`, `help sprintf`, `help plot iteration`.

Για να μελετήσουμε ποσοτικά την ποιότητα της κατάρρευσης ορίζουμε μία ποσότητα που θυμίζει το  $\chi^2/\text{dof}$  που ορίζεται σε μια κοινή προσαρμογή δεδομένων όπως παρουσιάστηκε στο Παράρτημα 13.7. Αυτή θα πρέπει να είναι μεγαλύτερη, όταν αυξάνεται η απόσταση μεταξύ των καμπύλων που προκύπτουν από την κατάρρευση των δεδομένων για κάθε  $L$ . Έστω ότι οι μετρήσεις μας αποτελούνται από  $N_L$  σετ μετρήσεων για  $L = L_1, L_2, \dots, L_{N_L}$ . Αφού επιλέξουμε ένα σετ παραμέτρων  $\mathbf{p} \equiv (\beta_c, \nu, \gamma/\nu)$  και ένα διάστημα  $\Delta x \equiv [x_{\min}, x_{\max}]$  υπολογίζουμε τα σετ των δεδομένων  $\{(x_{i,k}, F_\chi(x_{i,k}; \mathbf{p}, L_i))\}_{k=1, \dots, n_i}$  που προκύπτουν από τις μετρήσεις μας. Αυτά αποτελούνται από τα  $n_i$  σημεία στο σετ δεδομένων με  $L = L_i$  των οποίων τα  $x_k$  βρίσκονται μέσα στο διάστημα  $\Delta x$ . Για κάθε σημείο  $x_k$  υπολογίζουμε την προκύπτουσα συνάρτηση βάθμισης  $F_\chi(x_{i,k}; \mathbf{p}, L_i) = L_i^{-\gamma/\nu} \chi(\beta_{i,k}, L_i)$  η οποία εξαρτάται από τις επιλεχθείσες παραμέτρους  $\mathbf{p}$  και το επιλεγμένο μέγεθος  $L_i$ . Στη συνέχεια, επιλέγουμε έναν τρόπο παρεμβολής (interpolation) μεταξύ των παραπάνω σημείων η οποία ορίζει συνεχείς συναρτήσεις παρεμβολής  $F_\chi(x; \mathbf{p}, L_i)$ ,<sup>33</sup> ώστε να έχουμε μια καλή εκτίμηση της συνάρτησης βάθμισης μεταξύ των διακριτών σημείων που αποτελούν τα δεδομένα μας. Τότε κάθε σημείο στο σετ δεδομένων  $\{(x_{i,k}, F_\chi(x_{i,k}; \mathbf{p}, L_i))\}_i$  έχει δεδομένη απόσταση από κάθε άλλο σετ δεδομένων  $j$  ( $j = 1, \dots, N_L$  και  $j \neq i$ ) ίση με<sup>34</sup>  $|F_\chi(x_{i,k}; \mathbf{p}, L_i) - F_\chi(x_{i,k}; \mathbf{p}, L_j)|$ . Ορίζεται τότε η ποσότητα

$$\chi^2(\mathbf{p}; \Delta x) = \frac{1}{N_L(N_L - 1)n_{\text{points}}} \times \sum_{i=1}^{N_L} \sum_{\{j=1, j \neq i\}}^{N_L} \sum_{k=1}^{n_i} \frac{(F_\chi(x_{i,k}; \mathbf{p}, L_i) - F_\chi(x_{i,k}; \mathbf{p}, L_j))^2}{(\delta F_\chi(x_{i,k}; \mathbf{p}, L_i))^2} \quad (14.62)$$

όπου  $n_{\text{points}} = \sum_{i=1}^{N_L} n_i$  είναι ο αριθμός των όρων στο άθροισμα. Η κανονικοποίηση με  $N_L(N_L - 1)$  γίνεται, γιατί αυτός είναι ο αριθμός που οι καμπύλες συνδυάζονται ανά ζεύγη μεταξύ τους. Κάθε όρος ζυγίζεται με το σφάλμα του  $\delta F_\chi(x_{i,k}; \mathbf{p}, L_i)$  (που προέρχεται από το  $\delta\chi$ ), ώστε τα σημεία με μεγάλο σφάλμα να έχουν μικρότερη συνεισφορά από αυτά με μεγαλύτερο. Ο ορισμός αυτός χρησιμοποιήθηκε στο [70], αλλά ο αναγνώστης μπορεί να δει και εναλλακτικές προσεγγίσεις στα [4], [68].

<sup>33</sup> Αυτή είναι ίσως και η πιο σημαντική επιλογή που θα κάνουμε. Πολυωνυμική προσαρμογή (συμπ. και γραμμικής), παρεμβολή τύπου cspline ή γενίκευσης της, multihistogramming είναι δυνατές επιλογές. Πιο απαλλαγμένη από συστηματικά σφάλματα είναι η τελευταία.

<sup>34</sup> Εδώ φαίνεται η ανάγκη της παρεμβολής, αφού η τιμή  $x_{i,k}$  μπορεί να μην υπάρχει στο σετ δεδομένων  $j$ .

Το  $\chi^2(\mathbf{p}; \Delta x)$  εξαρτάται από τις παραμέτρους  $\mathbf{p}$  και το διάστημα  $\Delta x$ . Γίνεται καταρχήν μια ελαχιστοποίηση του ως προς τις παραμέτρους  $\mathbf{p}$  κρατώντας το  $\Delta x$  σταθερό. Το ελάχιστο δίνεται από τις τιμές των παραμέτρων  $\mathbf{p}_{\min}$  που είναι και η εκτίμηση των τιμών που θέλουμε να υπολογίσουμε. Για τον υπολογισμό των σφαλμάτων  $\delta\mathbf{p}$  η πιο συστηματική διαδικασία είναι η μέθοδος binning όπως αναφέραμε και παραπάνω στη σελίδα 660. Εναλλακτικά, αν υποθέσουμε μια κατανομή των μετρήσεων τύπου  $\chi^2$  και το αποτέλεσμα μας δίνει  $\chi^2 \lesssim 1$ , τότε τα διαστήματα των παραμέτρων μέσα στα όρια του σφάλματος είναι αυτά για τα οποία  $\chi^2 \lesssim 2$ .

Τα αποτελέσματα εξαρτώνται από το διάστημα  $\Delta x$ . Συνήθως [68] αυτό επιλέγεται να έχει το κέντρο του στο μέγιστο της  $F_\chi(x)$ , έτσι ώστε  $\Delta x = [x_{\max} - \delta x, x_{\max} + \delta x]$ . Αν το  $\delta x$  είναι μεγαλύτερο από όσο πρέπει, τότε το  $\chi^2(\mathbf{p}_{\min}; \Delta x)$  είναι μεγάλο, γιατί δεν έχουμε καλή βάρμηση. Αν είναι πολύ μικρό, τότε τα σφάλματα  $\delta\mathbf{p}$  θα είναι μεγάλα. Παίρνοντας το όριο  $\delta x \rightarrow 0$ , μελετάμε τη σύγκλιση των τιμών  $\mathbf{p}_{\min}$  (βλ. σχήμα 8.7, σελ. 238 στο [4] καθώς και το [68]) από όπου καταλήγουμε στις τελικές τιμές του υπολογισμού μας.

## 14.12 Binder Cumulant

Μέχρι τώρα μελετήσαμε τις διακυμάνσεις των φυσικών ποσοτήτων από τον υπολογισμό cumulant<sup>35</sup> δεύτερης τάξης (μαγνητική επιδεκτικότητα και ειδική θερμότητα). Σημαντική πληροφορία για τους κρίσιμους εκθέτες και την κρίσιμη θερμοκρασία μπορεί να ληφθεί από τη μελέτη των cumulant ανωτέρας τάξης. Ο πιο γνωστός, ο Binder cumulant, είναι τέταρτης τάξης<sup>36</sup> και παίρνει το όνομα του Kurt Binder που τον μελέτησε πρώτος [71, 72]

$$U = 1 - \frac{\langle m^4 \rangle}{3\langle m^2 \rangle^2}. \quad (14.63)$$

Στο παράρτημα 14.13 παρουσιάζονται με λεπτομέρεια οι ιδιότητές του. Αυτές που μας αφορούν άμεσα είναι πως σε μια συνεχή μετάβαση φάσης

<sup>35</sup>

<http://en.wikipedia.org/wiki/Cumulant>,

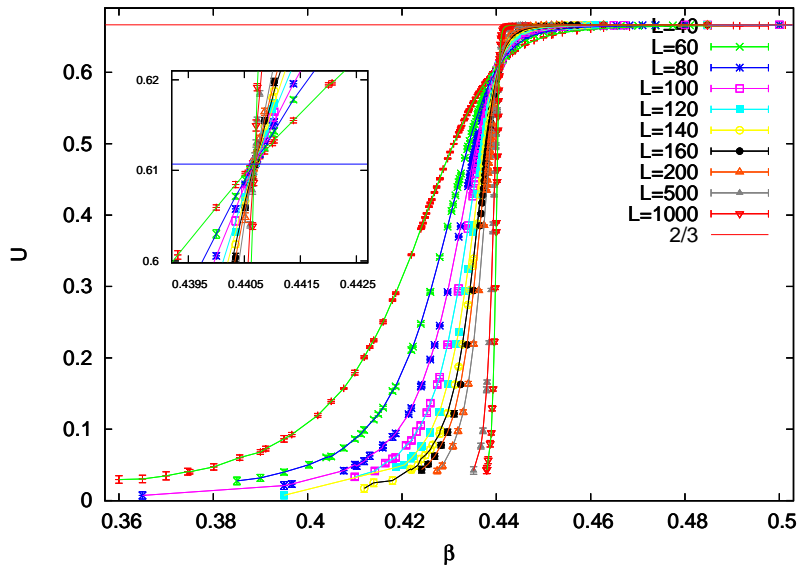
<http://mathworld.wolfram.com/Cumulant-GeneratingFunction.html>

<sup>36</sup>Στη στατιστική το cumulant 4ης τάξης μιας τυχαίας μεταβλητής  $x$  ισούται με  $\kappa_4 = \langle (x - \langle x \rangle)^4 \rangle - 3\langle (x - \langle x \rangle)^2 \rangle^2$  και  $\kappa_2 = \langle (x - \langle x \rangle)^2 \rangle$ , οπότε  $U = -\kappa_4/3\kappa_2^2$  για  $x = m$  και  $\langle m \rangle = 0$ .

$$U = \begin{cases} 0 & \beta \ll \beta_c \\ U^* & \beta = \beta_c \\ \frac{2}{3} & \beta \gg \beta_c \end{cases}, \quad (14.64)$$

όπου  $U^* = 0.610690(1)$  για το πρότυπο Ising στο τετραγωνικό πλέγμα [72]. Η τιμή  $U = 0$  ταυτίζεται με αυτή της γκαουσιανής κατανομής, ενώ  $U = 2/3$  αντιστοιχεί περίπου με την γκαουσιανή κατανομή μικρής διασποράς γύρω από δύο συμμετρικές τιμές  $\pm \langle m \rangle$  (βλ. ασκήσεις 14 και 15 στο Παράρτημα 14.13).

Στην πράξη, βρίσκεται πως οι διορθώσεις στην τιμή  $U^*$  από φαινόμενα επίδρασης πεπερασμένου μεγέθους είναι πολύ μικρές, οπότε ο υπολογισμός του  $U(\beta, L)$  δίνει με πολύ καλή ακρίβεια την κρίσιμη θερμοκρασία  $\beta_c$ . Το σημείο τομής των  $U(\beta, L)$  στο σημείο  $(\beta_c, U^*)$  για διαφορετικά  $L$  θα δίνει μια πολύ καλή εκτίμηση της  $\beta_c$ .



Σχήμα 14.30: Ο Binder cumulant για διαφορετικές θερμοκρασίες και πλεγματικά μεγέθη. Στο ένθετο μεγεθύνεται η κρίσιμη περιοχή. Η οριζόντια γραμμή είναι η αναμενόμενη τιμή  $U^* = 0.610690(1)$  [72].

Στο σχήμα 14.30 δείχνουμε τις μετρήσεις μας για το  $U(\beta, L)$ . Φαίνεται η εντυπωσιακή τομή των καμπύλων στο κρίσιμο σημείο  $(\beta_c, U^*)$ . Στον πίνακα 14.6 κάνουμε μια απόπειρα για συστηματικό υπολογισμό της  $\beta_c$  υπολογίζοντας την κρίσιμη θερμοκρασία από την τομή των  $U(\beta, L)$  για τρεις τιμές του  $L$ . Λαμβάνοντας υπόψη όλες τις μετρήσεις

$L$			$\beta_c$	$U^*$
40	60	80	0.44069(4)	0.6109(5)
60	80	100	0.44069(4)	0.6108(7)
80	100	120	0.44068(4)	0.6108(7)
100	120	140	0.44069(4)	0.6108(11)
120	140	160	0.44069(4)	0.6109(20)
140	160	200	0.44067(3)	0.6102(12)
160	200	500	0.44067(2)	0.6105(10)
200	500	1000	0.44068(1)	0.6106(9)

Πίνακας 14.6: Υπολογισμός της κρίσιμης θερμοκρασίας  $\beta_c$  και της τιμής  $U^*$  από την τομή των καμπύλων  $U(\beta, L)$  που παρουσιάζονται στο σχήμα 14.30. Για τον υπολογισμό χρησιμοποιούνται κάθε φορά τρεις τιμές του  $L$ . Οι αναμενόμενες τιμές από τη θεωρία και τη βιβλιογραφία [72] είναι  $\beta_c = 0.44068679\dots$ ,  $U^* = 0.610690(1)$ .

για  $L = 100 - 1000$ , το αποτέλεσμα που παίρνουμε είναι

$$\beta_c = 0.440678(9) \quad U^* = 0.6107(4), \quad (14.65)$$

που είναι σε πολύ καλή συμφωνία με τις αναμενόμενες τιμές  $\beta_c = 0.44068679\dots$ ,  $U^* = 0.610690(1)$ . Παρατηρούμε ότι στον υπολογισμό του  $U^*$ , ενώ το συστηματικό σφάλμα λόγω πεπερασμένου μεγέθους μειώνεται, καθώς αυξάνουμε το  $L$ , αυξάνεται το στατιστικό λόγω της αύξησης της κλίσης της καμπύλης  $U(\beta, L)$  κοντά στο σημείο  $\beta = \beta_c$ . Ο προσδιορισμός όμως της  $\beta_c$  γίνεται με αυξανόμενη ακρίβεια.

Οι σχέσεις βάρθμισης πεπερασμένου μεγέθους μπορούν να χρησιμοποιηθούν και για τον Binder cumulant προκειμένου να προσδιοριστεί και έτσι η κρίσιμη θερμοκρασία και ο κρίσιμος εκθέτης  $1/\nu$ . Από τις σχέσεις (14.90) (14.119) του Παραρτήματος 14.13 περιμένουμε ότι η βάρθμιση του  $U$  δίνεται από

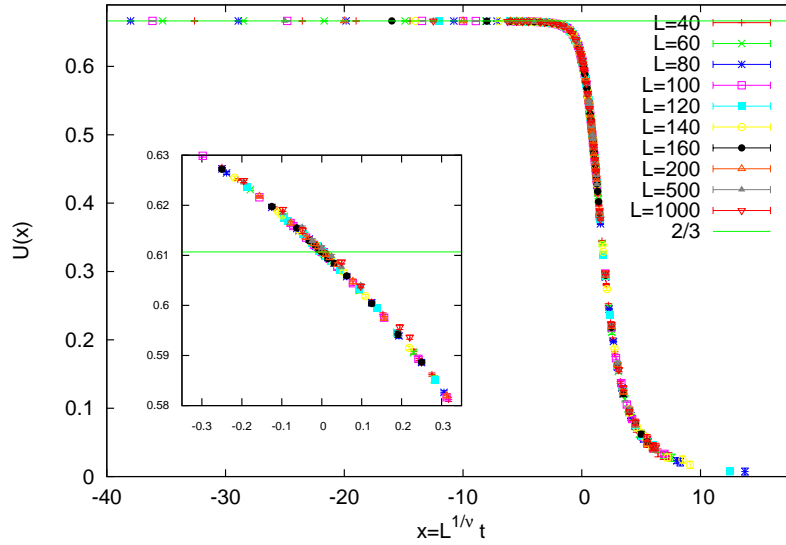
$$U = F_U(x) = F_U(L^{1/\nu}t). \quad (14.66)$$

Αυτό επιβεβαιώνεται εντυπωσιακά στο σχήμα 14.31. Από την τιμή  $F_U(x = 0)$  παίρνουμε  $U^* = 0.6107(4)$  σε συμφωνία με την (14.65).

Ο απευθείας υπολογισμός των κρίσιμων εκθετών, ιδιαίτερα του  $1/\nu$ , είναι αρκετά δύσκολος σε ένα γενικό υπό μελέτη σύστημα. Οπότε είναι θεμιτό να πάρει κανείς επιπλέον εκτιμήσεις από ποσότητες που είναι εύκολο να μετρηθούν και έχουν γνωστές και καλές ιδιότητες βάρθμισης. Παρακάτω παραθέτουμε μερικές από αυτές<sup>37</sup>. Χαρακτηριστικό τους εί-

<sup>37</sup>Ιδιαίτερη επιτυχία είχαν στη μελέτη του 3d πρότυπου Ising [74].





Σχήμα 14.31: Βάθμιση του Binder cumulant για  $1/\nu = 1$  χρησιμοποιώντας την ακριβή τιμή της  $\beta_c$  στην ανηγμένη θερμοκρασία  $t = (\beta_c - \beta)/\beta_c$ . Στο ένθετο μεγεθύνεται η κρίσιμη περιοχή. Η οριζόντια γραμμή είναι η αναμενόμενη τιμή  $U^* = 0.610690(1)$  [72].

ναι ότι δίνονται από το συσχετισμό της τιμής της ενέργειας με την τιμή (κάποιας δύναμης της) μαγνήτισης.

Η παράγωγος του Binder Cumulant είναι

$$D_U = \frac{\partial U}{\partial \beta} = \frac{\langle m^4 E \rangle \langle m^2 \rangle + \langle m^4 \rangle (\langle m^2 \rangle \langle E \rangle - 2 \langle m^2 E \rangle)}{3 \langle m^2 \rangle^3} \quad (14.67)$$

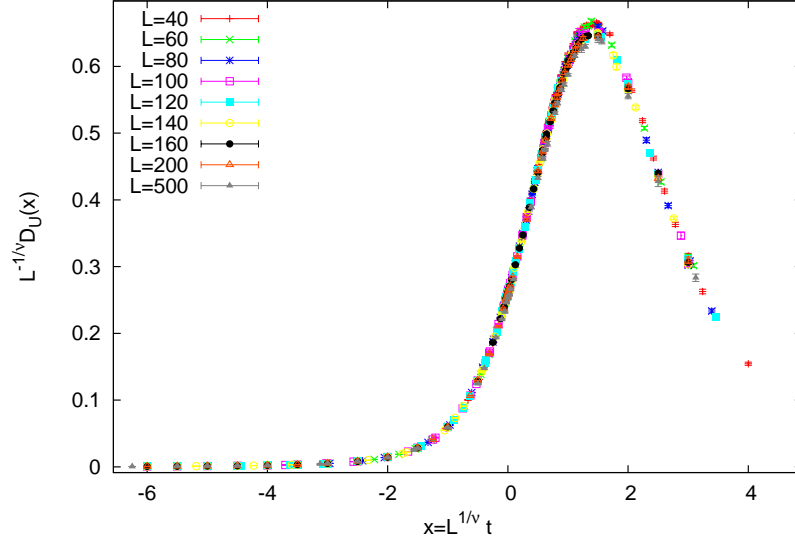
Η βάθμισή του δίνεται από τη σχέση (14.120)

$$D_U = L^{1/\nu} F_{D_U}(x) = L^{1/\nu} F_{D_U}(L^{1/\nu} t). \quad (14.68)$$

η οποία αναπαρίσταται στο σχήμα 14.32. Παρατηρούμε ότι η  $D_U$  ορίζει και αυτή μια ψευδοκρίσιμη περιοχή από το μέγιστο των αντίστοιχων διακυμάνσεων. Η βάθμιση και θέση του μεγίστου θα μπορούσε να μας δώσει τον κρίσιμο εκθέτη  $1/\nu$  σύμφωνα με την ανάλυση που κάναμε στα σχήματα 14.23 και 14.25 για τη μαγνητική επιδεκτικότητα.

Επίσης, μπορεί να φανεί χρήσιμη η μελέτη συναρτήσεων συσχετισμού της μορφής

$$D_{\ln m^n} = \frac{\partial \ln \langle m^n \rangle}{\partial \beta} = \langle E \rangle - \frac{\langle E m^n \rangle}{\langle m^n \rangle}, \quad (14.69)$$



Σχήμα 14.32: Βάθμιση της παραγώγου του Binder cumulant  $D_U$  (βλ. εξίσωση (14.67)) για  $1/\nu = 1$  χρησιμοποιώντας την ακριβή τιμή της  $\beta_c$  στην ανηγμένη θερμοκρασία  $t = (\beta_c - \beta)/\beta_c$ .

των οποίων η βάθμιση δίνεται από την εξίσωση (14.126) του Παραρτήματος 14.13

$$D_{\ln m^n} = L^{1/\nu} F_{D_{\ln m^n}}(x) = L^{1/\nu} F_{D_{\ln m^n}}(L^{1/\nu} t). \quad (14.70)$$

Ειδικότερα, μας ενδιαφέρει η περίπτωση  $n = 1$

$$D_{\ln |m|} = \frac{\partial \ln \langle |m| \rangle}{\partial \beta} = \langle E \rangle - \frac{\langle E |m| \rangle}{\langle |m| \rangle}, \quad (14.71)$$

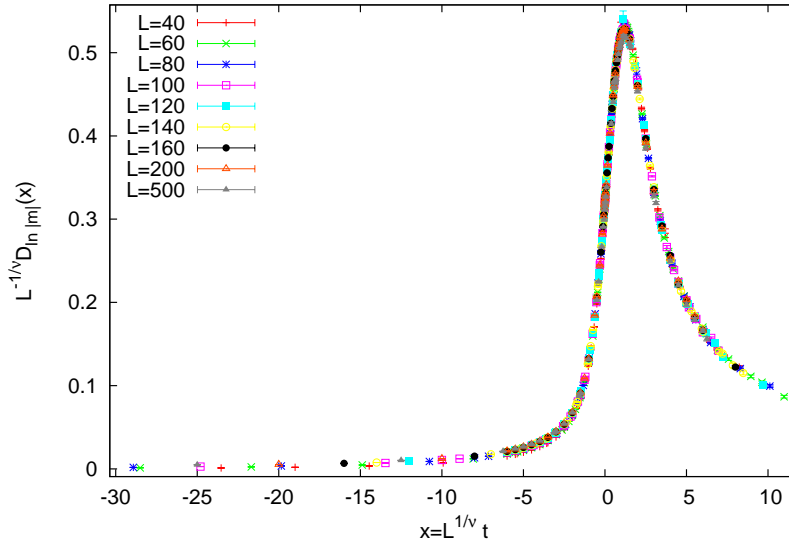
και  $n = 2$

$$D_{\ln m^2} = \frac{\partial \ln \langle m^2 \rangle}{\partial \beta} = \langle E \rangle - \frac{\langle E m^2 \rangle}{\langle m^2 \rangle}. \quad (14.72)$$

Τέλος, αναφέρουμε το Energy Cumulant  $V$

$$V = 1 - \frac{\langle e^4 \rangle}{3 \langle e^2 \rangle^2}. \quad (14.73)$$

Στην εργασία [75] δείχνεται ότι σε μια μετάβαση φάσης δεύτερης τάξης στο κρίσιμο σημείο  $V^* = 2/3$ , ενώ σε μια μετάβαση φάσης πρώτης τάξης παίρνουμε μη τετριμμένη τιμή. Έτσι, αυτή η ποσότητα μπορεί



Σχήμα 14.33: Βάθμιση της  $D_{\ln|m|}$  (βλ. εξίσωση (14.71)) για  $1/\nu = 1$  χρησιμοποιώντας την ακριβή τιμή της  $\beta_c$  στην ανηγμένη θερμοκρασία  $t = (\beta_c - \beta)/\beta_c$ .

να χρησιμοποιηθεί ως παράμετρος διαφοροποίησης μιας μετάβασης φάσης 1ης τάξης. Αυτό επιβεβαιώνεται στο σχήμα 14.35. Τα ελάχιστα των καμπύλων  $V(\beta, L)$  τείνουν προς την κρίσιμη θερμοκρασία σύμφωνα με την σχέση (14.50).

## 14.13 Παράρτημα: Βάθμιση

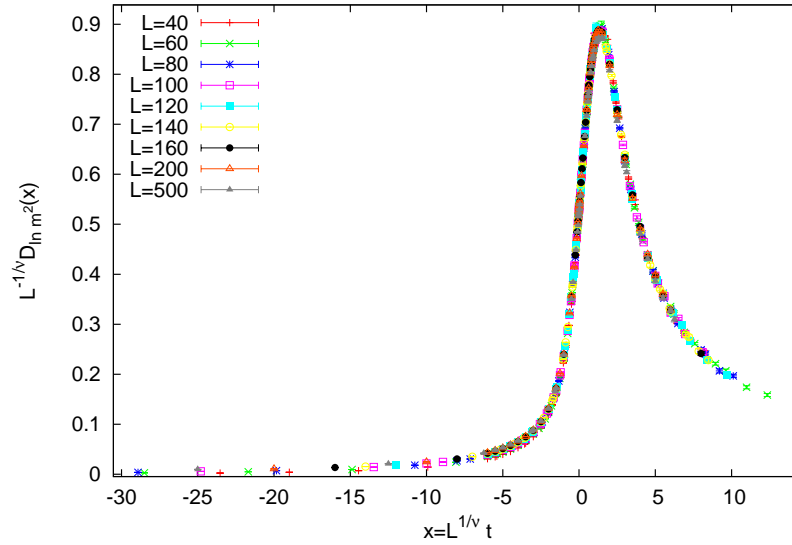
### 14.13.1 Binder Cumulant

Στο κεφάλαιο 14.12 μελετήσαμε αριθμητικά τις ιδιότητες βάθμισης της ποσότητας

$$U = 1 - \frac{\langle m^4 \rangle}{3\langle m^2 \rangle^2}, \quad (14.74)$$

γνωστής στη βιβλιογραφία ως Binder Cumulant. Εδώ χρησιμοποιώντας βασικές ιδιότητες βάθμισης ενός συστήματος κοντά στην κρίσιμη θερμοκρασία θα υπολογίσουμε τις ιδιότητες βάθμισης της ποσότητας αυτής και των παραγώγων της. Για περισσότερες λεπτομέρειες παραπέμπουμε τον αναγνώστη στα [72], [6]

Οι τιμές του  $U$  είναι απλές και τετριμμένες σε δύο περιπτώσεις: Όταν έχουμε γκαουσιανή κατανομή της μαγνήτισης, κάτι το οποίο συμβαίνει



Σχήμα 14.34: Βάθμιση της  $D_{\ln m^2}$  (βλ. εξίσωση (14.72)) για  $1/\nu = 1$  χρησιμοποιώντας την ακριβή τιμή της  $\beta_c$  στην ανηγμένη θερμοκρασία  $t = (\beta_c - \beta)/\beta_c$ .

όταν είμαστε στη φάση αταξίας (ψηλή θερμοκρασία)  $U = 0$  και όταν είμαστε στη φάση τάξης (χαμηλή θερμοκρασία)  $U = 2/3$ . Η απόδειξη είναι πολύ εύκολη και τη σκιαγραφούμε στις ασκήσεις 14 και 15.

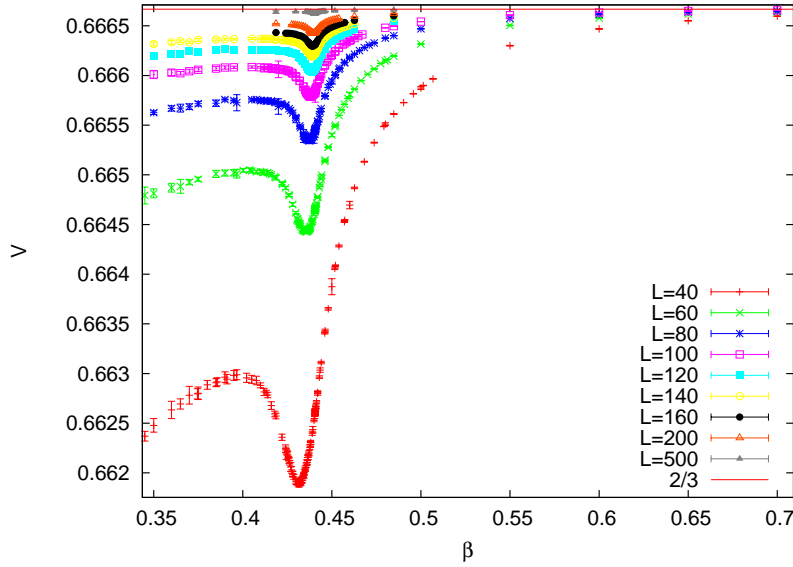
Όταν πλησιάζουμε την κρίσιμη θερμοκρασία μιας συνεχούς μετάβασης φάσης, το σύστημα παρουσιάζει ιδιότητες βάθμισης λόγω του απειριζόμενου μήκους συσχετισμού  $\xi$ , όπως περιγράψαμε στο κεφάλαιο 14. Ας υποθέσουμε ότι πλησιάζουμε από τις ψηλές θερμοκρασίες και η συνάρτηση κατανομής της μαγνήτισης ανά πλεγματοτική θέση  $s$  (όχι τις απόλυτες τιμές) είναι προσεγγιστικά της μορφής

$$P(L, s) = \frac{1}{\sqrt{2\pi\langle s^2 \rangle}} e^{-\frac{s^2}{2\langle s^2 \rangle}} = \left( \frac{\beta L^d}{2\pi\chi} \right)^{\frac{1}{2}} e^{-s^2 \frac{L^d \beta}{2\chi}}, \quad (14.75)$$

πού είναι μια γκαουσιανή με τυπική απόκλιση  $\sigma^2 = \langle s^2 \rangle = \chi/(\beta L^d)$  και προσωρινά έχουμε υποθέσει ότι το σύστημα είναι πάνω σε  $d$ -διάστατο υπερκυβικό πλέγμα με ακμή υπερκύβου  $L$ .

Όταν πλησιάζουμε την κρίσιμη θερμοκρασία, η βάθμιση παρουσιάζεται στη συνάρτηση κατανομής  $P(L, s)$  σύμφωνα με τη σχέση

$$P(L, s) = L^x p_0 \tilde{P}(aL^y s, \frac{L}{\xi}), \quad (14.76)$$



Σχήμα 14.35: Ο Energy Cumulant που ορίζεται στην εξίσωση (14.73). Καθώς αυξάνεται το μέγεθος του συστήματος, η τιμή του τείνει στην τιμή  $2/3$  όπως αναμένεται σε μια μετάβαση φάσης 2ης τάξης. Η θέση των ελαχίστων τείνουν στην κρίσιμη θερμοκρασία σαν  $L^{-1/\nu}$

όπου  $\xi = \xi(t) = \lim_{L \rightarrow \infty} \xi(\beta, L)$ ,  $t = (\beta_c - \beta)/\beta_c$ , είναι το μήκος συσχετισμού στο θερμοδυναμικό όριο. Καθώς πλησιάζουμε την κρίσιμη θερμοκρασία  $\lim_{t \rightarrow 0} \xi(t) = +\infty$ , έτσι ώστε  $\xi \sim |t|^{-\nu}$ . Η εξίσωση (14.76) είναι μια υπόθεση κλιμάκωσης (scaling hypothesis) η οποία παίζει θεμελιώδη ρόλο στη μελέτη των κρίσιμων φαινομένων.

Για να υπολογίσουμε τους εκθέτες στην παραπάνω σχέση, εφαρμόζουμε καταρχήν τη συνθήκη κανονικοποίησης της πιθανότητας

$$\begin{aligned}
 1 &= \int_{-\infty}^{+\infty} ds P(L, s) \\
 &= L^x p_0 \int_{-\infty}^{+\infty} ds \tilde{P}(aL^y s, \frac{L}{\xi}) \\
 &= L^x p_0 \frac{1}{aL^y} \int_{-\infty}^{+\infty} dz \tilde{P}(z, \frac{L}{\xi}) \\
 &= L^{x-y} p_0 \frac{1}{a} \int_{-\infty}^{+\infty} dz \tilde{P}(z, \frac{L}{\xi}), \tag{14.77}
 \end{aligned}$$

όπου θέσαμε  $z = aL^y s$ . Για να είναι μονάδα το αριστερό μέλος πρέπει

$x = y$ ,

$$C_0 \equiv \int_{-\infty}^{+\infty} dz \tilde{P}(z, \frac{L}{\xi}) < \infty \quad (14.78)$$

και  $p_0 = a/C_0$ .  $C_0 = C_0(L/\xi)$ ,  $p_0 = p_0(L/\xi)$  και  $p_0 C_0 = a$  είναι μία σταθερά ανεξάρτητη από τα  $L$  και  $\xi$ . Τελικά παίρνουμε

$$P(L, s) = \frac{a}{C_0} L^y \tilde{P}(aL^y s, \frac{L}{\xi}). \quad (14.79)$$

Οι ροπές της κατανομής των σπιν  $\langle s^k \rangle$  θα είναι τότε

$$\begin{aligned} \langle s^k \rangle &= \int_{-\infty}^{+\infty} ds s^k P(L, s) \\ &= \frac{a}{C_0} L^y \int_{-\infty}^{+\infty} ds s^k \tilde{P}(aL^y s, \frac{L}{\xi}) \\ &= \frac{a}{C_0} L^y \frac{1}{a^{k+1} L^{(k+1)y}} \int_{-\infty}^{+\infty} dz z^k \tilde{P}(z, \frac{L}{\xi}) \\ &= L^{-ky} F_k \left( \frac{L}{\xi} \right). \end{aligned} \quad (14.80)$$

Όταν πάρουμε το θερμοδυναμικό όριο και μετά πλησιάσουμε την κρίσιμη θερμοκρασία, τότε έχουμε αποκλίνον μήκος συσχετισμού  $\xi \rightarrow \infty$  με  $L/\xi \rightarrow \infty$ . Στην περιοχή  $\beta < \beta_c$  ( $\langle m \rangle = 0$ ) έχουμε  $\chi = \beta L^d \langle s^2 \rangle$  και από τη σχέση

$$\left. \begin{aligned} \chi &= \chi_+ t^{-\gamma} \\ \xi &= \xi_+ t^{-\nu} \end{aligned} \right\} \Rightarrow \chi = \frac{\chi_+}{\xi_+^{\gamma/\nu}} \xi^{\gamma/\nu} \sim \xi^{\gamma/\nu} \quad (14.81)$$

παίρνουμε

$$\langle s^2 \rangle = \beta^{-1} L^{-d} \chi = \frac{\chi_+}{\beta L^d \xi_+^{\gamma/\nu}} \xi^{\gamma/\nu} \sim \xi^{\gamma/\nu}. \quad (14.82)$$

Στις παραπάνω σχέσεις εισάγαμε τα “παγκόσμια” πλάτη (universal amplitudes)  $\chi_+$  και  $\xi_+$  τα οποία είναι παγκόσμιες σταθερές (ίδιες στην ίδια κλάση παγκοσμιότητας) και ορίζονται από τη σχέση (14.81). Στο όριο αυτό, για έχει συνεπή βάθμιση η σχέση (14.80) για  $k = 2$  στο δεξί και αριστερό μέλος, παίρνουμε (πρβλ. εξ. (14.57))

$$F_2 \left( \frac{L}{\xi} \right) \sim \left( \frac{L}{\xi} \right)^{-\gamma/\nu} \quad \text{για} \quad \frac{L}{\xi} \gg 1. \quad (14.83)$$

Για να βρούμε τη βάθμιση ως προς  $L$  αντικαθιστούμε τις παραπάνω σχέσεις στην (14.80) για  $k = 2$  και παίρνουμε

$$\frac{\chi^+}{\beta L^d \xi_+^{\gamma/\nu}} \xi^{\gamma/\nu} \sim L^{-2y} \left( \frac{L}{\xi} \right)^{-\gamma/\nu} \quad (14.84)$$

από την οποία παίρνουμε

$$L^{-d} \sim L^{-2y} L^{-\gamma/\nu} \Rightarrow d = 2y + \frac{\gamma}{\nu} \Rightarrow y = \frac{d\nu - \gamma}{2\nu} = \frac{\beta}{\nu}, \quad (14.85)$$

όπου χρησιμοποιήσαμε τη γνωστή<sup>38</sup> σχέση υπερβάθμισης (hyperscaling relation)

$$d\nu = \gamma + 2\beta. \quad (14.86)$$

Καταλήγουμε στις σχέσεις που ισχύουν στην περιοχή αταξίας  $\beta < \beta_c$

$$P(L, s) = \frac{a}{C_0} L^{\beta/\nu} \tilde{P}(a L^{\beta/\nu} s, \frac{L}{\xi}) \quad (14.87)$$

$$\langle s^2 \rangle = L^{-2\beta/\nu} F_2 \left( \frac{L}{\xi} \right) \quad (14.88)$$

$$\langle s^4 \rangle = L^{-4\beta/\nu} F_4 \left( \frac{L}{\xi} \right) \quad (14.89)$$

Από τη σχέση (14.74) βρίσκουμε ότι η κρίσιμη συμπεριφορά του Binder cumulant είναι

$$U \sim 1 - \frac{L^{-4\beta/\nu} F_4 \left( \frac{L}{\xi} \right)}{3 L^{-4\beta/\nu} F_2 \left( \frac{L}{\xi} \right)^2} = 1 - \frac{1}{3} \frac{F_4 \left( \frac{L}{\xi} \right)}{F_2 \left( \frac{L}{\xi} \right)^2} \quad (14.90)$$

Στην ψευδοκρίσιμη περιοχή υπερτερούν τα φαινόμενα πεπερασμένου μεγέθους τα οποία πνίγουν τις κρίσιμες διακυμάνσεις και θέτουν στις συναρτήσεις  $F_k(x)$  πεπερασμένες τιμές. Η ψευδοκρίσιμη περιοχή στο θερμοδυναμικό όριο δίνεται από το όριο  $L \rightarrow \infty$  κρατώντας πεπερασμένο το λόγο  $L/\xi$ . Άρα, αντιστοιχεί στις τιμές των συναρτήσεων  $F_k(x)$  για μικρά  $x$ . Οπότε παίρνουμε<sup>39</sup>

$$\lim_{L \rightarrow \infty} U(t=0, L) \equiv U^* = 1 - \frac{1}{3} \frac{F_4(0)}{F_2(0)^2} = \text{const.} \quad (14.91)$$

<sup>38</sup>Δείτε λ.χ. [73], σχέσεις 3.35, 3.36, 3.53.

<sup>39</sup>Ακριβώς για  $t = 0$  έχουμε  $\xi(0) = +\infty$ , οπότε για πεπερασμένο  $L$  έχουμε ότι  $L/\xi = 0$ .

Έτσι γίνεται τώρα απλή η κατανόηση του γεγονότος ότι το  $U$  στην κρίσιμη θερμοκρασία παίρνει τιμές σχεδόν ανεξάρτητες από το μέγεθος του συστήματος  $L$ . Η τιμή  $U^*$  βρίσκεται να εξαρτάται από τις συνοριακές συνθήκες και από την ανισοτροπικότητα των αλληλεπιδράσεων. Για τετραγωνικό πλέγμα στο πρότυπο Ising έχουμε [72] (Kamieniarz+Blöte)

$$U^* = 0.610690(1) \quad (14.92)$$

### 14.13.2 Βάθμιση

Η βασική υπόθεση για τη βάθμιση των θερμοδυναμικών ποσοτήτων στην περιοχή μιας συνεχούς μετάβασης φάσης κάτω από μια αλλαγή κλίμακας μήκους στο πλέγμα όπου το αδιάστατο μήκος συσχετισμού (στο θερμοδυναμικό όριο) αλλάζει ως<sup>40</sup>

$$\xi \rightarrow \frac{\xi}{b}. \quad (14.93)$$

όπου  $b$  είναι ο παράγοντας αλλαγής της κλίμακας μήκους, είναι ότι η ελεύθερη ενέργεια του συστήματος αλλάζει ως<sup>41</sup>

$$f(t, h) = b^{-d} f(tb^{y_t}, hb^{y_h}), \quad (14.94)$$

όπου  $t$  η ανηγμένη θερμοκρασία και  $h$  το εξωτερικό μαγνητικό πεδίο. Η παραπάνω σχέση συμπυκνώνει την υπόθεση βάθμισης (scaling hypothesis) και είναι μια σχέση ανάλογη με αυτή της (14.76). Η σχέση αυτή μπορεί να γίνει κατανοητή μέσω της προσέγγισης της ομάδας επανακανονικοποίησης και η βασική υπόθεση είναι η εμφάνιση μιας μοναδικής δυναμικής κλίμακας μήκους που απειρίζεται, καθώς πλησιάζουμε το κρίσιμο σημείο και από την οποία εξαρτώνται όλες οι φυσικές ποσότητες. Τα ορίσματα  $tb^{y_t}$  και  $hb^{y_h}$  δίνουν την αλλαγή που πρέπει να γίνει στις “σταθερές σύζευξης”  $t$  και  $h$  κάτω από την αλλαγή κλίμακας μήκους, ώστε να ισχύει η ισότητα.

Εφαρμόζοντας την παραπάνω σχέση  $n$  φορές, παίρνουμε

$$f(t, h) = b^{-nd} f(tb^{ny_t}, hb^{ny_h}). \quad (14.95)$$

Αν την εφαρμόσουμε ένα μεγάλο αριθμό από φορές έτσι ώστε  $n \rightarrow \infty$ ,  $t \rightarrow 0$  κρατώντας το γινόμενο  $tb^{ny_t} = t_0 = \mathcal{O}(1)$ , τότε αντικαθιστώντας

<sup>40</sup>λ.χ. κεφ. 3 στο [73].

<sup>41</sup>Για την ακρίβεια το *ιδιάζον* (singular) κομμάτι της ελεύθερης ενέργειας.



τη σχέση που προκύπτει  $b^n \sim t^{-1/y_t}$  παίρνουμε

$$\begin{aligned} f(t, h) &= t^{d/y_t} f(t_0, ht^{-y_h/y_t}) \\ &\equiv t^{d/y_t} \Psi(ht^{-y_h/y_t}) \\ &= t^{2-\alpha} \Psi(ht^{-y_h/y_t}), \end{aligned} \quad (14.96)$$

όπου στη δεύτερη γραμμή ορίσαμε τη συνάρτηση βάθμισης  $\Psi(z)$  και στην τρίτη ορίσαμε τον κρίσιμο εκθέτη

$$\alpha = 2 - \frac{d}{y_t}. \quad (14.97)$$

Εφαρμόζοντας την παραπάνω υπόθεση στη σχέση (14.93) για το μήκος συσχετισμού

$$\xi(t, h) = b^{-1} \xi(tb^{y_t}, hb^{y_h}) = \dots = b^{-n} \xi(tb^{ny_t}, hb^{ny_h}), \quad (14.98)$$

και παίρνοντας το όριο  $n \rightarrow \infty$ ,  $t \rightarrow 0$  κρατώντας το γινόμενο  $tb^{ny_t} \sim \mathcal{O}(1)$ , το αριστερό μέλος θα έχει μια πεπερασμένη τιμή, έστω  $\xi_0 < \infty$ , ενώ το δεξί θα μας δώσει

$$\xi_0 = t^{1/y_t} \xi(t_0, ht^{-y_h/y_t}). \quad (14.99)$$

Θεωρώντας την περίπτωση  $h = 0$  και συγκρίνοντας με τη γνωστή σχέση  $\xi \sim t^{-\nu}$  (εξίσωση (14.4)) που όρισε τον κρίσιμο εκθέτη  $\nu$ , παίρνουμε

$$\xi = \xi_0 t^{-1/y_t} \Rightarrow \nu = \frac{1}{y_t}. \quad (14.100)$$

Παραγωγίζοντας τη σχέση (14.96) ως προς τη θερμοκρασία, παίρνουμε

$$\frac{\partial f}{\partial t} \sim t^{1-\alpha} \Psi(ht^{-y_h/y_t}) + t^{2-\alpha} ht^{-y_h/y_t-1} \Psi'(ht^{-y_h/y_t}). \quad (14.101)$$

Στην παραπάνω σχέση, και σε όλες που ακολουθούν που έχουν το σύμβολο  $\sim$  θα αμελούμε σταθερές που είναι άσχετες με τις ιδιότητες βάθμισης.

Παραγωγίζοντας την παραπάνω σχέση άλλη μια φορά, και θέτοντας  $h = 0$  παίρνουμε την ειδική θερμότητα

$$c \sim \frac{\partial^2 f}{\partial t^2} \sim t^{-\alpha} \Psi(0). \quad (14.102)$$

Άρα, ο κρίσιμος εκθέτης  $\alpha$  δεν είναι άλλος από τον κρίσιμο εκθέτη της ειδικής θερμότητας που ορίσαμε στην σχέση (14.4).

Την μαγνητική επιδεκτικότητα την παίρνουμε με ανάλογο τρόπο παραγωγίζοντας την (14.96) ως προς  $h$

$$\frac{\partial f}{\partial h} \sim t^{d/y_t} t^{-y_h/y_t} \Psi'(ht^{-y_h/y_t}) \sim t^{\nu d - \nu y_h} \Psi'(ht^{-\nu y_h}) \quad (14.103)$$

Παραγωγίζοντας την παραπάνω σχέση άλλη μια φορά, και θέτοντας  $h = 0$ , παίρνουμε την μαγνητική επιδεκτικότητα

$$\chi \sim \frac{\partial^2 f}{\partial h^2} \sim t^{\nu d - 2\nu y_h} \Psi'(0), \quad (14.104)$$

και συγκρίνοντας με τη σχέση (14.3)  $\chi \sim t^{-\gamma}$  παίρνουμε

$$\gamma = 2\nu y_h - \nu d \Leftrightarrow y_h = \frac{1}{2}(d + \frac{\gamma}{\nu}) = d - \frac{\beta}{\nu} = \frac{\beta + \gamma}{\nu} \quad (14.105)$$

όπου στις τελευταίες δύο σχέσεις χρησιμοποιήσαμε τις “σχέσεις υπερβάθμισης” (hyperscaling relations)

$$\nu d = \gamma + 2\beta. \quad (14.106)$$

### 14.13.3 Βάθμιση Πεπερασμένου Μεγέθους

Ας επαναλάβουμε την ανάλυση της προηγούμενης παραγράφου στην περίπτωση που το σύστημά μας έχει πεπερασμένο μέγεθος. Για τον λόγο αυτό, θα υποθέσουμε ότι το σύστημα έχει τους βαθμούς ελευθερίας του πάνω σε ένα πλέγμα με γραμμική διάσταση  $l = La$  (ο όγκος  $V = l^d$ ,  $d$  ο αριθμός των διαστάσεων), όπου το  $L$  είναι ο (αδιάστατος) αριθμός των πλεγματοτικών θέσεων και  $a$  είναι το μήκος που δίνει την πλεγματοτική σταθερά. Θα υποθέσουμε ότι παίρνουμε το όριο  $L \rightarrow \infty$  με  $a \rightarrow 0$  κρατώντας το μήκος  $l$  σταθερό. Αλλάζοντας την κλίμακα του  $L$  μειώνοντας τις πλεγματοτικές θέσεις κατά ένα παράγοντα

$$L \rightarrow \frac{L}{b} \Leftrightarrow L^{-1} \rightarrow bL^{-1}, \quad (14.107)$$

έτσι ώστε να μειωθεί

$$a \rightarrow ba \quad (14.108)$$

η σχέση (14.94) γενικεύεται στην

$$f(t, h, L^{-1}) = b^{-nd} f(tb^{ny_t}, hL^{ny_h}, b^n L^{-1}). \quad (14.109)$$

Παίρνοντας πάλι το όριο (προσέγγιση κρίσιμου σημείου)  $tb^{ny_t} = t_0 < \infty \Rightarrow b^n \sim t^{-1/y_t}$  η παραπάνω σχέση γίνεται

$$f(t, h, L^{-1}) = t^{d/y_t} f(t_0, ht^{-y_h/y_t}, t^{-1/y_t} L^{-1}) = t^{d/y_t} \Psi(ht^{-y_h/y_t}, t^{-1/y_t} L^{-1}). \quad (14.110)$$

Παραγωγίζοντας όπως και στην προηγούμενη παράγραφο και θέτοντας  $h = 0$  παίρνουμε

$$\chi(t, L^{-1}) = \left. \frac{\partial^2 f}{\partial h^2} \right|_{h=0} = t^{-\gamma} \phi_2(L^{-1} t^{-\nu}) = t^{-\gamma} \phi_2\left(\frac{\xi}{L}\right). \quad (14.111)$$

όπου θέσαμε τις τιμές  $y_t = 1/\nu$ ,  $\phi_2(x) = \Psi^{(2,0)}(0, x) = \partial^2 \Psi(z, x)/\partial z^2|_{z=0}$ .

Το θερμοδυναμικό όριο το παίρνουμε για  $L \gg \xi$  όπου  $\phi_2(\frac{\xi}{L}) \rightarrow \phi_2(0) < \infty$ , οπότε παίρνουμε τη γνωστή σχέση  $\chi \sim t^{-\gamma}$ .

Αλλά όταν το  $L$  συγκρίνεται με το  $\xi$ , επικρατούν τα φαινόμενα επίδρασης πεπερασμένου μεγέθους. Αυτά “πνίγουν” τις μεγάλες διακυμάνσεις και η μαγνητική επιδεκτικότητα παρουσιάζει ένα μέγιστο στη θερμοκρασία crossover  $t_X \equiv (\beta_c - \beta_c(L))/\beta_c$  όπου  $t_X \sim L^{-1/\nu}$ , όπου η τελευταία σχέση ισχύει γιατί  $L \sim \xi \sim t^{-\nu}$ . Παίρνουμε τότε

$$\chi_{\max} \sim t_X^{-\gamma} \phi_2(L^{-1} t_X^{-\nu}) \sim L^{\gamma/\nu} \phi_2(L^{-1} L) \sim L^{\gamma/\nu} \phi_2(1) \sim L^{\gamma/\nu}. \quad (14.112)$$

Άρα, στην περιοχή του μεγίστου θα πάρουμε μια συναρτησιακή σχέση της μορφής

$$\chi(t, L^{-1}) = L^{\gamma/\nu} F_\chi(L^{1/\nu} t), \quad (14.113)$$

που δεν είναι άλλη από την εξίσωση (14.59). Η συνάρτηση  $F_\chi(x)$  είναι αναλυτική ως προς το όρισμά της  $x = L^{1/\nu} t$ , αφού για το πεπερασμένο σύστημα η  $\chi(t, L^{-1})$  είναι αναλυτική συνάρτηση της θερμοκρασίας<sup>42</sup>. Στο θερμοδυναμικό όριο ( $L \rightarrow \infty$ ,  $t$  πεπερασμένο άρα  $x \rightarrow \infty$ )

$$F_\chi(x) \sim x^{-\gamma} \quad x \gg 1, \quad (14.114)$$

ώστε  $\chi(t, L^{-1}) = L^{\gamma/\nu} F_\chi(L^{1/\nu} t) \sim L^{\gamma/\nu} (L^{1/\nu} t)^{-\gamma} \sim t^{-\gamma}$  και κοντά στο ψευδοκρίσιμο σημείο

$$F_\chi(x) = F_{\chi,0} + F_{\chi,1}x + F_{\chi,2}x^2 + \dots \quad x \ll 1 \quad (14.115)$$

οπότε περιμένουμε να έχουμε για  $L^{1/\nu} t \ll 1$

$$\chi(t, L^{-1}) = L^{\gamma/\nu} (1 + \chi_1 L^{1/\nu} t + \chi_2 L^{2/\nu} t^2 + \dots) \quad (14.116)$$

Οι παραπάνω σχέσεις μας οδηγούν στα συμπεράσματα:

<sup>42</sup>Προκύπτει από τον ορισμό της συνάρτησης επιμερισμού η οποία είναι αναλυτική συνάρτηση της θερμοκρασίας στην ψευδοκρίσιμη περιοχή. Άρα η  $x = L^{1/\nu} t$  είναι η μεταβλητή βάθμισης και όχι μια δύναμη αυτής λ.χ. η  $\tilde{x}$  που χρησιμοποιείται στις σχέσεις (14.53) και (14.54).

- Το ψευδοκρίσιμο σημείο μετατοπίζεται σαν  $\sim L^{-1/\nu}$  [σχέση (14.50)].
- Το ύψος της κορυφής της μαγνητικής επιδεκτικότητας αυξάνει σαν  $\chi_{\max} \sim L^{\gamma/\nu}$ .
- Η κατεύθυνση μετατόπισης της κορυφής της μαγνητικής επιδεκτικότητας εξαρτάται από τις συνοριακές συνθήκες:
  - Περιοδικές συνθήκες καταπιέζουν τα αποτελέσματα των διακυμάνσεων, αφού τα wave vectors περιορίζονται από  $\frac{2\pi}{L}n$ . Αυτό αυξάνει την ψευδοκρίσιμη θερμοκρασία  $T_c(L)$  ( $\beta_c(L) < \beta_c \Rightarrow c > 0$  στην (14.50)).
  - Ελεύθερες συνοριακές συνθήκες οδηγούν σε ελεύθερες διακυμάνσεις στο σύνορο οι οποίες μειώνουν την ψευδοκρίσιμη θερμοκρασία  $T_c(L)$  ( $\beta_c(L) > \beta_c \Rightarrow c < 0$  στην (14.50)).
  - Παγωμένα σπιν στο σύνορο οδηγούν σε αυξημένη τάξη στο σύστημα. Αυτό αυξάνει την ψευδοκρίσιμη θερμοκρασία  $T_c(L)$  ( $\beta_c(L) < \beta_c \Rightarrow c > 0$  στην (14.50)).

Καταλήγουμε ότι η  $F_\chi(L^{1/\nu}t)$  εξαρτάται από τις συνοριακές συνθήκες και τη γεωμετρία του πλέγματος.

Με ανάλογο τρόπο παίρνουμε

$$\langle m^k \rangle \sim L^{-d} \frac{\partial^k f}{\partial h^k} \sim L^{-d} t^{d/y_t - ky_h/y_t} \phi_k(L^{-1}t^{-\nu}) \sim L^{-d} t^{\nu d - ky_h} \phi_k\left(\frac{\xi}{L}\right), \quad (14.117)$$

και με ανάλογα επιχειρήματα που μας οδήγησαν στην εξίσωση (14.113) παίρνουμε

$$\langle m^k \rangle \sim L^{-d} L^{-d+ky_h} F_k(L^{1/\nu}t) \sim L^{k\frac{\beta+\gamma}{\nu}} F_k(L^{1/\nu}t). \quad (14.118)$$

Για το Binder cumulant θα έχουμε από τα παραπάνω

$$U = 1 - \frac{\langle m^4 \rangle}{3\langle m^2 \rangle^2} \sim 1 - \frac{L^{4y_h} F_4(L^{1/\nu}t)}{3(L^{2y_h} F_2(L^{1/\nu}t))^2} \sim U_* + U_1(L^{1/\nu}t) + U_2(L^{1/\nu}t)^2 + \dots, \quad (14.119)$$

όπου στην τελευταία ισότητα αναπτύξαμε τις αναλυτικές συναρτήσεις  $F_{2,4}(L^{1/\nu}t)$  για μικρά  $L^{1/\nu}t$ . Βλέπουμε τότε ότι

$$\frac{\partial U}{\partial \beta} \sim \partial_t U \sim L^{1/\nu}. \quad (14.120)$$

Παραγωγίζοντας την (14.110) ως προς τη θερμοκρασία, παίρνουμε

$$\begin{aligned}
 \frac{\partial f}{\partial t} &\sim t^{d/y_t-1} \Psi(ht^{y_h/y_t}, L^{-1}t^{-1/y_t}) \\
 &\quad + t^{d/y_t} (ht^{y_h/y_t-1}) \Psi^{(1,0)}(ht^{y_h/y_t}, L^{-1}t^{-1/y_t}) \\
 &\quad + t^{d/y_t} L^{-1}t^{-1/y_t-1} \Psi^{(0,1)}(ht^{y_h/y_t}, L^{-1}t^{-1/y_t}) \\
 &\sim t^{\nu d-1} \Psi(ht^{\nu y_h}, L^{-1}t^{-\nu}) \\
 &\quad + ht^{\nu d+\nu y_h-1} \Psi^{(1,0)}(ht^{\nu y_h}, L^{-1}t^{-\nu}) \\
 &\quad + L^{-1}t^{\nu d-1-\nu} \Psi^{(0,1)}(ht^{\nu y_h}, L^{-1}t^{-\nu}), \tag{14.121}
 \end{aligned}$$

όπου υιοθετήσαμε το συμβολισμό  $\Psi^{(n,m)}(x, z) = \partial^{n+m} \Psi(x, z) / \partial x^n \partial z^m$ . Ο όρος που είναι ανάλογος του  $h$  εξαφανίζεται, όταν θέσουμε  $h = 0$ . Στην ψευδοκρίσιμη περιοχή με  $t_X \sim L^{-1/\nu}$  ο πρώτος και ο τρίτος όρος είναι ίδιας τάξης ως προς  $L$  και παίρνουμε

$$\left. \frac{\partial f}{\partial t} \right|_{h=0} = L^{-d+\frac{1}{\nu}} F^1(L^{1/\nu}t), \tag{14.122}$$

και με διαδοχικές παραγωγίσεις

$$\left. \frac{\partial^k f}{\partial t^k} \right|_{h=0} = L^{-d+\frac{k}{\nu}} F^k(L^{1/\nu}t). \tag{14.123}$$

Οι παράγωγοι

$$\left. \frac{\partial^2 f}{\partial t \partial h} \right|_{h=0} = L^{-d+y_h+\frac{1}{\nu}} F_1^1(L^{1/\nu}t) = L^{\frac{1-\beta}{\nu}} F_1^1(L^{1/\nu}t), \tag{14.124}$$

$$\left. \frac{\partial^{1+k} f}{\partial t \partial h^k} \right|_{h=0} = L^{-d+ky_h+\frac{1}{\nu}} F_k^1(L^{1/\nu}t). \tag{14.125}$$

Ειδικότερα

$$\frac{\langle E m^k \rangle}{\langle m^k \rangle} = \frac{\left. \frac{\partial^{1+k} f}{\partial t \partial h^k} \right|_{h=0}}{\left. \frac{\partial^k f}{\partial h^k} \right|_{h=0}} \sim \frac{L^{-d+ky_h+\frac{1}{\nu}}}{L^{-d+ky_h}} \sim L^{1/\nu} \tag{14.126}$$

$$\frac{\langle e^4 \rangle}{\langle e^2 \rangle^2} = \frac{L^{-d} \left. \frac{\partial^4 f}{\partial t^4} \right|_{h=0}}{\left( L^{-d} \left. \frac{\partial^2 f}{\partial t^2} \right|_{h=0} \right)^2} \sim \frac{L^{-\frac{4}{\nu}}}{(L^{-\frac{2}{\nu}})^2} \sim \text{σταθ.} \tag{14.127}$$

## 14.14 Παράρτημα: Κρίσιμοι Εκθέτες

### 14.14.1 Ορισμοί

$$\begin{aligned}
 \alpha : c &\sim t^{-\alpha}, & c_{\max} &\sim L^{\alpha/\nu}, & c(t, L) &= L^{\alpha/\nu} F^2(L^{1/\nu} t) \\
 \beta : m &\sim t^\beta, & m &\sim L^{-\beta/\nu}, & m(t, L) &= L^{-\beta/\nu} F_1(L^{1/\nu} t) \\
 \gamma : \chi &\sim t^\gamma, & \chi_{\max} &\sim L^{\gamma/\nu}, & \chi(t, L) &= L^{\gamma/\nu} F_2(L^{1/\nu} t) \\
 \nu : \xi &\sim t^{-\nu}, & \xi &\sim L, \\
 \delta : M &\sim h^{1/\delta} \\
 z : \tau &\sim \xi^z
 \end{aligned} \tag{14.128}$$

μαζί με τη σχέση βάθμισης που ορίζει τους εκθέτες  $y_t, y_h$

$$f(t, h) = t^{d/y_t} \Psi(ht^{y_h/y_t}) \tag{14.129}$$

και την σχέση που ορίζει τον εκθέτη  $\eta$  από τη συνάρτηση συσχετισμού δύο σημείων  $G(\mathbf{r}, t) = \langle \mathbf{s}(\mathbf{r}) \cdot \mathbf{s}(\mathbf{0}) \rangle$  όπου στο κρίσιμο σημείο  $t = 0$  έχουμε

$$G(\mathbf{r}, t = 0) \sim \frac{1}{r^{d-2+\eta}}. \tag{14.130}$$

### 14.14.2 Σχέσεις

Από τους ορισμούς και τις σχέσεις υπερβάθμισης (hyperscaling relations) έχουμε

$$\begin{aligned}
 \alpha + 2\beta + \gamma &= 2 \\
 \gamma + 2\beta &= \nu d \\
 2 - \nu d &= \alpha \\
 \alpha + \beta(1 + \delta) &= 2 \\
 \nu(2 - \eta) &= \gamma
 \end{aligned} \tag{14.131}$$

$$y_t = \frac{1}{\nu} = \frac{d}{2 - \alpha} \quad y_h = \frac{\beta + \gamma}{\nu} = \frac{1}{2} \left( d + \frac{\gamma}{\nu} \right) = d - \frac{\beta}{\nu} \tag{14.132}$$

$$\alpha = 2 - \frac{d}{y_t} \quad \beta = \frac{d - y_h}{y_t} \quad \gamma = \frac{2y_h - d}{y_t} \quad \delta = \frac{y_h}{d - y_h} \tag{14.133}$$

$$\eta = d + 2 - 2y_h \Leftrightarrow d - 2 + \eta = 2(d - y_h) \tag{14.134}$$

Μοντέλο	$\nu$	$\alpha$	$\beta$	$\gamma$	$\delta$	$\eta$	$y_t$	$y_h$
q=0 Potts (2d) [65]	$\infty$	$-\infty$	$\frac{1}{6}$	$\infty$	$\infty$	0	0	2
q=1 Potts (2d) [65]	$\frac{4}{3}$	$-\frac{2}{3}$	$\frac{5}{36}$	$2\frac{7}{18}$	$18\frac{1}{5}$	$\frac{5}{24}$	$\frac{3}{4}$	$\frac{91}{48}$
Ising (2d) [65]	1	0	$\frac{1}{8}$	$\frac{7}{4}$	15	$\frac{1}{4}$	1	$\frac{15}{8}$
q=3 Potts (2d) [65]	$\frac{5}{6}$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{13}{9}$	14	$\frac{4}{15}$	$\frac{6}{5}$	$\frac{28}{15}$
q=4 Potts (2d) [65, 77]	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{12}$	$\frac{7}{6}$	15	$\frac{1}{4}$	$\frac{3}{2}$	$\frac{15}{8}$
classical (4d) [76]	$\frac{1}{2}$	0	$\frac{1}{2}$	1	3	0	2	3
Spherical (3d) [76]	1	-1	$\frac{1}{2}$	2	5	0	1	$\frac{5}{2}$
Ising (3d) [76]	—	$\frac{1}{8}$	$\frac{5}{16}$	$\frac{5}{4}$	5	—	—	—
Ising (3d) [80]	0.631	0.108(5)	0.327(4)	1.237(4)	4.77(5)	0.039	—	—
Heisenberg (3d) [78]	0.70	-0.1	0.36	1.4	5	0.03	—	—
XY (3d) [79]	0.663	—	—	1.327(8)	—	—	—	—
AF q=3 Potts (3d) [81]	0.66	-0.011	0.351	1.309	4.73	—	—	—

Πίνακας 14.7: Τιμές για τους κρίσιμους εκθέτες για τα μοντέλα που αναφέρονται στην πρώτη στήλη. Όπου οι τιμές είναι με υποδιαστολή είναι προσεγγιστικές, οι υπόλοιπες ακριβώς υπολογισμένες. Στις προσεγγιστικές τιμές δεν εφαρμόζουμε σχέσεις υπερβάθμισης, αλλά αναφέρουμε απλά τις τιμές της βιβλιογραφίας. Οι τιμές για το πρότυπο 3d Ising από την [76] είναι εικασία. Για 3d Ising δεξ, επίσης, [42] p. 244. 3d XY και 3d AF q=3 Potts ειμάζεται ότι ανήκουν στην ίδια κλάση παγκοσμιότητας. Wu for q=1 Potts (2d) reports  $y_h = 51/48$  and for q=4 (2d)  $\eta = 1/2$ , not compatible with hyperscaling relations - assume typos.

## 14.15 Ασκήσεις

Στα αρχεία `all` και `allem` του συνοδευτικού λογισμικού αυτού του κεφαλαίου, υπάρχουν μετρήσεις που μπορείτε να τις χρησιμοποιήσετε για ανάλυση δεδομένων ή για να τα συγκρίνετε με τις δικές σας μετρήσεις.

1. Υπολογίστε τον μέσο λόγο αποδοχής  $\bar{A}$  στον αλγόριθμο Metropolis ως συνάρτηση της θερμοκρασίας για  $L = 10, 40, 100$ . Στις ίδιες θερμοκρασίες υπολογίστε για τον αλγόριθμο Wolff το μέσο μέγεθος των κατασκευαζόμενων clusters  $\langle n \rangle$ . Στη συνέχεια, υπολογίστε τον αριθμό των Wolff clusters που ισοδυναμούν με ένα Metropolis sweep για κάθε θερμοκρασία που προσομοιώσατε. Παραστήστε γραφικά τις παραπάνω ποσότητες.
2. Στα σχήματα 14.6–14.10 προσθέστε ανάλογα σημεία για  $L = 50, 120, 140, 160, 180, 200$ .
3. Στα σχήματα 14.11–14.12 προσθέστε ανάλογα σημεία για  $L = 50, 90, 130, 150, 190, 250$ . Επαναυπολογίστε τον κρίσιμο εκθέτη  $z$  χρησιμοποιώντας και τα δικά σας δεδομένα.
4. Στο σχήμα 14.13 προσθέστε ανάλογα σημεία για  $L = 30, 50, 70, 90$ . Επαναυπολογίστε τον κρίσιμο εκθέτη  $z$  χρησιμοποιώντας και τα δικά σας δεδομένα.
5. Να αναπαράξετε τα αποτελέσματα του πίνακα 14.1. Προσθέστε και μια 6η στήλη υπολογίζοντας  $\tau_{m,\text{Metropolis}}^A = \tau_{m,\text{Metropolis}} \bar{A}$ , όπου  $\bar{A}$  είναι ο μέσος λόγος αποδοχής του αλγόριθμου Metropolis. Αυτό αλλάζει τη μονάδα χρόνου σε  $N$  δεκτές αλλαγές των σπιν. Αυτοί είναι και οι αριθμοί που θα πρέπει να συγκριθούν άμεσα με τους  $\tau_m$ .
6. Εκτελέστε προσομοιώσεις με σκοπό να προσδιορίσετε τα μέγιστα της μαγνητικής επιδεκτικότητας και της ειδικής θερμότητας με ακρίβεια ανάλογη με αυτή που παρουσιάζεται στον πίνακα 14.5 για  $L = 10, 20, 40, 80, 100$ . Υπολογίστε τις αντίστοιχες ψευδοκρίσιμες θερμοκρασίες. Σε κάθε περίπτωση να κάνετε εκτίμηση των σφαλμάτων.
7. Εκτελέστε τις προσαρμογές που οδηγούν στα αποτελέσματα (14.38), (14.39), (14.41) και (14.43)



$L$	$\chi(\beta_c, L)$		$\langle m \rangle(\beta_c, L)$		$c(\beta_c, L)$	
40	20.50	0.02	0.6364	0.0001	0.4883	0.0007
60	41.78	0.08	0.6049	0.0002	0.5390	0.0008
80	69.15	0.09	0.5835	0.0001	0.5743	0.0012
100	102.21	0.25	0.5673	0.0002	0.6026	0.0014
120	140.18	0.11	0.5548	0.0001	0.6235	0.0010
140	183.95	0.33	0.5442	0.0002	0.6434	0.0006
160	232.93	0.55	0.5351	0.0001	0.6584	0.0020
200	342.13	0.72	0.5206	0.0001	0.6858	0.0014
500	1687.2	4.4	0.4647	0.0002	0.7794	0.0018
1000	6245	664	0.4228	0.0040	—	—

Πίνακας 14.8: Ο πίνακας των τιμών  $\chi(\beta_c, L)$ ,  $\langle m \rangle(\beta_c, L)$  και  $c(\beta_c, L)$  στην κρίσιμη θερμοκρασία για διαφορετικά  $L$  που δίνεται για την άσκηση 9.

8. Μελετήστε τη βύθμιση της ειδικής θερμότητας ως συνάρτηση της θερμοκρασίας. Συγκρίνετε την ποιότητα των προσαρμογών στις συναρτήσεις  $a \log |t|$  και  $a |t|^\alpha$  μετρώντας το  $\chi^2/\text{dof}$  όπως περιγράφεται στο Παράρτημα 13.7 μετά τη Σελ. 577.
9. Δίνεται ο πίνακας 14.8 των μετρήσεων των τιμών των  $\chi(\beta_c, L)$ ,  $\langle m \rangle(\beta_c, L)$  και  $c(\beta_c, L)$ . Κάντε τις απαραίτητες προσαρμογές, ώστε να πάρετε τους εκθέτες  $\gamma/\nu$ ,  $\beta/\nu$  και  $\alpha$  όπως περιγράφεται στο κείμενο. Ειδικά για τον  $\alpha$  να δοκιμάσετε προσαρμογή σε δύναμη και σε λογάριθμο και να συγκρίνετε τα αποτελέσματα σύμφωνα με το κείμενο.
10. Δίνεται ο πίνακας 14.5 των μετρήσεων των τιμών των  $L$ ,  $\beta_c(L)$ ,  $\chi_{\max}$ ,  $\beta'_c(L)$  και  $c_{\max}$ . Κάντε τις απαραίτητες προσαρμογές, ώστε να πάρετε τους εκθέτες  $1/\nu$ ,  $\gamma/\nu$ ,  $\alpha/\nu$  και την κρίσιμη θερμοκρασία  $\beta_c$  όπως περιγράφεται στο κείμενο. Ειδικά για τον  $\alpha$  να δοκιμάσετε προσαρμογή σε δύναμη και σε λογάριθμο και να συγκρίνετε τα αποτελέσματα σύμφωνα με το κείμενο.
11. Αναπαράγετε τις “καταρρεύσεις” των σχημάτων 14.27-14.29. Χρησιμοποιήστε τα δεδομένα από το αρχείο all από το συνοδευτικό λογισμικό, το οποίο περιγράφεται στο κείμενο. Καθορίστε αρχικά τις γνωστές παραμέτρους και υπολογίστε τις συναρτήσεις βύθμισης  $F_{\chi, m, c}$ . Μεταβάλετε κάθε παράμετρο χωριστά μέχρι η “κατάρρευση” να μην είναι ικανοποιητική και πάρτε τη μεταβολή της παραμέτρου να είναι το σφάλμα προσδιορισμού της. Προσδιορί-

στε το διάστημα της μεταβλητής  $x = L^{1/\nu}t$  όπου η βάθμιση είναι ικανοποιητική. Επαναλάβετε τον υπολογισμό σας, κάνοντας τις απαραίτητες μετρήσεις για  $L = 10, 20$ , και κρατώντας τις μετρήσεις μόνο για  $L = 10, 20, 40, 80, 120$ . Συγκρίνετε τα αποτελέσματα που θα πάρετε με τα προηγούμενα.

12. Αποδείξτε ότι για οποιαδήποτε παρατηρήσιμη ποσότητα  $\mathcal{O}$  ισχύει  $\partial\langle\mathcal{O}\rangle/\partial\beta = -\langle E\mathcal{O}\rangle + \langle\mathcal{O}\rangle\langle E\rangle = -\langle(E - \langle E\rangle)(\mathcal{O} - \langle\mathcal{O}\rangle)\rangle$ . Από τη σχέση αυτή υπολογίστε την παράγωγο του Binder Cumulant  $D_U$  και αποδείξτε τη σχέση (14.67).
13. Από το μέγιστο της παραγώγου του Binder Cumulant  $D_U$ , υπολογίστε τον κρίσιμο εκθέτη  $1/\nu$  σύμφωνα με την ανάλυση που κάναμε στα σχήματα 14.23 και 14.25 για τη μαγνητική επιδεκτικότητα.
14. Δείξτε ότι για μία γκαουσιανή κατανομή  $f(x) = ae^{-x^2/2\sigma^2}$  έχουμε  $\langle x^2\rangle = \sigma^2$  και  $\langle x^4\rangle = 3\sigma^4$ . Συμπεράνετε ότι  $1 - \langle x^2\rangle/(3\langle x^4\rangle) = 0$ .
15. Θεωρήστε την κατανομή που δίνεται από την πυκνότητα πιθανότητας

$$f(x) = a \left( e^{-\frac{(x-m)^2}{2\sigma^2}} + e^{-\frac{(x+m)^2}{2\sigma^2}} \right)$$

Κάντε τη γραφική της παράσταση και εξηγήστε γιατί μοιάζει με την κατανομή της μαγνήτισης στη φάση χαμηλής θερμοκρασίας  $\beta \gg \beta_c$ . Δείξτε ότι  $\langle x^4\rangle = m^4 + 6m^2\sigma^2 + 3\sigma^4$  και  $\langle x^2\rangle = m^2 + \sigma^2$ . Ερμηνεύστε τα αποτελέσματα, δηλ. τι εκφράζει η κάθε αναμενόμενη τιμή. Δείξτε ότι για  $\sigma \ll m$  παίρνουμε  $U \approx 2/3$ . Βεβαιωθείτε ότι η προσέγγιση που κάνατε αφορά το σύστημα που μελετάμε στη χαμηλή θερμοκρασία.

16. Υπολογίστε την παράγωγο  $\partial U/\partial\beta$  σαν συνάρτηση των  $\langle em^4\rangle$ ,  $\langle em^2\rangle$ ,  $\langle m^4\rangle$ ,  $\langle m^2\rangle$ . Εφαρμόστε τα επιχειρήματα βάθμισης πεπερασμένου μεγέθους και αποδείξτε την εξίσωση (14.120).
17. Από τις σχέσεις (14.131) και τις  $y_t = 1/\nu$ ,  $\gamma = (2y_h - d)/y_t$  να δείξετε τις υπόλοιπες ισότητες στις σχέσεις (14.132) και (14.133).

# ΒΙΒΛΙΟΓΡΑΦΙΑ

## [Συγγράμματα]

- [1] [www.physics.ntua.gr/~konstant/ComputationalPhysics/](http://www.physics.ntua.gr/~konstant/ComputationalPhysics/) Ο ιστότοπος του βιβλίου. Εκεί θα βρείτε το συνοδευτικό λογισμικό και συμπληρωματικό υλικό.
- [2] H. Gould, J. Tobochnik and H. Christian, “*Computer Simulation Methods, Application to Physical Systems*”, Third Edition, Addison Wesley (2007). Ένα εξαιρετικό εισαγωγικό βιβλίο στην υπολογιστική φυσική. Ο προγραμματισμός γίνεται σε περιβάλλον Java. Το λογισμικό δίνεται με άδεια ανοιχτού λογισμικού και μπορεί να ανακτηθεί από τη διεύθυνση [opensourcephysics.org](http://opensourcephysics.org)
- [3] R. Landau, M. J. Páez and C. C. Bordeianu, “*Computational Physics: Problem Solving with Computers*”, Wiley-VCH, 2 ed. (2007).
- [4] M. E. J. Newman and G. T. Barkema, “*Monte Carlo Methods in Statistical Physics*”, Clarendon Press, Oxford (2002). Εξαιρετικό βιβλίο για ένα εισαγωγικό, αλλά και πιο προχωρημένο, μάθημα στις μεθόδους Μόντε Κάρλο στη φυσική.
- [5] B. A. Berg, “*Markov Chain Monte Carlo Simulations and Their Statistical Analysis. With Web-Based Fortran Code*”, World Scientific, 2004. Μόντε Κάρλο για ένα μεταπτυχιακό μάθημα από έναν από τους κορυφαίους του πεδίου. Διδάσκει πολλές από τις πιο προχωρημένες μεθόδους.
- [6] D. P. Landau and K. Binder, “*A Guide to Monte Carlo Simulations in Statistical Physics*”, Cambridge University Press, 3rd Edition, 2009.
- [7] K. Binder and D. W. Heermann, “*Monte Carlo Simulation in Statistical Physics*”, Fifth Edition, Springer (2010).

- [8] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, “*Numerical Recipes, The Art of Scientific Computing*”, Third Edition, Cambridge University Press (2007), [www.nr.com](http://www.nr.com) Το απόλυτα απαραίτητο εγχειρίδιο για κάθε επιστήμονα, με συνταγές για τις πιο βασικές αριθμητικές μεθόδους.

### [Κεφάλαιο 1]

- [9] M. Metcalf, J. Reid, M. Cohen, “*Modern Fortran Explained*”, 4th Edition, OUP Oxford (2011).
- [10] J. C. Adams, W. S. Brainerd, R. A. Hendrickson, R. E. Maine, J. T. Martin and B. T. Smith, “*The Fortran 2003 Handbook: The Complete Syntax, Features and Procedures*”, Springer (2009).
- [11] T. M. R. Ellis, I. R. Philips and T. M. Lahey, “*Fortran 90 Programming*”, Addison-Wesley (1994).
- [12] C. G. Page, “*Professional Programmer’s Guide to Fortran77*”, <http://www.star.le.ac.uk/~cgp/prof77.html>
- [13] Gnuplot official site <http://gnuplot.info/>
- [14] P. K. Janert, “*Gnuplot in Action: Understanding Data with Graphs*”, Manning Publications (2009).
- [15] tcsh homepage: <http://www.tcsh.org/Home>
- [16] P. DuBois, “*Using csh & tcsh*”, O’Reilly and Associates (1995), [www.kitebird.com/csh-tcsh-book/](http://www.kitebird.com/csh-tcsh-book/)
- [17] M. J. Currie, “*C-shell Cookbook*”, <http://www.astro.soton.ac.uk/unixtut/sc4.pdf>
- [18] Wiki book: “*C Shell Scripting*”, [http://en.wikibooks.org/wiki/C\\_Shell\\_Scripting](http://en.wikibooks.org/wiki/C_Shell_Scripting)
- [19] G. Anderson and P. Anderson, “*The Unix C Shell Field Guide*”, Prentice Hall (1986).

### [Κεφάλαιο 3]

- [20] R. M. May, “*Simple Mathematical Models with Very Complicated Dynamics*”, *Nature* **261** (1976) 459.

- [21] C. Efthimiou, “*Introduction to Functional Equations: Theory and Problem-Solving Strategies for Mathematical Competitions and Beyond*”, MSRI Mathematical Circles Library (2010). Δείτε την ενότητα 16.7.
- [22] P. Cvitanović, R. Artuso, R. Mainieri, G. Tanner and G. Vattay, “*Chaos: Classical and Quantum*”, ChaosBook.org, Niels Bohr Institute (2012).
- [23] L. Smith, “*Chaos: A Very Short Introduction*”, Oxford University Press (2007).
- [24] M. Schroeder, “*Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*”, W.H. Freeman (1991).
- [25] S. H. Strogatz, “*Non Linear Dynamics and Chaos*”, Addison-Wesley (1994).
- [26] Wikipedia: “Chaos Theory”, “Logistic Map”, “Bifurcation Diagram”, “Liapunov Exponents”, “Fractal Dimension”, “Feigenbaum constants”.
- [27] Wikipedia: “List of chaotic maps”.
- [28] Wikipedia: “Newton’s method”.
- [29] M. Jakobson, “*Absolutely continuous invariant measures for one-parameter families of one-dimensional maps*”, *Commun. Math. Phys.* **81** (1981) 39.

#### [Κεφάλαιο 4]

- [30] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, “*Numerical Recipes, The Art of Scientific Computing*”, Third Edition, Cambridge University Press (2007), [www.nr.com](http://www.nr.com) Δείτε τα κεφάλαια πάνω στις μεθόδους Runge–Kutta.
- [31] E. W. Weisstein, “*Runge-Kutta Method*”, from MathWorld—A Wolfram Web Resource.  
<http://mathworld.wolfram.com/Runge-KuttaMethod.html>
- [32] J. H. E. Cartwright and O. Piro, “*The dynamics of Runge-Kutta methods*”, *Int. J. Bifurcation and Chaos* **2**, (1992) 427-449.
- [33] J. H. Mathews and K. Fink, “*Numerical Methods Using Matlab*”, Prentice Hall (2003), Chapter 9.

- [34] J. H. Mathews, “*Numerical Analysis - Numerical Methods Project*”, <http://math.fullerton.edu/mathews/numerical.html>
- [35] I. Percival and D. Richards, “*Introduction to Dynamics*”, Cambridge University Press (1982). Δείτε επίσης την [37].
- [36] J. B. McLaughlin, “*Period Doubling bifurcations and chaotic motion for a parametrically forced pendulum*”, *J. Stat. Phys.* **24** (1981) 375–388.

### [Κεφάλαιο 5]

- [37] J. V. José and E. J. Saletan, “*Classical Dynamics, a Contemporary Approach*”, Cambridge University Press, 1998.

### [Κεφάλαιο 6]

- [38] R. W. Brankin, I. Gladwell, and L. F. Shampine, “*RKSUITE: a suite of Runge-Kutta codes for the initial value problem for ODEs*”, Softreport 92-S1, Department of Mathematics, Southern Methodist University, Dallas, Texas, U.S.A (1992). Διαθέσιμο από τη διεύθυνση [www.netlib.org/ode/rksuite](http://www.netlib.org/ode/rksuite) και από το συνοδευτικό λογισμικό του βιβλίου.

### [Κεφάλαιο 9]

- [39] See the Mathematica Notebooks of Peter West <http://young.physics.ucsc.edu/115/>
- [40] U. Wolff, B. Bunk, F. Knechtli, “*Computational Physics I*”, [http://www.physik.hu-berlin.de/com/teachingandseminars/previous\\_CPI\\_CPII](http://www.physik.hu-berlin.de/com/teachingandseminars/previous_CPI_CPII)
- [41] F. T. Hioe and E. W. Montroll, “*Quantum theory of anharmonic oscillators. I. Energy levels of oscillators with positive quartic anharmonicity*”, *J. Math. Phys.* **16** (1975) 1945, <http://dx.doi.org/10.1063/1.522747>

### [Κεφάλαιο 11]

- [42] L. Kadanoff, “*Statistical Physics – Statics, Dynamics and Renormalization*”, World Scientific (2000).
- [43] J. Ambjørn, B. Durhuus and T. Jonsson, “*Quantum Geometry*”, Cambridge Monographs on Mathematical Physics, Cambridge University Press (1997).

- [44] C. Itzykson and J. M. Drouffe, “*Statistical Field Theory*”, Volume 1, Cambridge Monographs on Mathematical Physics, Cambridge University Press (1989).
- [45] D. E. Knuth, “*Seminumerical Algorithms*”, Vol. 2 of “*The Art of Computer Programming*”, Addison-Wesley (1981).
- [46] M. Lüscher, *Comput. Phys. Commun.* **79** (1994) 100; F. James, *Comput. Phys. Commun.* **79** (1994) 111; *Erratum* **97** (1996) 357. Το πρόγραμμα είναι διαθέσιμο στις διευθύνσεις  
[http://cpc.cs.qub.ac.uk/summaries/ACPR\\_v1\\_0.html](http://cpc.cs.qub.ac.uk/summaries/ACPR_v1_0.html)  
[http://wwwasd.web.cern.ch/wwwasd/cernlib/download/2001\\_wnt/src/mathlib/gen/v/ranlux.F](http://wwwasd.web.cern.ch/wwwasd/cernlib/download/2001_wnt/src/mathlib/gen/v/ranlux.F)
- [47] L. Schrage, “A More Portable Fortran Random Number Generator”, *ACM Transactions on Mathematical Software*, **5** (1979) 132-138; P. Bratley, B. L. Fox and L. Schrage, “*A Guide to Simulation*”, Springer-Verlag, 1983.
- [48] G. Marsaglia and A. Zaman, *Ann. Appl. Prob.* **1** (1991) 462.
- [49] B. Li, N. Madras and A. D. Sokal, “Critical Exponents, Hyperscaling and Universal Amplitude Ratios for Two- and Three-Dimensional Self-Avoiding Walks”, *J.Statist.Phys.* **80** (1995) 661-754 [arXiv:hep-lat/9409003]; G. Slade, “The self-avoiding walk: A brief survey”, *Surveys in Stochastic Processes*, pp. 181-199, eds. J. Blath, P. Imkeller and S. Roelly, European Mathematical Society, Zurich, (2011), [http://www.math.ubc.ca/~slade/spa\\_proceedings.pdf](http://www.math.ubc.ca/~slade/spa_proceedings.pdf)

## [Κεφάλαιο 12]

- [50] J. J. Binney, N. J. Dowrick, A. J. Fisher and M. E. J. Newman, “*The Theory of Critical Phenomena*”, Clarenton Press (1992).
- [51] R. K. Pathria and P. D. Beale, “*Statistical Mechanics*”, Third Edition, Elsevier (2011).
- [52] F. Mandl, “*Statistical Physics*”, Second Edition, Wiley (1988).
- [53] R. J. Baxter, “*Exactly Solved Models in Statistical Mechanics*”, Dover Publications (2008).

## [Κεφάλαιο 13]

- [54] E. Ising, “*Beitrag zur Theorie des Ferromagnetismus*”, *Z. Phys.* **31** (1925) 253–258.
- [55] L. Onsager, “*Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition*”, *Phys. Rev.* **65** (1944) 117–119.
- [56] K. Huang, “*Statistical Mechanics*”, John Wiley & Sons, New York, (1987). Λεπτομερής παρουσίαση της λύσης Onsager.
- [57] C. N. Yang, “*The Spontaneous Magnetization of a Two-Dimensional Ising Model*”, *Phys. Rev.* **85** (1952) 809.
- [58] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. J. Teller, “*Perspective on “Equation of state calculations by fast computing machines”*”, *Chem. Phys.* **21** (1953) 1087.
- [59] M. P. Nightingale and H. W. J. Blöte, “*Dynamic Exponent of the Two-Dimensional Ising Model and Monte Carlo Computation of the Subdominant Eigenvalue of the Stochastic Matrix*”, *Phys. Rev. Lett.* **76** (1996) 4548.
- [60] H. Müller-Krumbhaar and K. Binder, “*Dynamic properties of the Monte Carlo method in statistical mechanics*”, *J. Stat. Phys.* **8** (1973) 1.
- [61] B. Efron, “*Computers and the Theory of Statistics: Thinking the Unthinkable*”, *SIAM Review* **21** (1979) 460; “*Bootstrap Methods: Another Look at the Jackknife*”, *Ann. Statist.* **7** (1979) 1; B. Efron and R. Tibshirani, “*Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy*”, *Statistical Science* **1** (1986) 54. Ελεύθερα διαθέσιμο από το [projecteuclid.org](http://projecteuclid.org)

## [Κεφάλαιο 14]

- [62] R. H. Swendsen and J.-S. Wang, “*Nonuniversal critical dynamics in Monte Carlo simulations*”, *Phys. Rev. Lett.* **58** (1987) 86.
- [63] U. Wolff, “*Collective Monte Carlo Updating for Spin Systems*”, *Phys. Rev. Lett.* **62** (1989) 361.
- [64] A. Pelissetto and E. Vicari, “*Critical Phenomena and Renormalization-Group Theory*”, *Phys. Reports* **368** (2002) 549.
- [65] F. Y. Wu, “*The Potts Model*”, *Rev. Mod. Phys.* **54** (1982) 235.



- [66] P. D. Coddington and C. F. Baillie, “*Empirical relations between static and dynamic exponents for Ising model cluster algorithms*”, *Phys. Rev. Lett.* **68** (1992) 962.
- [67] H. Rieger, “*Critical behavior of the three-dimensional random-field Ising model: Two-exponent scaling and discontinuous transition*”, *Phys. Rev. B* **52** (1995) 6659.
- [68] M. E. J. Newman and G. T. Barkema, “*Monte Carlo study of the random-field Ising model*”, *Phys. Rev. E* **53** (1996) 393.
- [69] A. E. Ferdinand and M. E. Fisher, “*Bounded and Inhomogeneous Ising Models. I. Specific-Heat Anomaly of a Finite Lattice*”, *Phys. Rev.* **185** (1969) 832; N. Sh. Izmailian and C. -K. Hu, “*Exact amplitude ratio and finite-size corrections for the  $M \times N$  square lattice Ising model*”, *Phys. Rev. E* **65** (2002) 036103; J. Salas, “*Exact finite-size-scaling corrections to the critical two-dimensional Ising model on a torus: II. Triangular and hexagonal lattices*”, *J. Phys. A* **34** (2001) 1311; W. Janke and R. Kenna, “*Exact finite-size scaling with corrections in the two-dimensional Ising model with special boundary conditions*”, *Nucl. Phys. (Proc. Suppl.)* **106** (2002) 929.
- [70] J. Ambjørn and K. N. Anagnostopoulos, “*Quantum geometry of 2D gravity coupled to unitary matter*”, *Nucl. Phys. B* **497** (1997) 445.
- [71] K. Binder, “*Critical Properties from Monte Carlo Coarse Graining and Renormalization*”, *Phys. Rev. Lett.* **47** (1981) 693.
- [72] K. Binder, “*Finite size scaling analysis of ising model block distribution functions*”, *Z. Phys. B* **43** (1981) 119; G. Kamieniarz and H. W. J. Blöte, “*Universal ratio of magnetization moments in two-dimensional Ising models*”, *J. Phys. A* **26** (1993) 201.
- [73] J. Cardy, “*Scaling and Renormalization in Statistical Physics*”, 1st Edition, Cambridge University Press (1996).
- [74] A. M. Ferrenberg and D. P. Landau, “*Critical behavior of the three-dimensional Ising model: A high-resolution Monte Carlo study*, *Phys. Rev. B* **44** (1991) 5081.
- [75] M. S. S. Challa, D. P. Landau and K. Binder, “*Finite-size effects at temperature-driven first-order transitions*”, *Phys. Rev. B* **34** (1986) 1841.

- [76] H. E. Stanley, “*Introduction to Phase Transitions and Critical Phenomena*”, Oxford (1971).
- [77] R. Creswick and S.-Y. Kim, “*Critical exponents of the four-state Potts model*, *J. Phys. A: Math.Gen.* **30** (1997) 8785.
- [78] C. Holm and W. Janke, “*Critical exponents of the classical three-dimensional Heisenberg model: A single-cluster Monte Carlo study*”, *Phys. Rev. B* **48** (1993) 936 [arXiv:hep-lat/9301002].
- [79] M. Hasenbusch and S. Meyer, “*Critical exponents of the 3D XY model from cluster update Monte Carlo*”, *Phys. Lett. B* **241** (1990) 238.
- [80] M. Kolesik and M. Suzuki, “*Accurate estimates of 3D Ising critical exponents using the coherent-anomaly method*”, *Physica A* **215** (1995) 138.
- [81] M. Kolesik and M. Suzuki, “*Critical exponents of the 3D antiferromagnetic three-state Potts model using the coherent-anomaly method*”, *Physica A* **216** (1995) 469.