ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μελέτη των Self – avoiding Random Walks

Ρούσσος Ιωσήφ
Α.Μ: 09102046

Επιβλέπων Καθηγητής
Κωνσταντίνος Ν. Αναγνωστόπουλος

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Ο αυτοαποφεύγων τυχαίος περιπατητής είναι ένα μοντέλο που εφευρέθηκε περίπου μισό αιώνα πριν. Αρχικά κανείς δεν είχε φανταστεί την εκπληκτική του χρησιμότητα στη προσομοίωση πολυμερών και πολλοί το θεωρούσαν ως ένα χοντροκομμένο μοντέλο που έμοιαζε με καρικατούρα. Πλέον ελάχιστοι είναι αυτοί που δεν πιστεύουν πως αποτελεί ένα εξαιρετικό, αν όχι τέλειο, μοντέλο για να περιγράψουμε πάρα πολλές από τις ιδιότητες των πολυμερών σε καλό διάλυμα. Τα τελευταία χρόνια γίνεται ευρεία μελέτη για την κατανόησή του και τον ακριβή υπολογισμό των κρίσιμων μεγεθών του, κάποια από τα οποία παρουσιάζουν παγκοσμιότητα. Σκοπός αυτής της εργασίας είναι η μελέτη του και ο αριθμητικός υπολογισμός πολλών μεγεθών του στις δύο διαστάσεις μέσα από προσομοιώσεις με τη μέθοδο Μόντε Κάρλο, καθώς και η σύγκριση της απλής δειγματοληψίας με τη δειγματοληψία με κριτήριο σημαντικότητας που εφαρμόζουμε χρησιμοποιώντας τον αλγόριθμο pivot.

Στη μέθοδο της απλής δειγματοληψίας υπολογίζουμε τον κρίσιμο εκθέτη ν για διάφορες θερμοκρασίες, τη σταθερά φθοράς λ του αλγορίθμου και τη συνδετική σταθερά του πλέγματος μ. Ακόμη εξετάζουμε αποδοτικότητά της καθώς και τους περιορισμούς που επιβάλει στα μήκη των αλυσίδων που μπορεί να περιγράψει.

Στη μέθοδο της δειγματοληψίας με κριτήριο σημαντικότητας χρησιμοποιούμε τον αλγόριθμο που pivot που είναι το state of the art έτσι όπως έχει υλοποιηθεί από τον Tom Kennedy. Εδώ οι προσομοιώσεις και οι υπολογισμοί μας γίνονται πάνω σε τρία διαφορετικά πλέγματα: το τετράγωνο, το τριγωνικό και το εξαγωνικό. Και πάλι υπολογίζουμε τον κρίσιμο και παγκόσμιο εκθέτη ν, τον εκθέτη p και τους χρόνους αυτοσυσχετισμού των δειγμάτων που παράγουμε.

Σ' αυτό το σημείο θα ήθελα να ευχαριστήσω τον καθηγητή Κωνσταντίνο Ν. Αναγνωστόπουλο του οποίου η συμβολή για την κατανόηση και ανάπτυξη του θέματος τόσο ως προς το τεχνικό όσο και ως προς το ουσιαστικό μέρος ήταν ιδιαίτερα χρήσιμη όποτε αυτό χρειάστηκε.

Ρούσσος Ιωσήφ

Ζωγράφου 26 – 10 – 2009

1. ΕΙΣΑΓΩΓΗ

1.1 Ο Αυτοαποφεύγων τυχαίος περιπατητής (Self-avoifing random walker – SAW)

Ο αυτοαποφεύγων τυχαίος περιπατητής (self-avoiding random walker – SAW) είναι ένα μοντέλο που προτάθηκε για πρώτη φορά περίπου μισό αιώνα πριν, και το οποίο θα περιέγραφε γραμμικά πολυμερή σε καλό διάλυμα. Σε μια πρώτη προσέγγιση είχε φανεί σαν ένα χοντροκομμένο μοντέλο που έμοιαζε με καρικατούρα! Τα πραγματικά πολυμερή υπάρχουν στο συνεχή χώρο, έχουν τετραεδρικούς δεσμούς και σχετικά πολύπλοκη αλληλεπίδραση μεταξύ των μονομερών τους. Από την άλλη μεριά ο αυτοαποφεύγων τυχαίος περιπατητής "υπάρχει" πάνω σε διακριτό πλέγμα χωρίς τετραεδρικές γωνίες, έχει ενέργεια που είναι ανεξάρτητη από τη στροφή των δεσμών και σταθερό απωστικό δυναμικό μεταξύ των μονομερών.

Παρά όλες αυτές τις κάπως υπερβολικές απλουστεύσεις, ελάχιστες είναι πλέον οι αμφιβολίες για το ότι είναι όχι απλά ένα πολύ καλό, αλλά σχεδόν τέλειο μοντέλο για κάποιες (πάντως όχι όλες!) από τις ιδιότητες ενός γραμμικού πολυμερούς σε καλό διάλυμα. Αυτό το φαινομενικό "θαύμα" προκύπτει από τη παγκοσμιότητα (universality) που παίζει κεντρικό ρόλο στη σύγχρονη θεωρία των κρίσιμων φαινομένων. Εν συντομία, βασικά συστήματα στατιστικής μηχανικής μπορούν να κατηγοριοποιηθούν σε διάφορες κλάσεις παγκοσμιότητας (univarsality classes) με βάση κάποιες γενικές ιδιότητες όπως τις χωρικές διαστάσεις, τις συμμετρίες και άλλα. Στην περιοχή γύρω από το κρίσιμο σημείο, και μόνο εκεί, η ασυμπτωτική συμπεριφορά που συστήματος είναι ΑΚΡΙΒΩΣ Η ΙΔΙΑ για όλα τα συστήματα της ίδιας κλάσης παγκοσμιότητας! Επομένως το καίριο ζήτημα πλέον είναι να προσδιορίσουμε ποιες ποσότητες είναι παγκόσμιες και ποιες όχι στο φυσικό σύστημα που μελετάμε.

Για τον υπολογισμό των μη παγκόσμιων ποσοτήτων μπορούμε να χρησιμοποιήσουμε τις κλασικές μεθόδους της θεωρητικής φυσικής: Θεωρούμε ένα μοντέλο κάπως ρεαλιστικό ή και ημιεμπειρικό, το οποίο θα υφίσταται σταδιακές προσεγγίσεις. Επομένως τα αποτελέσματά του θα περιμένουμε να είναι μία απλή προσέγγιση και μόνο της πραγματικότητας. Το ίδιο βεβαίως θα περιμένουμε ακόμη

και αν το μαθηματικό μοντέλο λύνεται αναλυτικά, αφού το ίδιο από μόνο του είναι μια προσέγγιση της πραγματικότητας!

Από την άλλη μεριά, για τον υπολογισμό των παγκόσμιων ποσοτήτων ακολουθούμε μία αρκετά διαφορετική διαδικασία. Μπορούμε να επιλέξουμε ένα μαθηματικό μοντέλο (όσο πιο απλό τόσο το καλύτερο) που ανήκει στη ίδια κλάση με το υπό μελέτη σύστημα. Λύνοντάς το, θα έχουμε υπολογίσει ακριβώς τις τιμές των παγκόσμιων ποσοτήτων! Φυσικά, κάποια μαθηματικά μοντέλα μπορεί να μην λύνονται αναλυτικά, επομένως επιπλέον προσεγγίσεις ή αριθμητικές προσομοιώσεις ίσως είναι απαραίτητες αλλά οι τελευταίες θα είναι η μόνη πηγή σφαλμάτων στα αποτελέσματά μας. Σε αυτή τη περίπτωση είναι αναγκαίο να επιβεβαιώσουμε ότι το παραλλαγμένο μοντέλο, με τις επιπλέον προσεγγίσεις, ανήκει επίσης στην ίδια κλάση παγκοσμιότητας. Τότε μπορούμε να γνωρίζουμε με σιγουριά ότι οι προσεγγίσεις δεν θα έχουν καμία επίδραση στις παγκόσμιες ποσότητες. Επικεντρώνοντας λοιπόν το ενδιαφέρον μας στον αυτοαποφεύγοντα τυχαίο περιπατητή (SAW) διαπιστώνουμε ότι η συμπεριφορά ενός τέτοιου περίπατου στο όριο που το μήκος του τείνει στο άπειρο περιγράφεται με κάποια παγκόσμια μεγέθη.

Έχει βρεθεί εμπειρικά ότι το τετράγωνο της μέσης απόστασης από άκρη σε άκρη ενός περίπατου που αποτελείται από N βήματα έχει την παρακάτω ασυμπτωτική συμπεριφορά:

$$< R^2 > \; = \; AN^{2\nu}[1+O(N^{-\Delta})] \qquad N \to \infty \quad (1.1.1)$$

όπου ο κρίσιμος εκθέτης ν είναι παγκόσμιος για όλα τα πολυμερή, διαλύματα και θερμοκρασίες και η τιμή του εκτιμάται στα *3/4*. Αντίθετα, ο κρίσιμος συντελεστής πλάτους A δεν είναι παγκόσμιος αλλά εξαρτάται από το πολυμερές, το διάλυμα και τη θερμοκρασία και η εξάρτησή του από τα προηγούμενα δεν αναμένεται να είναι μια απλή υπόθεση.

Ακόμη, γνωρίζουμε πλέον ότι στην προσομοίωση τέτοιων περιπάτων από τον αλγόριθμο pivot (τον οποίο θα δούμε στη συνέχεια), το ποσοστό των pivots που γίνονται αποδεκτά (acceptance fraction) τείνει στο μηδέν με το μήκος σύμφωνα με:

$$< f > \; = \; AN^{-p}[1+O(N^{-\Delta})] \qquad N \to \infty \quad (1.1.2)$$

όπου ο κρίσιμος εκθέτης p εκτιμάται περίπου *0.19* στις δύο διαστάσεις.

Τέλος, για το χρόνο αυτοσυσχετισμού των μετρήσεων του αλγόριθμου pivot πιστεύουμε ότι:

$$\tau \sim N^p \qquad N \to \infty \quad (1.1.3)$$

1.2 Στοιχεία Στατιστικής Φυσικής

Η στατιστική φυσική έχει σαν σκοπό να περιγράψει συστήματα με πολύ μεγάλο αριθμό βαθμών ελευθερίας N. Απλά συστήματα έχουν τυπικά N περίπου 10^{23} έως 10^{44}. Για τέτοια συστήματα οι εξισώσεις που περιγράφουν μικροσκοπικά το σύστημα είναι αδύνατο να λυθούν και τελικά μάλλον άχρηστο. Αρκούν μόνο μερικές σωστά ορισμένες "χονδροειδείς ιδιότητες" (bulk properties) που θα μας δώσουν τις χρήσιμες πληροφορίες. Λ.χ. σε ένα μαγνήτη πολλές φορές μας αρκεί να γνωρίζουμε την ενέργεια και τη μαγνήτιση του υλικού, σε ένα ρευστό την ενέργεια και τη πυκνότητά του κ.ο.κ. Η αναλυτική γνώση της θέσης, ορμής, ενέργειας κλπ. του κάθε σωματιδίου του συστήματος εκτός του ότι δεν είναι εφικτή αποτελεί και όγκο πληροφορίας που δεν είναι διαχειρίσιμος. Αλλά ακόμη και να ήταν, τα αποτελέσματα που θα προέκυπταν από μία τέτοια διαδικασία δε θα διέφεραν τελικά από τα αποτελέσματα που θα προέκυπταν από τη στατιστική μελέτη του συστήματος!

Θα κάνουμε τις, όχι ιδιαίτερα περιοριστικές, υποθέσεις ότι το σύστημά μας αποτελείται από διακριτές καταστάσεις που απαριθμούνται σε ένα σύνολο {μ} με αντίστοιχες ενέργειες $E_0<E_1<...<E_\mu<...$.Το σύστημα βρίσκεται σε επαφή με δεξαμενή θερμότητας που έχει θερμοκρασία $\beta = 1/kT$ με το οποίο μπορεί να αλληλεπιδρά. Η επαφή με τη δεξαμενή έχει σαν αποτέλεσμα να υπάρχουν *τυχαίες* μεταβάσεις του συστήματος με τρόπο που προσδιορίζεται από τη δυναμική του. Οι θεμελιώδεις ποσότητες που μας ενδιαφέρουν είναι τα βάρη (weights) $\omega_\mu(t)$ που δίνουν τη πιθανότητα το σύστημα να βρίσκεται στη κατάσταση μ τη χρονική στιγμή t. Με αυτό το τρόπο γίνεται η κωδικοποίηση της μικροσκοπικής φυσικής στη στατιστική φυσική.

Έστω ότι $R(\mu \to \nu)$ είναι ο ρυθμός μετάβασης από την κατάσταση $\mu \to \nu$

Άρα, $R(\mu \to \nu)dt$ είναι η πιθανότητα μετάβασης $\mu \to \nu$ σε χρόνο dt

Τότε μπορούμε να γράψουμε την πολύ γενική "δεσπόζουσα" εξίσωση:

$$\frac{d\omega_\mu(t)}{dt} = \sum_\nu \{\omega_\nu(t) R(\nu \to \mu) - \omega_\mu(t) R(\mu \to \nu)\} \quad (1.2.1)$$

$$\sum_\mu \omega_\mu(t) = 1 \quad (1.2.2)$$

Η (1.2.1) αναπαριστά τη μεταβολή του βάρους $\omega_\mu(t)$ που ισούται με το ρυθμό που εισέρχεται το σύστημα στην κατάσταση μ από οποιαδήποτε άλλη κατάσταση ν, μείον το ρυθμό που εξέρχεται από την κατάσταση μ.
Η (1.2.2) εκφράζει ότι τα βάρη είναι πιθανότητες και άρα η πιθανότητα το σύστημα να βρίσκεται σε κάποια κατάσταση είναι 1.

Οι ρυθμοί μετάβασης $R(\mu \to \nu)$ προκύπτουν από τη θερμική φύση του συστήματος με τη θερμική δεξαμενή και στη πράξη προσομοιώνονται με κατάλληλες επιλογές κατά τη διάρκεια υπολογισμών Μόντε Κάρλο. Οι $R(\mu \to \nu)$ θεωρούνται ανεξάρτητοι του χρόνου οπότε το παραπάνω σύστημα εξισώσεων για τα τα $\omega_\mu(t)$ είναι γραμμικό, και ο περιορισμός $0 \le \omega_\mu(t) \le 1$ οδηγεί στο μη τετριμμένο συμπέρασμα ότι σε άπειρο χρόνο τα $\omega_\mu(t)$ θα συγκλίνουν σε αριθμούς p_μ, τις πιθανότητες κατάληψης ισορροπίας. Δηλαδή μετά από κάποιο χρόνο

$$\frac{d\omega_\mu(t)}{dt} = 0 \quad (1.2.3)$$

$$p_\mu = \lim_{t \to \infty} \omega_\mu(t) \qquad \text{και} \qquad \sum_\mu p_\mu = 1 \quad \text{(1.2.4) και (1.2.5)}$$

Οι πιθανότητες p_μ για σύστημα σε ισορροπία με δεξαμενή θερμοκρασίας $\beta = 1/kT$ μπορεί να δειχθεί (Gibbs 1902) ότι ακολουθούν τη κατανομή Boltzmann.

$$p_\mu = \frac{1}{Z} e^{-\beta E_\mu} \quad (1.1.6)$$

Η παράμετρος β συχνά αναφέρεται απλά ως η θερμοκρασία του συστήματος και βλέπουμε ότι μέσω του εκθετικού στην εξίσωση (1.1.6) καθορίζει μία χαρακτηριστική ενέργεια στο σύστημα.
Η σταθερά Z στην εξίσωση (1.1.6) είναι η συνάρτηση επιμερισμού του συστήματος και η σταθερά κανονικοποίησης της κατανομής p_μ. Η σχέση (1.2.5) μας δίνει:

$$Z(\beta)=\sum_{\mu} e^{-\beta E_\mu} \quad (1.2.7)$$

Η τιμή μιας φυσικής ποσότητας που μετριέται στο εργαστήριο έχει στοχαστικό χαρακτήρα. Για συστήματα με πολύ μεγάλο αριθμό βαθμών ελευθερίας N πρακτικά κανείς ενδιαφέρεται για τη μέση τιμή μιας ποσότητας μιας και η πιθανότητα να μετρήσει μια τιμή που να διαφέρει σημαντικά από αυτή είναι αμελητέα.

Σύμφωνα με τα παραπάνω η μέση τιμή $<Q>$ μιας φυσικής ποσότητας Q η οποία παίρνει την τιμή Q_μ στη κατάσταση μ είναι:

$$<Q>=\sum_{\mu} p_\mu Q_\mu=\frac{1}{Z}\sum_{\mu} Q_\mu e^{-\beta E_\mu} \quad (1.2.8)$$

Όπως αποδεικνύεται, η τυπική απόκλιση ΔQ για ένα τυπικό θερμοδυναμικό σύστημα είναι τέτοια ώστε:

$$\frac{\Delta Q}{Q} \sim \frac{1}{\sqrt{N}} \quad (1.2.9)$$

ποσοστό που είναι αμελητέο για συνήθη θερμοδυναμικά συστήματα (πχ. για $N \sim 10^{23}$ έχουμε $\Delta Q/Q \sim 10^{-11}$). Για το λόγο αυτό όταν το σύστημα είναι μεγάλο οι διακυμάνσεις μπορούν να αγνοηθούν. Το όριο $N \to \infty$ ονομάζεται **θερμοδυναμικό όριο** και το ενδιαφέρον μας συνήθως εστιάζεται στη συμπεριφορά του συστήματος στο όριο αυτό. Ενώ τα συστήματα στο εργαστήριο είναι τις περισσότερες φορές κοντά στο όριο αυτό, στις προσομοιώσεις μας δεν είναι πάντα εφικτό να μελετήσουμε τόσο 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.

2. Η ΜΕΘΟΔΟΣ MONTE CARLO

Οι μέθοδοι Monte Carlo είναι μία κατηγορία υπολογιστικών αλγορίθμων που βασίζονται σε επανειλημμένες τυχαίες δειγματοληψίες της συνάρτησης επιμερισμού του υπό μελέτη συστήματος. Αυτές οι μέθοδοι χρησιμοποιούνται όταν είναι πρακτικά αδύνατο να υπολογίσουμε ένα ακριβές αποτέλεσμα μέσω ενός ντετερμινιστικού αλγορίθμου. Το μεγάλο τους πλεονέκτημα είναι ότι ακριβή συμπεράσματα μπορούν να προκύψουν από τη μελέτη δειγμάτων πολλές τάξεις μεγέθους μικρότερες σε πλήθος από αυτές του συνολικού χώρου καταστάσεων. Σε αυτό το κεφάλαιο θα μιλήσουμε για τους τρόπους που μπορεί να γίνει μία δειγματοληψία, τα χαρακτηριστικά που πρέπει να έχει ένας αλγόριθμος καθώς και τα κριτήρια που πρέπει να ικανοποιεί ώστε να χρησιμοποιηθεί για μία προσομοίωση με τη μέθοδο Μόντε Κάρλο. Τέλος, θα αναφερθούμε στη στατιστική και στους χρόνους αυτοσυσχετισμού των δειγμάτων που παράγονται από τέτοιους αλγορίθμους.

2.1 Δειγματοληψία

Όταν κάνουμε δειγματοληψία ο στόχος μας είναι να προσδιορίσουμε τη μέση τιμή ενός στατιστικού μεγέθους Q στη κανονική συλλογή:

$$<Q> = \sum_{\mu} p_{\mu} Q_{\mu} = \frac{\sum_{\mu} Q_{\mu} e^{-\beta E_{\mu}}}{\sum_{\mu} e^{-\beta E_{\mu}}} \qquad (2.1.1)$$

Για να το πετύχουμε αυτό επιλέγουμε ένα δείγμα από M καταστάσεις $\{\mu_1,\mu_2,...,\mu_M\}$ οι οποίες κατανέμονται σύμφωνα με τη κατανομή πιθανότητας P_μ. Ορίζουμε τον εκτιμητή (estimator) Q_M της Q:

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} P_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{M} P_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}} \qquad (2.1.2)$$

Η παραπάνω σχέση γίνεται εύκολα κατανοητή αν σκεφτούμε πως για μεγάλο δείγμα P_{μ_i} περιμένουμε ότι:

$$<Q> = \lim_{M \to \infty} Q_M \qquad (2.1.3)$$

Ο στόχος μας λοιπόν είναι μία καλή επιλογή της P_μ έτσι ώστε η παραπάνω σύγκλιση να γίνεται γρήγορα.

Μπορούμε τώρα πλέον, επικεντρώνοντας τη συζήτηση στον αυτοαποφεύγοντα τυχαίο περιπατητή, να διακρίνουμε τις εξής περιπτώσεις:

2.1.1 Απλή δειγματοληψία στα SAW (Simple Sampling)

Εδώ η επιλογή που κάνουμε είναι $P_\mu=1$. Άρα

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{M} e^{-\beta E_{\mu_i}}} \qquad (2.1.1.1)$$

Η πιο απλή τεχνική για να παράγουμε ένα SAW μήκους N, είναι να παράγουμε τυχαίες διαδρομές ενός απλού τυχαίου περιπατητή και να τις δεχόμαστε μόνο εάν είναι αυτοαποφεύγουσες (self-avoiding). Αυτή τη διαδικασία θα την επαναλαμβάνουμε μέχρι να επιτύχουμε... Είναι εύκολο να καταλάβουμε ότι κάθε τέτοια διαδρομή μήκους N παράγεται από τον αλγόριθμο με ίδια πιθανότητα. Φυσικά για να εξοικονομήσουμε χρόνο θα πρέπει να κάνουμε έλεγχο για κάθε βήμα του περιπατητή ώστε να επιβεβαιώνουμε ότι αυτός παραμένει αυτοαποφεύγων.

Ο αλγόριθμος σε ψευδοκώδικα θα είναι ο εξής:

title *Simple Sampling*

function *ssamp(N)*

comment *Αυτή η ρουτίνα επιστρέφει μια τυχαία διαδρομή SAW N-βημάτων*

$\omega_0 \leftarrow 0$ *(ω_i είναι το i-οστό βήμα ενός περίπατου ω, ω_0 είναι το αρχικό σημείο του)*

start: ***for*** *i=1* ***to*** *N* ***do***

$\omega_i \leftarrow$ *ένας τυχαίος πλησιέστερος γείτονας της* ω_{i-1}

if $\omega_i \in \{\omega_0,...,\omega_{i-1}\}$ ***goto*** *start*

enddo

return ω

Το πρόβλημα σε αυτό τον αλγόριθμο είναι, προφανώς, ο εκθετικά αυξανόμενος αριθμός δειγμάτων που πρέπει να παραχθούν ώστε να καταφέρουμε SAW όσο αυξάνεται το μήκος N. Είναι εύκολο να σκεφτούμε ότι η πιθανότητα για να είναι μια διαδρομή μήκους N αυτοαποφεύγουσα είναι $c_N/(2d)^N$. Εδώ c_N είναι το σύνολο όλων των πιθανών περιπάτων μήκους N, ενώ $(2d)^N$ είναι το σύνολο όλων των απλών περιπάτων μήκους N. Για μεγάλα N περιμένουμε ότι:

$$\frac{c_N}{(2d)^N} \sim \left(\frac{\mu}{2d}\right)^N N^{\gamma-1} \quad (2.1.1.1)$$

$$\sim e^{-\lambda N} N^{\gamma-1} \quad (2.1.1.2)$$

όπου (2.1.1.3)

$$\lambda = \ln\frac{2d}{\mu}$$

είναι η σταθερά φθοράς (attrition constant), μ είναι η συνδετική σταθερά του πλέγματος (connective constant) που εκφράζει το μέσο όρο των διαθέσιμων βημάτων σε ένα απείρου μήκους περίπατο, και γ ένας κρίσιμος εκθέτης που είναι παγκόσμιος . Επομένως ο μέσος όρος των προσπαθειών που απαιτούνται για να δημιουργήσουμε ένα SAW μήκους N είναι $(2d)^N/c_N$ και μεγαλώνει με ρυθμό $e^{\lambda N}$. Βλέπουμε ότι αυτή η μέθοδος είναι εξαιρετικά αναποτελεσματική για την αναπαραγωγή περιπάτων για

μήκη N > 10/λ και τελικά είναι αδύνατο να φτιάξουμε SAWs με μήκος μεγαλύτερο των 20-60 βημάτων περίπου.

Κάποια βελτίωση μπορούμε να παρατηρήσουμε αν χρησιμοποιήσουμε τον αλγόριθμο του μη επιστρέφοντα τυχαίου περιπατητή (Non-reversal random walker – NRRW) καθώς το νέο λ (λ') είναι:

$$\lambda' = \ln \frac{2d-1}{\mu} \qquad (2.1.1.4)$$

και μπορούμε πλέον να παράγουμε διαδρομές με το πολύ 300 βήματα. Το μήκος όμως παραμένει μικρό και το υπολογιστικό κόστος εξαιρετικά μεγάλο...

2.1.2 Δειγματοληψία με κριτήριο σημαντικότητας στα SAW (Importance Sampling)

Γενικώς όταν δειγματοληπτούμε, οι καταστάσεις που δίνουν σημαντική συνεισφορά στον υπολογισμό του $<Q>$ είναι ένα πολύ μικρό μόνο μέρος του συνολικού χώρου των καταστάσεων. Αν επιλέγουμε το δείγμα με πιθανότητα

$$P_\mu = p_\mu = \frac{e^{-\beta E_\mu}}{Z} \qquad (2.1.2.1)$$

περιμένουμε να δειγματοληπτήσουμε ακριβώς μέσα στον υπόχωρο αυτό. Τότε ο υπολογισμός του εκτιμητή Q_M δίνεται από:

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} (e^{-\beta E_{\mu_i}})^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{M} (e^{-\beta E_{\mu_i}})^{-1} e^{-\beta E_{\mu_i}}} = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu_i} \qquad (2.1.2.2)$$

Η παραπάνω δειγματοληψία λέγεται δειγματοληψία με κριτήριο σημαντικότητας (importance sampling) και είναι η μέθοδος που χρησιμοποιείται για προσομοιώσεις

στατιστικών συστημάτων στη κανονική συλλογή.

Εμείς για τη προσομοίωση του SAW χρησιμοποιήσαμε τον αλγόριθμο **pivot** τον οποίο εξηγούμε αναλυτικά σε παρακάτω κεφάλαιο. Εν συντομία αναφέρουμε τα βήματα του αλγορίθμου:

1. Διάλεξε έναν οποιοδήποτε περίπατο $\omega_0 \in W$ όπου W είναι το σύνολο των περιπάτων μήκους N.

2. Διάλεξε τυχαία έναν ακέραιο i από το σύνολο {0, 1, 2, ... , N-1}. Όρισε $x = \omega_t(i)$ ως τη τοποθεσία που θα γίνει το pivot. Διάλεξε τυχαία μία συμμετρία πλέγματος g από το σύνολο G του πλέγματος. Θέσε $\bar{\omega}(k) = \omega_t(k)$ για $k \leq i$ και $\bar{\omega}(k) = g(\omega_t(k))$ για $k > i$.

3. Εάν $\bar{\omega}$ είναι self-avoiding θέσε $\omega_{t+1} = \bar{\omega}(k)$ αλλιώς $\omega_{t+1} = \omega_t$

4. Αύξησε το t κατά 1 και πήγαινε στο βήμα 2

2.2 Διαδικασίες Markov

Το "κόλπο" στις προσομοιώσεις Monte Carlo είναι η δημιουργία μιας κατάλληλης ακολουθίας καταστάσεων του συστήματος που μελετάμε σύμφωνα με τη κατανομή Boltzmann. Αρχικά, δεν είναι σοφό να διαλέγουμε καταστάσεις στην τύχη και να τις δεχόμαστε ή να τις απορρίπτουμε με πιθανότητα ανάλογη του $e^{-\beta E\mu}$. Αυτό θα είναι τόσο χρονοβόρο και αναποτελεσματικό όπως η απλή δειγματοληψία που διαλέγουμε δείγματα στην τύχη. Δηλαδή, στο τέλος θα καταλήξουμε να έχουμε απορρίψει σχεδόν όλες τις πιθανές καταστάσεις καθώς οι πιθανότητες αποδοχής τους είναι εκθετικά μικρές. Γι' αυτό το λόγο δημιουργούμε μία διαδικασία όπου δεδομένης μιας κατάστασης του συστήματος μ, μας παράγει με στοχαστικό τρόπο μία νέα κατάσταση ν. Αυτή η διαδικασία λέγεται διαδικασία Markov και δημιουργεί μία αλυσίδα καταστάσεων:

$$\mu_0 \to \mu_1 \to \mu_2 \to \mu_3 \to \ldots \to \mu_N \quad (2.2.1)$$

Η παραπάνω αποκαλείται και αλυσίδα Markov και θα αποτελέσει το δείγμα $\{\mu_i\}\equiv\{\mu_1,\mu_2,\mu_3,...,\mu_N\}$. Δεχόμαστε ότι η επιλογή κατάστασης μ_i γίνεται στο "χρόνο" i. Η πιθανότητα μετάβασης $P(\mu\to\nu)$ (transition probability) στην κατάσταση ν όταν το σύστημα βρίσκεται στη κατάσταση μ πρέπει να ικανοποιεί τις παρακάτω συνθήκες:

1. *Να είναι ανεξάρτητη του "χρόνου"*

2. *Να εξαρτάται μόνο από τις καταστάσεις μ και ν και όχι από τη διαδρομή που ακολουθήθηκε για να φτάσουμε μέχρι τη μ*

3. *Να ικανοποιείται η σχέση:* *(2.2.2)*

$$\sum_{\nu} P(\mu\to\nu)=1$$

Προσοχή, συνήθως $P(\mu\to\mu) > 0$ καθώς το ενδεχόμενο το σύστημα να παραμείνει στην ίδια κατάσταση είναι υπαρκτό!

4. *Για $t\to\infty$ το δείγμα μ_i ακολουθεί τη κατανομή P_μ (Boltzmann)*

Η προσομοίωση Monte Carlo γίνεται με αυτό το τρόπο. Επιλέγουμε κατάλληλα μία αρχική κατάσταση μ_0 για το σύστημα και εφαρμόζοντας τον παραπάνω αλγόριθμο. Η μεγαλύτερη προσπάθεια επικεντρώνεται στο προσδιορισμό των πιθανοτήτων μετάβασης έτσι ώστε η σύγκλιση 4. να επιτυγχάνεται γρήγορα.

Σημαντική είναι και η επιλογή της αρχικής κατάστασης μ_0. Αν αυτή δεν είναι μια τυπική κατάσταση του συστήματος τότε θα πρέπει να περάσει κάποιος χρόνος μέχρι το σύστημα να βρεθεί σε "κατάσταση ισορροπίας" και να δειγματοληπτούμε πλέον από τη σωστή κατανομή. Σε μία τέτοια περίπτωση ο χρόνος που απαιτείται (**thermalization time**) μπορεί να είναι σημαντικό μέρος της προσπάθειάς μας αν γίνει λάθος επιλογή της αρχικής κατάστασης μ_0 ή των πιθανοτήτων μετάβασης ή και των δύο μαζί.

Απαραίτητη προϋπόθεση όμως για να πετύχουμε το δείγμα να ακολουθεί τη ζητούμενη κατανομή σε μία τέτοια διαδικασία είναι να ικανοποιείται το κριτήριο της **εργοδικότητας**.

2.3 Εργοδικότητα (Ergodicity)

Η συνθήκη της εργοδικότητας είναι απαραίτητη έτσι ώστε η Μαρκοβιανή αλυσίδα να μπορεί να περάσει από όλες τις δυνατές καταστάσεις μέσα από ένα πεπερασμένο πλήθος βημάτων. Ουσιαστικά η συνθήκη της εργοδικότητας μας επιτρέπει να θεωρήσουμε κάποιες από τις πιθανότητες μετάβασης της Μαρκοβιανής αλυσίδας μηδενικές, αλλά πρέπει πάντα να υπάρχει τουλάχιστον ένα μονοπάτι που συνδέει τις δύο καταστάσεις που έχουμε επιλέξει και έχει μη μηδενικές πιθανότητες μετάβασης. Στην πράξη οι περισσότεροι Μόντε Κάρλο αλγόριθμοι θέτουν σχεδόν όλες τις πιθανότητες μετάβασης μηδενικές, φροντίζοντας όμως παράλληλα ο αλγόριθμος που δημιουργείται να μην παραβιάζει τη συνθήκη της εργοδικότητας.

2.4 Συνθήκη Λεπτομερούς Ισοζύγισης (Detailed Balance)

Άλλη μία συνθήκη που πρέπει να ικανοποιεί μία Μαρκοβιανή διαδικασία είναι η συνθήκη λεπτομερούς ισοζύγισης. Είναι η συνθήκη που μας εξασφαλίζει ότι η κατανομή που παράγουμε είναι η κατανομή Boltzmann και όχι κάποια άλλη αφού το σύστημα φτάσει σε ισορροπία. Από την εξίσωση (1.2.1) μπορούμε εύκολα να καταλάβουμε ότι για να βρεθεί το σύστημα σε κατάσταση ισορροπίας στην κατανομή p_μ, οι πιθανότητες μετάβασης πρέπει να ικανοποιούν τη σχέση:

$$\sum_{\nu} p_\mu P(\mu \to \nu) = \sum_{\mu} p_\nu P(\nu \to \mu) \qquad (2.4.1)$$

Αυτό σημαίνει ότι ο ρυθμός με τον οποίο μεταβαίνει το σύστημα από τη κατάσταση μ σε κάποια άλλη, είναι ίσος με το ρυθμό με τον οποίο μεταβαίνει στην μ από κάποια άλλη. Προφανώς η σχέση (2.2.2) μας δίνει:

$$p_\mu = \sum_{\mu} p_\nu P(\nu \to \mu) \quad (2.4.2)$$

Η παραπάνω συνθήκη είναι αναγκαία αλλά δεν είναι ικανή. Μία ικανή αλλά όχι αναγκαία συνθήκη είναι η συνθήκη λεπτομερούς ισοζύγισης (detailed balanced) η

οποία όταν ικανοποιείται από τις πιθανότητες μετάβασης τότε είναι δυνατόν να δείξει κανείς ότι το σύστημα αργά ή γρήγορα θα φτάσει σε κατάσταση θερμικής ισορροπίας.

$$p_\mu P(\mu \to \nu) = p_\nu P(\nu \to \mu) \quad (2.4.3)$$

Αθροίζοντας και τα δύο μέλη της (2.4.3) προκύπτει η συνθήκη ισορροπίας (2.4.1). Για την κατανομή της κανονικής συλλογής έχουμε:

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p_\nu}{p_\mu} = e^{-\beta(E_\nu - E_\mu)} \quad (2.4.4)$$

Μπορεί να δειχθεί ότι αν οι πιθανότητες μετάβασης ικανοποιούν τις παραπάνω συνθήκες τότε η κατανομή ισορροπίας του συστήματος θα είναι η κατανομή Boltzmann (1.1.6). Ένα πρόγραμμα προσομοίωσης Monte Carlo μπορεί να συνοψιστεί στα παρακάτω βήματα:

1. Γράφουμε λογισμικό που κωδικοποιεί κατάλληλα επιλεγμένες πιθανότητες μετάβασης $P(\mu \to \nu)$ που ικανοποιούν την (2.4.4)

2. Επιλέγουμε κατάλληλη αρχική κατάσταση μ_0

3. Αφήνουμε το σύστημα να εξελιχθεί μέχρι να προσεγγίσουμε την κατανομή Boltzmann (1.1.6) (thermalization)

4. Συλλέγουμε δεδομένα για τις παρατηρήσιμες ποσότητες Q και ύστερα υπολογίζουμε τον εκτιμητή Q_M.

5. Σταματάμε μόλις έχουμε πετύχει την επιθυμητή ακρίβεια

Η εξίσωση (2.4.4) έχει πολλές λύσεις. Το ποια θα επιλέξουμε εξαρτάται από την αποδοτικότητά τους σε ένα συγκεκριμένο πρόβλημα και πρέπει να εξεταστεί κατά

περίπτωση. Παραδείγματα τέτοιων επιλογών είναι:

$$P(\mu \to \nu) = A \cdot e^{-\frac{1}{2}\beta(E_\nu - E_\mu)} \quad (2.4.5)$$

$$P(\mu \to \nu) = A \cdot \frac{e^{-\beta(E_\nu - E_\mu)}}{1 + e^{-\beta(E_\nu - E_\mu)}} \quad (2.4.6)$$

$$P(\mu \to \nu) = A \cdot \begin{cases} e^{-\beta(E_\nu - E_\mu)} & E_\nu - E_\mu > 0 \\ 1 & E_\nu - E_\mu \leq 0 \end{cases} \quad (2.4.7)$$

για κατάλληλα επιλεγμένες $\nu \neq \mu$ και

$$P(\mu \to \mu) = 1 - \sum_n P(\mu \to \nu) \quad (2.4.8)$$

ενώ $P(\mu \to \nu') = 0$ για οποιαδήποτε άλλη κατάσταση ν'. Οι σταθερές Α πρέπει να επιλεγούν κατάλληλα έτσι ώστε

$$\sum_{\mu \neq \nu} P(\mu \to \nu) < 1 \quad (2.4.9)$$

για να έχει νόημα η (2.4.8)

Η σχέση (2.4.9) μας δίνει μεγάλη ελευθερία στην επιλογή των πιθανοτήτων μετάβασης. Στην πράξη οι $P(\mu \to \nu)$ σπάνε σε δύο κομμάτια

$$P(\mu \to \nu) = g(\mu \to \nu) A(\mu \to \nu) \quad (2.4.10)$$

τα οποία αντιστοιχούν σε διακριτά βήματα στον αλγόριθμο.

Η πιθανότητα $g(\mu \to \nu)$ είναι η **πιθανότητα επιλογής** (transition probability) της κατάστασης ν όταν το σύστημα βρίσκεται στη κατάσταση μ. Το πρώτο βήμα δηλαδή είναι να επιλέξουμε μια κατάσταση $\nu \neq \mu$ με πιθανότητα $g(\mu \to \nu)$.

Το δεύτερο βήμα είναι να επιλέξουμε μια πιθανότητα $A(\mu \to \nu)$ αν το σύστημα θα μεταβεί στη κατάσταση ν. Αν η απάντηση είναι όχι τότε παραμένουμε στη

κατάσταση μ. Με το τρόπο αυτό ικανοποιείται η (2.4.8). Οι πιθανότητες $A(\mu \to \nu)$ λέγονται **λόγοι αποδοχής**.

Ο στόχος μας επικεντρώνεται στην εύρεση αλγορίθμου τέτοιου ώστε οι πιθανότητες επιλογής να δίνουν τους μέγιστους λόγους αποδοχής για καταστάσεις ν ασυσχέτιστες κατά το μέγιστο δυνατό από την κατάσταση μ. Ιδανική περίπτωση είναι να έχω $A(\mu \to \nu) = 1$ για όλα τα ν για τα οποία $g(\mu \to \nu) > 0$.

2.5 Ανεξαρτησία - Αυτοσυσχετισμός

Για να καταφέρουμε να δημιουργήσουμε ένα δείγμα από στατιστικά ανεξάρτητες μετρήσεις θα έπρεπε μέσα από μία διαδικασία Markov η επόμενη κατάσταση να μην είναι στατιστικά συσχετισμένη με τη προηγούμενη. Γίνεται όμως φανερό πως κάτι τέτοιο τις περισσότερες φορές δεν είναι εφικτό. Επομένως ανάλογα το πρόβλημα και τη μαρκοβιανή αλυσίδα του αλγόριθμου που χρησιμοποιούμε θα έχουμε ισχυρό ή ασθενή αυτοσυσχετισμό στις μετρήσεις του δείγματος που παράγουμε. Για την ποσοτική μελέτη του αυτοσυσχετισμού των διατάξεών μας, και στη προκειμένη περίπτωση των περιπάτων μας, θα χρησιμοποιήσουμε τη συνάρτηση αυτοσυσχετισμού (autocorrelation function). Έστω Q μία φυσική ποσότητα και $Q(t)$ η τιμή της μετά από Μόντε Κάρλο "χρόνο" t. Οι μονάδες t μπορεί να είναι στη περίπτωσή μας όπως θα δούμε και παρακάτω pivots ή πολλαπλάσια αυτών. Η συνάρτηση αυτοσυσχετισμού $\rho_Q(t)$ της φυσικής ποσότητας Q για κάθε χρόνο t ορίζεται όπως παρακάτω:

$$\rho_Q(t) = \frac{<(Q(t')-<Q>)(Q(t'+t)-<Q>)>_{t'}}{<(Q-<Q>)^2>} \quad (2.5.1)$$

όπου ο αριθμητής του κλάσματος είναι μια μέση τιμή παίρνοντας ως αρχική τιμή όλα τα στοιχεία του δείγματος με $t' < t_{max} - t$, ενώ η κανονικοποίηση που βάζουμε στο παρονομαστή είναι τέτοια ώστε $\rho_Q(0) = 1$.

Αυτό που θα συμβαίνει τελικά όταν η τιμή μιας ποσότητας μετά από χρόνο t είναι ισχυρά συσχετισμένη με την αρχική είναι το γινόμενο στον αριθμητή της (2.5.1) να είναι πάντοτε θετικό, με αποτέλεσμα η μέση τιμή να είναι πάντοτε θετική.

Αντίθετα, όταν ο συσχετισμός είναι ασθενής το γινόμενο θα είναι πότε θετικό και πότε αρνητικό, έτσι ώστε ο μέσος όρος να είναι μικρός. Αν υπάρχει αντισυσχετισμός τότε το κλάσμα είναι αρνητικό και αυτές οι τιμές πρέπει να απορρίπτονται γιατί οφείλονται στο πεπερασμένο μέγεθος του δείγματος. Η συνάρτηση $\rho_Q(t)$ θα ελαττώνεται ασυμπτωτικά με εκθετικό τρόπο:

$$\rho_Q(t) \sim e^{-t/\tau_Q} \quad (2.5.2)$$

Η κλίμακα χρόνου τ_Q χαρακτηρίζει τον αποσυσχετισμό των μετρήσεων του Q και ονομάζεται **χρόνος αυτοσυσχετισμού** της Q. Μετά από χρόνο $2\tau_Q$ θεωρούμε ότι η $\rho_Q(t)$ έχει πέσει στο $1/e^2 \approx 14\%$ της αρχικής τιμής. Συμβατικά θεωρούμε ότι τότε έχουμε μια στατιστικά ανεξάρτητη μέτρηση της Q. Επομένως αν έχουμε t_{max} μετρήσεις ο αριθμός των ανεξάρτητων μετρήσεων της Q θα είναι:

$$n_Q = \frac{t_{max}}{2\tau_Q} \quad (2.5.3)$$

Δηλαδή, αν μια μέτρηση κοστίζει σε χρόνο από τα παραπάνω βλέπουμε πως μας συμφέρει να τη μετράμε σε διαστήματα χρόνου $\sim \tau_Q$. Αν όμως το κόστος της μέτρησης δεν είναι μεγάλο τότε θα τη μετράμε συχνότερα και αυτό διότι ακόμη και στις συσχετισμένες μετρήσεις υπάρχει στατιστική πληροφορία. Σημειώνουμε πως όταν κάνουμε μέτρηση του τ_Q θα πρέπει να μετράμε και για $t << \tau_Q$.

Ένας καλύτερος εκτιμητής του χρόνου αυτοσυσχετισμού είναι ο ολοκληρωμένος χρόνος αυτοσυσχετισμού $\tau_{int,Q}$ (integrated autocorrelation time). Ο ορισμός του προκύπτει από την (2.5.2) και είναι:

$$\tau_{int,Q} = \int_0^{+\infty} dt \rho_Q(t) \sim \int_0^{+\infty} dt\, e^{-t/\tau_Q} = \tau_Q \quad (2.5.4)$$

Δηλαδή τελικά $\tau_{int,Q} \sim \tau_Q$, αλλά στον υπολογισμό υπεισέρχονται συστηματικά σφάλματα γι αυτό οι δύο ορισμοί μπορεί να διαφέρουν λίγο στη πράξη. Επίσης τα συστηματικά σφάλματα, ιδιαίτερα από τη συμπεριφορά της $\rho_Q(t)$ για μεγάλους

χρόνους t >> τ_Q επιδρούν με διαφορετικούς τρόπους στους δύο ορισμούς. Αν, το δείγμα είναι καλό τότε θα έχουμε καλύτερη συμφωνία. Επίσης, επειδή δεν μπορούμε ποτέ να έχουμε άπειρο δείγμα, θα πρέπει να επιλέξουμε ένα t_{max} που θα κόψουμε την ολοκλήρωση. Για μικρούς χρόνους δε θα έχουμε τη συμπεριφορά (2.5.2) και αυτό θα προσθέσει στο συστηματικό σφάλμα. Παρόμοια προβλήματα βέβαια έχουμε και όταν προσπαθούμε να υπολογίσουμε τον τ_Q με προσαρμογή των δεδομένων στην (2.5.2). Εκεί θα πρέπει να γίνει εκτίμηση των ορίων στο διάστημα του χρόνου κάτι που θέλει εμπειρία. Όταν θέλουμε μια γρήγορη εκτίμηση του χρόνου, αγνοούμε τα συστηματικά σφάλματα για μικρούς χρόνους και υπολογίζουμε τον

$$\tau_{\mathrm{int},Q} = \int_0^{t_{max}} dt \rho_Q(t) \quad (2.5.5)$$

Η εκτίμηση γίνεται κάνοντας τη γραφική παράσταση $\tau_{int,Q}(t_{max})$. Όταν παρατηρήσουμε την τιμή του $\tau_{int,Q}(t_{max})$ να σταθεροποιείται για κάποιο χρονικό διάστημα του του t_{max}, παίρνουμε αυτή την τιμή ως εκτιμητή του $\tau_{int,Q}$.

Η διαφορά των δύο χρόνων αυτοσυσχετισμού συνήθως είναι μικρή, παρόλα αυτά ο υπολογισμός του $\tau_{int,Q}$ προτιμάται. Αυτό όμως που μας ενδιαφέρει περισσότερο είναι η εξάρτηση αυτού από τις παραμέτρους του συστήματος.

3. Ο ΑΛΓΟΡΙΘΜΟΣ PIVOT

3.1 Εισαγωγή

Ο αλγόριθμος pivot είναι ένας γρήγορος Monte Carlo αλγόριθμος που προσομοιώνει το υπό μελέτη μοντέλο. Υπάρχουν πολλές υλοποιήσεις και μελέτες πάνω στο συγκεκριμένο μοντέλο. Σημαντική ήταν η μελέτη του από τον ALAN D. SOKAL στη δεκαετία του 90 και του TOM KENNEDY λίγο μετά το 2000. Εμείς θα χρησιμοποιήσουμε την υλοποίηση του τελευταίου που είναι και η πιο γρήγορη υπολογιστικά για τους λόγους που θα εξηγήσουμε σε παρακάτω παράγραφο.

3.2 Περιγραφή

Σε κάθε επανάληψη ο αλγόριθμος προτείνει ένα “pivot” κατά το οποίο ένα μέρος τη αλυσίδας κρατιέται σταθερό ενώ το υπόλοιπο περιστρέφεται σύμφωνα με κάποια συμμετρία που έχουμε ορίσει. Εάν ο καινούργιος περίπατος που προκύπτει είναι self-avoiding τον δεχόμαστε και συνεχίζουμε... αλλιώς τον απορρίπτουμε και ξαναπροσπαθούμε νέο pivot. Τα περισσότερα από τα προτεινόμενα pivots απορρίπτονται, γι αυτό στη μελέτη μας για την αποδοτικότητα του αλγορίθμου θα πρέπει να διακρίνουμε τον απαιτούμενο χρόνο της κάθε επανάληψης της Μαρκοβιανής αλυσίδας και τον απαιτούμενο χρόνο της κάθε επιτυχημένης προσπάθειας για pivot. Το κλάσμα των αποδεκτών pivots τείνει στο μηδέν με το μήκος σύμφωνα με N^{-p} για κάποιον εκθέτη $p>0$ ο οποίος εξαρτάται από τη διάσταση. Οπότε αυτοί οι δύο χρόνοι διαφέρουν κατά ένα παράγοντα N^p. Για παγκόσμιες ποσότητες όπως είναι η απόσταση από άκρη σε άκρη πιστεύεται ότι μόνο μερικά αποδεκτά pivots είναι αρκετά για να παραχθεί μία αποτελεσματικά ανεξάρτητη διαδρομή. Κατά συνέπεια μετρώντας το χρόνο που χρειάζεται για να προκύψει ένα αποδεκτό pivot μπορώ να υπολογίσω το χρόνο που χρειάζεται για να παράγω ένα ουσιαστικά ανεξάρτητο δείγμα μιας παγκόσμιας ποσότητας.

Ο πιο αφελής τρόπος για να ελέγξουμε τα self-intersections θέλει χρόνο $O(N^2)$. Με τη χρήση ενός hash table ο χρόνος μπορεί να γίνει της τάξης του $O(N)$. Αν ο έλεγχος αρχίσει από το σημείο του γίνεται το pivot και προχωρεί προς τα έξω τμήματα της αλυσίδας οι διασταυρώσεις διαπιστώνονται πολύ πιο γρήγορα. Από αυτή την άποψη πιστεύεται πως ο χρόνος που απαιτείται για τον έλεγχο διασταυρώσεων είναι $O(N^{1-p})$. Επομένως ο χρόνος για κάθε αποδεκτό pivot είναι $O(N)$, πιθανών με μία λογαριθμική διόρθωση. Ο απαιτούμενος χρόνος για την εφαρμογή του pivot είναι $O(N)$ και αυτός είναι απαραίτητος μόνο στη περίπτωση που έχουμε αποδεκτό pivot. Τελικά οι παλαιότερες υλοποιήσεις του αλγόριθμου pivot χρειαζόταν χρόνο $O(N)$ ανά αποδεκτό pivot.

Κάποιοι υποστήριζαν πως αυτός ο αλγόριθμος δε θα μπορούσε να γίνει γρηγορότερος αφού μόνο και μόνο η καταγραφή μιας διαδρομής μήκους N θέλει χρόνο $O(N)$. Παρόλα αυτά θα δείξουμε με ποιο τρόπο έχει υλοποιηθεί ώστε να γίνει γρηγορότερος με χρόνο $O(N^q)$ με $q<1$ ανά αποδεκτό pivot. Η υλοποίηση έχει γίνει

από τον καθηγητή **Tom Kennedy** και ο κώδικας που είναι ελεύθερο λογισμικό και καλύπτεται από την άδεια GNU General Public Licinse είναι διαθέσιμος στην ιστιοσελίδα http://math.arizona.edu/~tgk/. Μία ακριβής εκτίμηση του εκθέτη q είναι σχεδόν αδύνατη αλλά για τις δύο διαστάσεις εκτιμάται πως η τιμή του είναι μικρότερη από 0.57 και στις τρεις από 0.85. Η "ενεργός τιμή" του q εξαρτάται ισχυρά από μήκος του περιπάτου και μειώνεται με αυτό. Πρακτικά, παρομοιώσεις που έχουν γίνει για μακρείς περιπάτους στις δύο διαστάσεις, έχουν δείξει ότι ο αλγόριθμος pivot είναι γρηγορότερος κατά ένα παράγοντα γύρω στο 80 σε σχέση με παλαιότερες υλοποιήσεις του αυτοαποφεύγοντα τυχαίου περιπατητή. Στις τρεις διαστάσεις ο παράγοντας είναι περίπου 7.

3.3 Υλοποίηση (Tom Kennedy)

Με τον αλγόριθμο του pivot μπορούμε να εφαρμόσουμε μία Monte Carlo μέθοδο που προσομοιάζει τον αυτοαποφεύγοντα τυχαίο περιπατητή (SAW). Ο pivot προσδιορίζει μία Μαρκοβιανή αλυσίδα στο σύνολο των περιπάτων σταθερού μήκους. Έτσι μπορούμε να παράγουμε δείγματα περιπάτων τρέχοντας τη διαδικασία Markov. Μία επανάληψη της διαδικασίας Markov ξεκινάει διαλέγοντας στη τύχη ένα σημείο του περιπάτου. Στο επόμενο βήμα γίνεται η επιλογή μίας συμμετρίας g του πλέγματος. Το μέρος του περιπάτου από την αρχή έως το σημείο που επιλέχθηκε στοχαστικά παραμένει το ίδιο. Το υπόλοιπο περιστρέφεται (κάνει pivot) εφαρμόζοντας τη g. Πρέπει να πούμε ότι ο αλγόριθμος αποδεικνύεται ότι ικανοποιεί τη συνθήκη **λεπτομερούς ισοζύγισης** αν οι πιθανότητες επιλογής g και g^{-1} είναι ίσες και ότι είναι **εργοδικός**.

Σημειώνουμε τα πλεγματικά σημεία του περιπάτου ως $\omega(i)$ με $0 \leq i \leq N$. Οι περίπατοι ξεκινούν από την αρχή των αξόνων οπότε $\omega(0) = 0$. Ο περιορισμός του πλησιέστερου γείτονα για τα διαδοχικά σημεία του περιπάτου σημαίνει ότι $\|\omega(i) - \omega(i-1)\| = 1$ για $0 \leq i \leq N$. Το γεγονός ότι είναι αυτοαποφεύγων σημαίνει ότι $\omega(i) \neq \omega(j)$ για $i \neq j$. Είναι παραπλανητικό να σκεφτόμαστε το δείκτη i με την έννοια του χρόνου και γι αυτό όταν αναφερόμαστε σ' αυτόν θα εννοούμε σημεία κατά μήκος του περιπάτου.

Υπάρχουν δύο κύρια βήματα στον αλγόριθμο του pivot που στις μέχρι πρότινος υλοποιήσεις του αμφότερα περιορίζουν την απόδοσή του σε O(N). Το πρώτο είναι ο έλεγχος για διασταυρώσεις με τον εαυτό του (self-intersections) και το δεύτερο είναι η εφαρμογή των pivots.

Το σημείο κλειδί έτσι ώστε να ξεπεράσουμε το υπολογιστικό κόστος που οφείλεται στο πρώτο βήμα είναι να εκμεταλλευτούμε το γεγονός ότι ο περίπατος εκτελεί βήματα μόνο μεταξύ των πλησιέστερων γειτόνων. Όταν συγκρίνουμε δύο σημεία i και j του περιπάτου, δεν ελέγχουμε απλά αν ω(i) = ω(j). Αντί γι' αυτό υπολογίζουμε την απόσταση $d = \|\omega(i) - \omega(j)\|$. Σημειώνουμε ότι η νόρμα που πρέπει να χρησιμοποιηθεί είναι ο ελάχιστος αριθμός βημάτων μεταξύ πλησιέστερων γειτόνων που χρειάζονται ώστε να πάμε από το ω(i) στο ω(j). Εάν το d είναι μη μηδενικό τότε μπορούμε να συμπεράνουμε όχι μόνο ότι $\omega(i) \neq \omega(j)$, αλλά επίσης ότι:

$$\omega(i') \neq \omega(j'), \qquad \alpha\nu \qquad |i - i'| + |j - j'| < d \quad (3.3.1)$$

Επομένως μπορούμε να αποκλείσουμε έναν μεγάλο αριθμό δυνητικών διασταυρώσεων αν το d είναι μεγάλο. Φυσικά, αυτή η παρατήρηση είναι μάλλον άχρηστη εκτός και αν βρούμε έναν αλγόριθμο που θα επιλέγει ποιες τιμές i και j να ελέγχει. Πριν όμως αναφερθούμε σε αυτόν ας εξηγήσουμε καλύτερα πως η παρατήρηση (3.3.1) μπορεί να χρησιμοποιηθεί.

Έστω l η τοποθεσία μέσα στον περίπατο που θα γίνει το pivot. Ας επιλέξουμε ένα i ανάμεσα στο l+1 και στο N. Θέλουμε να ελέγξουμε αν $\omega(i)$ είναι ίσο με $\omega(j)$ για κάθε j από 1 έως l – 1. Μία αφελής προσέγγιση θα ήταν απλώς να ελέγξουμε για όλες τις τιμές από l - 1 έως 1. Η ιδέα είναι δεδομένου ενός j να υπολογίσουμε την απόσταση $d = \|\omega(i) - \omega(j)\|$. Αν d = 0 έχουμε βρει διασταύρωση και έχουμε τελειώσει με τον έλεγχο. Αν $d > 0$ τότε ξέρουμε πως το $\omega(i)$ δεν συμπίπτει με το $\omega(j)$, αλλά επίσης μπορούμε να συμπεράνουμε ότι και το $\omega(j-1)$,..., $\omega(j-d+1)$ δεν συμπίπτουν επίσης με το $\omega(i)$ αφού ο περίπατος κάνει βήματα μόνο μεταξύ πλησιέστερων γειτόνων. Επομένως αντί να μειώσουμε το j κατά 1, μπορούμε με ασφάλεια να το μειώσουμε κατά d. Αυτό θα επιταχύνει δραματικά τον έλεγχο των διασταυρώσεων σε σύγκριση με την αφελή προσέγγιση του ελέγχου όλων των i και j, αλλά και πάλι χρειάζονται τουλάχιστον O(N) πράξεις για να ελέγξουμε έναν περίπατο

χωρίς διασταυρώσεις αφού πρέπει να γίνει έλεγχος για κάθε τιμή του i από $l + 1$ έως N. (Στην πραγματικότητα χρειάζονται πολύ περισσότερες πράξεις από αυτές)

Με βάση τον παραπάνω μηχανισμό, ο αλγόριθμος που θα χρησιμοποιήσουμε είναι ο εξής. Ορίζουμε l τη θέση μέσα στον περίπατο που θα γίνει το pivot. Σε όλο τον αλγόριθμο i και j θα είναι σημεία του περιπάτου τέτοια ώστε $j < l < i$ και με την παρακάτω ιδιότητα:

$$\omega(i') \neq \omega(j') \qquad \forall i', j' \qquad j < j' < l < i' < i \quad (3.3.2)$$

Αρχικά, $i = l + 1$ και $j = l - 1$. Σε κάθε βήμα ο αλγόριθμος είτε μειώνει το j είτε αυξάνει το i με τέτοιο τρόπο έτσι ώστε η (3.3.2) να παραμένει αληθής. Ο τρόπος που αυξάνεται το i και μειώνεται το j είναι απόλυτα ανάλογος. Θα εξηγήσουμε μόνο τη διαδικασία όπου αυξάνεται το i. Έστω m_i η απόσταση από το $\omega(i)$ έως το $\{\omega(k): j < k < l\}$

$$m_i = min\{||\omega(i) - \omega(k)||: j < k < l\} \quad (3.3.3)$$

Αν $m_i = 0$ τότε υπάρχει διασταύρωση.

Αν $m_i > 0$ τότε ξέρω ότι $\omega(i') \neq \omega(j')$ για κάθε i', j' με $i \leq i' + m_i$ και $j < j' < l$. Οπότε αντί να αυξήσουμε το i κατά 1, μπορούμε να το αυξήσουμε κατά m_i. Δεν είναι απαραίτητο να υπολογίσουμε το m_i ακριβώς. Μπορούμε να προσδιορίσουμε ένα κατώτατο όριο b_i του m_i και να αυξήσουμε το i κατά b_i. Στην πραγματικότητα αυτό είναι που κάνει ο αλγόριθμος χρησιμοποιώντας ένα loop πάνω στο j' για τιμές από $l - 1$ έως j ως εξης:

Αρχικά θέτουμε $b_i = N$. Πριν ακόμη συγκρίνουμε τα $\omega(i)$ και $\omega(j')$, το b_i θα είναι το κατώτατο όριο του m_i για την απόσταση από $\omega(i)$ έως $\{\omega(k): j' + 1 \leq k < l\}$. Ας υποθέσουμε τώρα ότι $d = \|\omega(i) - \omega(j')\|$. Διαλέγουμε έναν ακέραιο s με τον περιορισμό $s < d$. Τότε η απόσταση από το $\omega(i)$ έως το $\{\omega(k): j' - s \leq k < l\}$ θα είναι τουλάχιστον $d - s$. Άρα αν αντικαταστήσουμε το b_i με το $\min\{b_i, d - s\}$, το νέο b_i θα είναι το κατώτατο όριο της απόστασης του $\omega(i)$ με το $\{\omega(k): j' - s \leq k < l\}$. Επομένως μπορούμε να μειώσουμε το j' κατά $1 + s$. Τελικά γίνει το j' ίσο με j το b_i θα είναι το κατώτατο όριο του m_i.

Υπάρχουν πολλοί τρόποι για να διαλέξουμε το *s*. Σημειώστε πως ο μοναδικός περιορισμός για την επιλογή του είναι: $s < d$ όπου $d = \|\omega(i) - \omega(j')\|$. Η πιο απλή επιλογή είναι να πάρουμε το *s* ίσο με *d*/2. (Αν το *d* είναι περιττό στρογγυλοποιούμε το *d*/2 προς τα κάτω) Παρόλα αυτά έχει βρεθεί πως ο αλγόριθμος είναι σημαντικά γρηγορότερος με την εξής επιλογή:

- Αν $d < b_i$ παίρνουμε $s=d/2$
- Αν $d \geq b_i$ παίρνουμε $s=d - b_i$ αυτό αφήνει το b_i ίδιο και μειώνει το j' κατά $1 + d - b_i$.

Ο έλεγχος για διασταυρώσεις θα τελειώσει όταν $j \leq -1$ και $i \geq N + 1$. Αν φτάσουμε έως αυτό το σημείο τότε ξέρουμε πλέον ότι το pivot είναι αποδεκτό. Όσο $j > -1$ και $i < N + 1$, έχουμε την ελευθερία είτε να αυξήσουμε *i*, είτε να μειώσουμε το *j*. Η επιλογή που τελικά κάνουμε είναι να αυξήσουμε το *i* εάν είναι πιο κοντά στο *l*, ή σε διαφορετική περίπτωση να μειώσουμε το *j*. Η επιλογή αυτή σημαίνει ότι το *i* και το *j* απομακρύνονται από το *l* με τον ίδιο ρυθμό. Δηλαδή ο αλγόριθμος ελέγχει για διασταυρώσεις κοντά στο *l* πριν ελέγξει σε θέσεις που είναι μακριά από αυτό.

Δεδομένου όμως ότι χρειάζεται χρόνος τάξης O(N) για να καταγραφεί ένας περίπατος *N* βημάτων, φαίνεται ότι το δεύτερο βήμα του αλγορίθμου θα διατηρήσει το υπολογιστικό κόστος σε O(N). Για να δούμε βελτίωση σ' αυτό το τομέα η κεντρική ιδέα είναι να μην εφαρμόζουμε το pivot κάθε φορά που είναι αποδεκτό, αλλά να σημειώνουμε τα pivots που γίνονται αποδεκτά και να τα εφαρμόζουμε μόνο μετά από έναν συγκεκριμένο πλήθος.

Το pivot δρα σε έναν περίπατο ω και παράγει έναν νέο περίπατο $\bar{\omega}$ σύμφωνα με την εξίσωση:

$$\bar{\omega}(j) = \begin{cases} \omega(j), & \text{για } j \leq l \\ g[\omega(j) - \omega(l)] + \omega(l), & \text{για } j \geq l \end{cases} \quad (3.3.4)$$

Εδώ *l* είναι μία θέση τέτοια ώστε $0 \leq l < N$, στην οποία θα αναφερόμαστε ως τη θέση μέσα στον περίπατο που θα γίνει το pivot και *g* είναι η συμμετρία που εφαρμόζεται. Το pivot καθορίζεται πλήρως από το τα *l* και *g*, οπότε δημιουργώντας μία λίστα με

αυτά έχουμε μία λίστα από πλήρως καθορισμένα pivots. Εμείς όμως δε θα κάνουμε αυτό ακριβώς διότι ο χρόνος που χρειάζεται για να υπολογίσουμε μία θέση στον περίπατο είναι σημαντικός. Αντίθετα θα αναπαραστήσουμε τον περίπατο με τον παρακάτω τρόπο.

Ας υποθέσουμε ότι έχουμε δεχθεί n pivots και πως οι θέσεις αυτών στον περίπατο είναι $l_1<l_2<...<l_n$. Σημειώστε πως τα l είναι κατά αύξουσα σειρά και όχι στη σειρά που προτάθηκαν και έγιναν αποδεκτά. Τότε ω θα είναι ο περίπατος μετά από αυτά τα pivots και ω' πριν από αυτά. Μπορούμε να υποθέσουμε ένα τμήμα του ω' από τις τοποθεσίες l_i έως l_{i+1} που παραμένει ακίνητο. Το αντίστοιχο τμήμα του ω λαμβάνεται απλά, εφαρμόζοντας μία συμμετρία g, και μεταφράζεται στο τμήμα του ω'. Η αναπαράσταση του περιπάτου γίνεται με την παρακάτω δομή.

(i) ο παλιός περίπατος ω'. Αυτός ο περίπατος είναι ένας αριθμός επαναλήψεων πριν από το παρόν

(ii) Ένας ακέραιος n που είναι ο αριθμός των pivots που έχουν γίνει αποδεκτά αλλά δεν έχουν εφαρμοσθεί ακόμη

(iii) Οι θέσεις των pivots στον περίπατο, $l_1 < l_2 < ... < l_n$

(iv) Οι συμμετρίες πλέγματος $g_1, g_2,...,g_n$

(v) Τα σημεία στο πλέγμα $x_1, x_2,..., x_n$

Ο περίπατος ω προσδιορίζεται από αυτά τα δεδομένα σύμφωνα με την εξίσωση:

$$\omega(j)=g_i\omega'(j)+x_i, \qquad για \quad l_i\le j\le l_{i+1} \quad (3.3.5)$$

με l_0 να είναι 0 και l_n N. Μπορούμε να παρατηρήσουμε έναν πλεονασμό σ' αυτή τη δομή. Λαμβάνοντας υπόψη το γεγονός ότι ο ω είναι ένας περίπατος με βήματα μεταξύ των πλησιέστερων γειτόνων μόνο, κάποιος θα μπορούσε να προσδιορίσει τα x_i από τα υπόλοιπα δεδομένα. Παρόλα αυτά κάτι τέτοιο θα χρειαζόταν σημαντικό χρόνο και

είναι γρηγορότερο να συμπεριλάβουμε και αυτά στην παραπάνω δομή δεδομένων.

Όταν προτείνεται ένα pivot δεν το εισάγουμε αμέσως στις λίστες *(iii) – (v)*. Ας υποθέσουμε ότι ω είναι ο περίπατος πριν το pivot και $\bar{\omega}(j)$ ο περίπατος μετά το pivot. Για να ελέγξουμε εάν το προτεινόμενο pivot πρέπει να γίνει αποδεκτό χρειάζεται να υπολογίσουμε το $\bar{\omega}(j)$ για επιλεγμένες τιμές του *j*. Χρησιμοποιούμε τη δομή των δεδομένων και την (3.3.5) για να υπολογίσουμε το *ω(j)* και μετά χρησιμοποιούμε την (3.3.4) για να υπολογίσουμε το $\bar{\omega}(j)$.

Τώρα ας υποθέσουμε ότι έχουμε δεχθεί ένα pivot στη θέση *l* με συμμετρία πλέγματος *g*. Τότε ο καινούργιος περίπατος $\bar{\omega}(j)$ θα δίνεται από την (3.3.4). Πρέπει να προσδιορίσουμε πώς θα ενημερώσουμε τη δομή των δεδομένων. Αυτό θα το κάνουμε σε δύο βήματα. Πρώτα προσθέτουμε απλώς τη θέση του pivot στη λίστα. Έστω *k* τέτοιο ώστε $l_k < l < l_{k+1}$. Τότε οι αλλαγές θα είναι:

(ii) $n \rightarrow n + 1$

(iii) $l_1, l_2,\ldots,l_n \rightarrow l_1, l_2,\ldots, l_k, l, l_{k+1},\ldots,l_n$

(iv) $g_1, g_2,\ldots, g_n \rightarrow g_1, g_2,\ldots,g_{k-1}, g_k, g_k, g_{k+1},\ldots, g_n$

(v) $x_1, x_2,\ldots, x_n \rightarrow x_1, x_2,\ldots,x_{k-1}, x_k, x_k, x_{k+1},\ldots,x_n$

Σ' αυτό το στάδιο η δομή των δεδομένων εξακολουθεί να αναπαριστά τον περίπατο πρίν το pivot. Είναι δυνατόν το *l* να είναι το ίδιο με ένα από τα l_i. Σ' αυτή τη περίπτωση απλώς προσπερνάμε το πρώτο βήμα.

Το δεύτερο βήμα είναι η εφαρμογή του pivot κάτω από την προϋπόθεση ότι το *l* έχει τοποθετηθεί στη λίστα $l_1, l_2,\ldots, l_n$. Έστω $l = l_k$. Δεχόμενοι ότι το *j* ικανοποιεί την $l_i \le j \le l_{i+1}$ με $i \ge k$. Τότε

$$\bar{\omega}(j) = g[\omega(j) - x] + x = g[g_i\omega'(j) + x_i - x] + x = g\,g_i\omega'(j) + gx_i - gx + x \quad (3.3.6)$$

όπου $x = \omega(l_k)$. Τότε για $i \ge k$,

$$g_i \rightarrow g\, g_i \quad (3.3.7)$$

$$x_i \rightarrow g\, x_i - gx + x \quad (3.3.8)$$

Για $i < k$, g_i και x_i δεν έχουν υποστεί αλλαγή. Και στα δύο βήματα ο "παλιός" περίπατος μένει ανεπηρέαστος.

Χρησιμοποιούμε λοιπόν αυτή τη δομή δεδομένων για να εφαρμόσουμε το pivot ως εξής. Θέτουμε έναν μεγάλο ακέραιο N_{pivot} ο οποίος θα είναι μικρός συγκρινόμενος το N, τον αριθμό των βημάτων στον περίπατο. Καθώς παίρνουμε αποδεκτά pivots, ενημερώνουμε τη δομή των δεδομένων όπως εξηγήσαμε παραπάνω. Όταν το n γίνει ίσο με το N_{pivot} χρησιμοποιούμε την (3.3.5) για να υπολογίσουμε τον περίπατο ω και να αντικαταστήσουμε τον ω' στη δομή των δεδομένων. Στη συνέχεια θέτουμε n = 0 και σβήνουμε τις λίστες (iii) – (v). Σημειώνουμε ότι ο ακέραιος n στο (ii) της δομής δεδομένων δεν είναι πάντοτε ίσος με τον αριθμό των αποδεκτών pivots που δεν έχουν εφαρμοστεί ακόμα. Μπορεί να είναι ελαφρώς μικρότερος καθώς μερικές θέσεις pivot που γίνονται αποδεκτές μπορεί να υπάρχουν ήδη στη λίστα (iii).

Όπως είπαμε και προηγουμένως, αυτή η δομή των δεδομένων είναι χρήσιμη μόνο εάν μπορούμε να υπολογίσουμε το *ω(j)* γρήγορα για δεδομένο *j*. Αυτό γίνεται με την (3.3.5). Το ουσιαστικό μέρος είναι η εύρεση ενός *i* τέτοιου ώστε $l_i < j < l_{i+1}$. Αυτό μπορεί να γίνει σε χρόνο τάξης ln(*n*).

3.4 Θεωρητική ανάλυση

Υπάρχουν τρία βήματα στον αλγόριθμο του pivot για τα οποία ο χρόνος που απαιτείται εξαρτάται από το μήκος των περιπάτων.

1. Για κάθε pivot που προτείνεται πρέπει να αποφασίσουμε αν θα το αποδεχτούμε ή όχι

2. Για κάθε αποδεκτό pivot πρέπει να ενημερώσουμε τα στοιχεία (ii) – (v) στη δομή των δεδομένων

3. Για κάθε N_{pivot} αποδεκτά pivots πρέπει να τα εφαρμόσουμε, ενημερώνοντας ουσιαστικά το στοιχείο (i) της δομής δεδομένων

Μιας και δε μπορούμε να προσδιορίσουμε a priori πόσα βήματα χρειάζονται για τον έλεγχο διασταυρώσεων, η ανάλυση του πρώτου βήματος θα είναι απαιτεί μια κατά κάποιο τρόπο εμπειρική μελέτη. Σ αυτό το βήμα πρέπει να χρησιμοποιήσουμε την (3.3.5) επαναλαμβανόμενα. Σ' αυτήν την εξίσωση μας δίνεται το *j* και πρέπει να βρούμε ένα *i* τέτοιο ώστε $l_i < j < l_{i+1}$. Οι λίστες (iii) – (v) αποθηκεύονται ως γραμμικοί πίνακες σε σειρά που προσδιορίζεται από τη συνθήκη $l_1 < l_2 < ... < l_n$. Εφαρμόζοντας τη μέθοδο της διχοτόμησης το *i* μπορεί να βρεθεί σε χρόνο τάξης ln(*n*), που κατά μέσο όρο είναι O(ln(N_{pivot})). Έστω D(N) ο αριθμός των υπολογισμών της απόστασης $\|\omega(i) - \omega(j)\|$ ανά αποδεκτό pivot. Υποθέτουμε ότι D(N) μεγαλώνει ως N^{σ}. Εκτός από τους υπολογισμούς που έχουν να κάνουν με την εύρεση του *i* στην (3.3.5), ο αριθμός των βημάτων που χρειάζονται για τον έλεγχο των διασταυρώσεων ανα αποδεκτό pivot είναι ανάλογος του D(N). Επομένως ο χρόνος που απαιτείται για το πρώτο βήμα είναι $O(N^{\sigma}\ln(N_{pivot}))$ ανά αποδεκτό pivot. Η εκτίμηση του σ γίνεται τρέχοντας τις προσομοιώσεις και μετρώντας τον αριθμό των υπολογισμών της απόστασης $\|\omega(i) - \omega(j)\|$. Αυτό το πλήθος στη συνέχεια το διαιρούμε με τον αριθμό των αποδεκτών pivots.

Όσον αφορά στο δεύτερο βήμα, δεδομένου ότι οι λίστες (iii) – (v) αποθηκεύονται ως γραμμικοί πίνακες το πλήθος των διαδικασιών που χρειάζονται για να εισαχθούν τα νέα δεδομένα είναι της τάξης του n. Ακόμη πρέπει να εφαρμόσουμε τις ενημερώσεις (3.3.7) και (3.3.8). Αυτό κατά μέσο όρο χρειάζεται επίσης n διαδικασίες. Η εισαγωγή των νέων δεδομένων θα μπορούσε να γίνει πιο γρήγορα μία πιο έξυπνη δομή δεδομένων, αλλά από τη στιγμή που οι (3.3.7) και (3.3.8) χρειάζονται n διαδικασίες, δεν είναι ξεκάθαρο αν μπορεί να υπάρξει σημαντική βελτίωση. Οπότε ο χρόνος που χρειάζεται το δεύτερο βήμα είναι τάξης n. Καθώς το n μεγαλώνει παίρνει τιμές από 0 έως N_{pivot}, κατά μέσο όρο ο χρόνος που απαιτείται είναι $O(N_{pivot})$.

Τέλος, ας δούμε το τρίτο βήμα. Ο περίπατος στην (i) ενημερώνεται χρησιμοποιώντας την (3.3.5). Η εύρεση του *i* μπορεί να αγνοηθεί εδώ. Μπορούμε απλά να σχηματίσουμε ένα βρόγχο στο *i* και μέσα σ' αυτόν να σχηματίσουμε έναν

βρόγχο στο $j = l_i,...,l_{i+1}$. Επομένως κάθε εφαρμογή της (3.3.5) χρειάζεται χρόνο τάξης 1. Άρα για να ενημερώσουμε τον περίπατο χρειάζεται χρόνος τάξης N. Επειδή όμως αυτό γίνεται κάθε N_{pivot} αποδεκτά pivots ο χρόνος για κάθε αποδεκτό pivot είναι O(N/ N_{pivot}).

Τελικά ο συνολικός χρόνος ανα αποδεκτό pivot είναι:

$$O(N^{\sigma}\ln(N_{pivot}))+O(N_{pivot})+O\left(\frac{N}{N_{pivot}}\right) \quad (3.3.9)$$

Θεωρούμε ότι το N_{pivot} είναι ανάλογο του $N^{1/2}$. Έχει βρεθεί ότι στις προσομοιώσεις η επιλογή $N_{pivot} = (N/40)^{1/2}$ είναι καλή επιλογή. Τότε ο συνολικός χρόνος ανά αποδεκτό pivot θα είναι:

$$O(N^{\sigma}\ln(N))+O(N^{1/2}) \quad (3.3.10)$$

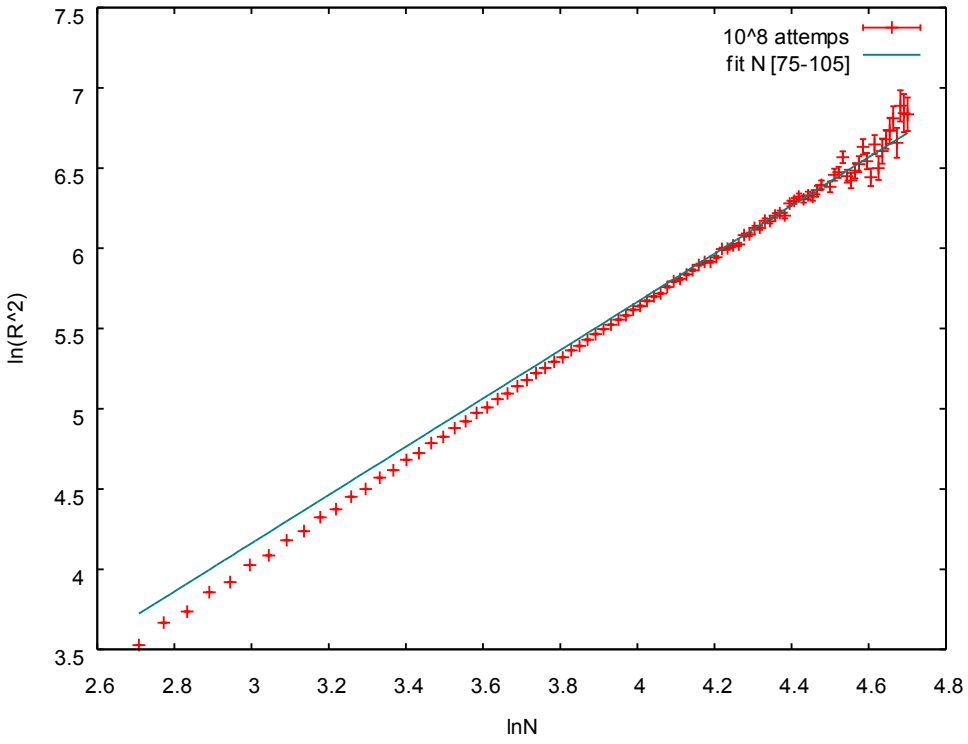
4. Προσομοιώσεις – Αποτελέσματα – Ανάλυση

4.1 Απλή δειγματοληψία (Simple sampling)

Οι προσομοιώσεις μας για το μοντέλο του αυτοαποφεύγοντα τυχαίου περιπατητή στις δύο διαστάσεις με τη μέθοδο της απλής δειγματοληψίας, έγιναν χρησιμοποιώντας το μοντέλο του μη-επιστρέφοντα τυχαίου περιπατητή (Non-Revearsal Random Walker – NRRW). Η υλοποίηση είναι του κ. Αναγνωστόπουλου. Ο κώδικας σε σύγκριση με τον NRRW έχει τη προσθήκη ενός σταδίου που σε κάθε βήμα ελέγχει τον περίπατο για διασταυρώσεις με τον εαυτό του. Αυτό λαμβάνει χώρα μέσω ενός πίνακα του οποίου τα στοιχεία αντιστοιχούν στα σημεία του πλέγματος που κινείται ο περιπατητής. Ο ρόλος του είναι να κρατάει σημαίες για τις κατειλημμένες από τον περίπατο θέσεις του πλέγματος. Ας υποθέσουμε ότι μετά από έναν αριθμό βημάτων, και δεδομένου ότι ο μέχρι τώρα περίπατος είναι αυτοαποφεύγων, προτείνεται στοχαστικά το επόμενο βήμα. Σε περίπτωση που το τυχαίο βήμα μας οδηγήσει σε θέση (i,j), θέση που δεν έχει περάσει ο περίπατος έως

τώρα (κατεβασμένη σημαία πχ xy[i,j] = 0), το βήμα γίνεται αποδεκτό, σηκώνεται η σημαία κατάληψης της θέσης και προτείνεται νέο βήμα. Σε περίπτωση όμως που το τυχαίο βήμα μας οδηγήσει σε θέση που έχει περάσει ήδη ο περίπατος (σηκωμένη σημαία πχ. xy[i,j] = 1), τότε το βήμα απορρίπτεται και αποθηκεύεται ο τρέχων περίπατος, με το μήκος που έχει καταφέρει να αποκτήσει.

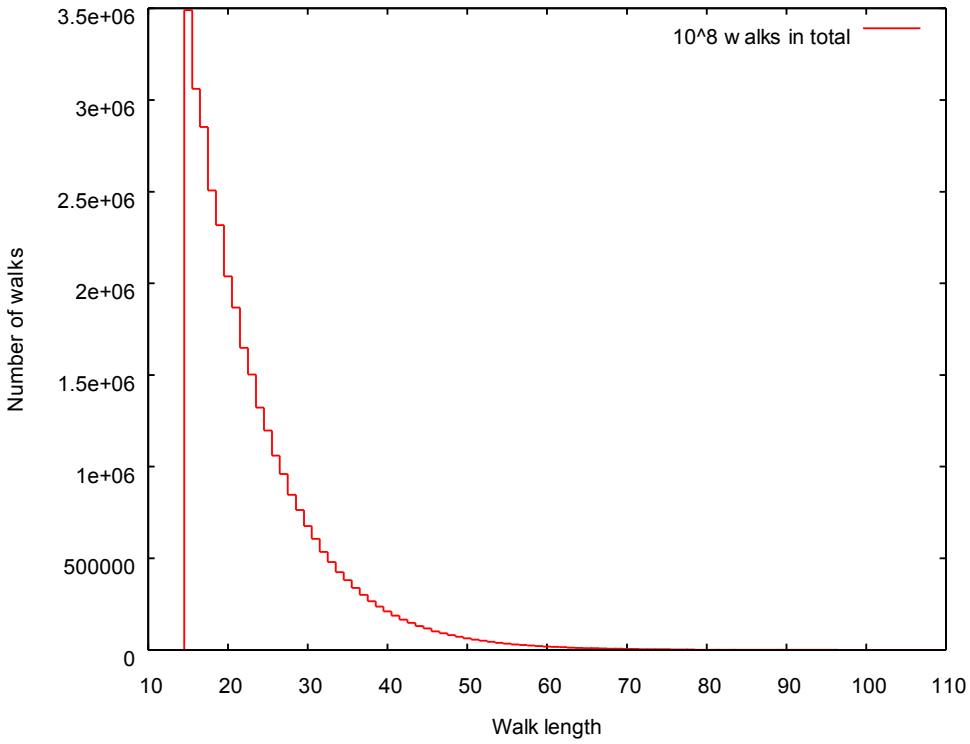
Τρέχουμε τον κώδικα ώστε να παράγουμε 100,000,000 περιπάτους και καταγράφουμε αυτούς που καταφέρνουν από N = 15 βήματα και πάνω. Στη συνέχεια κάνουμε τη γραμμική προσαρμογή $\ln(R^2)$ - $\ln(N)$ που φαίνεται στο σχήμα 4.1 από όπου προκύπτει ο κρίσιμος και παγκόσμιος εκθέτης ν. Για προσαρμογή από N = 75 – 105, μετρήσαμε ότι $2\nu = 1.50 \pm 0.06$. Παρατηρούμε ότι για μικρά μήκη περιπάτων N έως 50 με 60 περίπου, η βάθμωση του R^2 δεν είναι η αναμενόμενη. Αυτό όμως οφείλεται στο ότι βρισκόμαστε ακόμη πολύ μακριά από το όριο $N \to \infty$ (finite size effects). Αυξάνοντας σε μήκος παρατηρούμε και επιβεβαιώνουμε τη συμπεριφορά που περιμένουμε.



Σχήμα 4.1

Συνεχίζοντας όμως για μήκη μεγαλύτερα των 100 περίπου, βλέπουμε πως οι μετρήσεις παρουσιάζουν αρκετό θόρυβο. Εδώ αρχίζει να φαίνεται έντονα η αδυναμία του αλγορίθμου να παράγει αρκετά δείγματα ώστε να έχουμε καλή στατιστική. Μία ποιοτική και ποσοτική εικόνα γι' αυτό μπορούμε να έχουμε από το σχήμα 4.2 όπου φαίνεται και η εκθετική μείωση της πιθανότητας που έχει ο αλγόριθμος να παράγει SAW. Η μείωση αυτή είναι

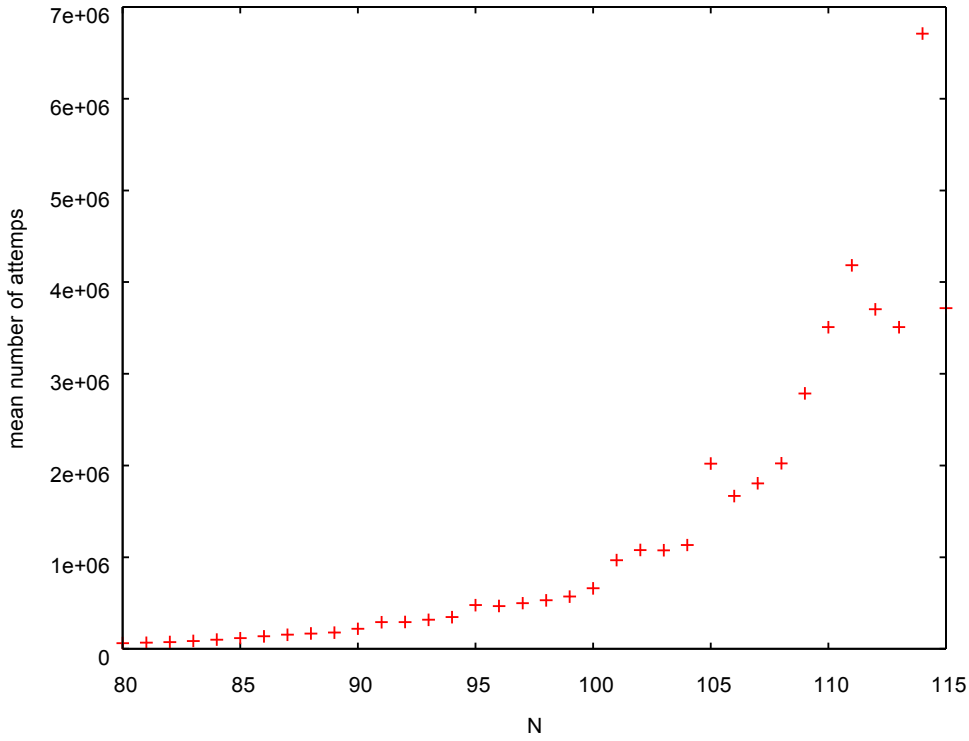
$$\sim e^{-\lambda N} N^{\gamma-1}$$



Σχήμα 4.2

Όπως έχουμε ήδη αναφέρει στην παράγραφο 2.1.1, ο μέσος όρος των προσπαθειών που απαιτούνται για να δημιουργήσουμε ένα SAW μήκους N με τον RW είναι $(2d)^N/c_N$(σχήμα 4.3). Αντίστοιχα, στο μοντέλο του NRRW που χρησιμοποιούμε εμείς είναι $2d(2d\text{-}1)^{N\text{-}1}/c_N$ που μεγαλώνει με ρυθμό $e^{\lambda N}$. λ είναι η *σταθερά φθοράς* ίση με $ln[(2d\text{-}1)/\mu]$. Αυτό θα προσπαθήσουμε να επιβεβαιώσουμε τώρα. Κάναμε μία ακόμη προσομοίωση δημιουργώντας 10^9 περιπάτους. (Για οικονομία δεν υπολογίσαμε κανένα μέγεθος αλλά καταγράψαμε μόνο πόσοι

κατάφεραν να επιτύχουν μήκος N, για N > 54). Στο παρακάτω διάγραμμα φαίνεται ο μέσος όρος των προσπαθειών που χρειάζεται ο αλγόριθμος για να παράγει μία διαδρομή για κάθε μήκος.



Σχήμα 4.3

Εφαρμόζοντας γραμμική προσαρμογή όπως φαίνεται στο σχήμα 4.4, υπολογίζουμε ότι η σταθερά φθοράς $\lambda = 0.129 \pm 0.002$. Στην βιβλιογραφία υπολογίζεται επίσης 0.129.

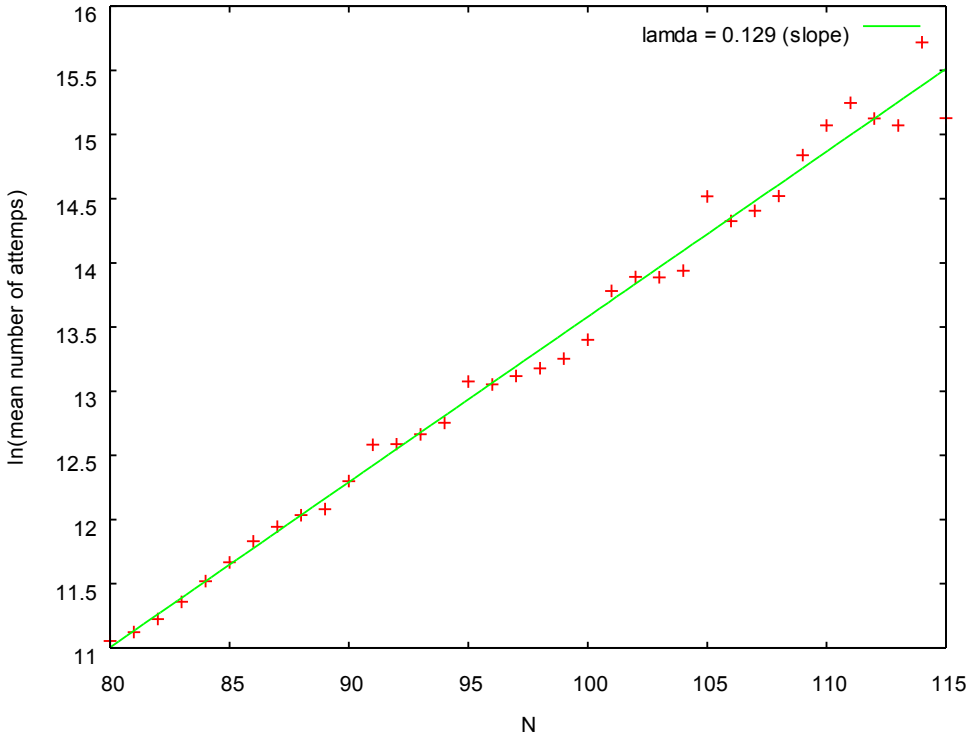
Από την (2.1.1.4) προκύπτει ότι:

$$\mu = \frac{2d-1}{e^{\lambda}} \quad (4.1.1)$$

και

$$\delta\mu = \left|\frac{\partial \mu}{\partial \lambda}\right| \delta\lambda = \left|\frac{-(2d-1)}{e^{\lambda}}\right| \delta\lambda \quad (4.1.2)$$

και τελικά τη συνδετική σταθερά του πλέγματος μ την υπολογίζουμε $\mu = 2.637 \pm 0.005$. Στη βιβλιογραφία υπολογίζεται 2.638 158 5 (10).



Σχήμα 4.4

Η παραπάνω προσομοίωση έγινε χωρίς να λάβουμε υπόψη μας τη θερμοκρασία. Αλλιώς, θεωρήσαμε ότι η θερμοκρασία είναι άπειρη και κατ' επέκταση η ενέργεια αλληλεπίδρασης των μονομερών του πολυμερούς μηδενική. Στη συνέχεια θα ξαναδούμε το πρόβλημα λαμβάνοντας υπόψη τον παράγοντα αυτό. Αυτό που κάναμε εμείς ήταν να τροποποιήσουμε την υλοποίηση του κ. Αναγνωστόπουλου έτσι ώστε σε κάθε περίπατο που παράγει, να υπολογίζει, εκτός των άλλων, και την ενέργειά του (ΠΑΡΑΡΤΗΜΑ) . Εισάγαμε λοιπόν μία υπορουτίνα που το υλοποιεί. Ως ενέργεια του περιπάτου ορίζουμε τον αριθμό των σημείων που έχουν πλησιέστερη γειτνίαση με άλλα που δεν είναι διαδοχικά τους. Η εκτιμήτρια που χρησιμοποιήσαμε για τον υπολογισμό ήταν η σχέση (2.1.1) για $Q = R^2$.

Δηλαδή:

$$R_M^2 = \frac{\sum_{i=1}^{M} R_{\mu_i}^2 e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{M} e^{-\beta E_{\mu_i}}}$$

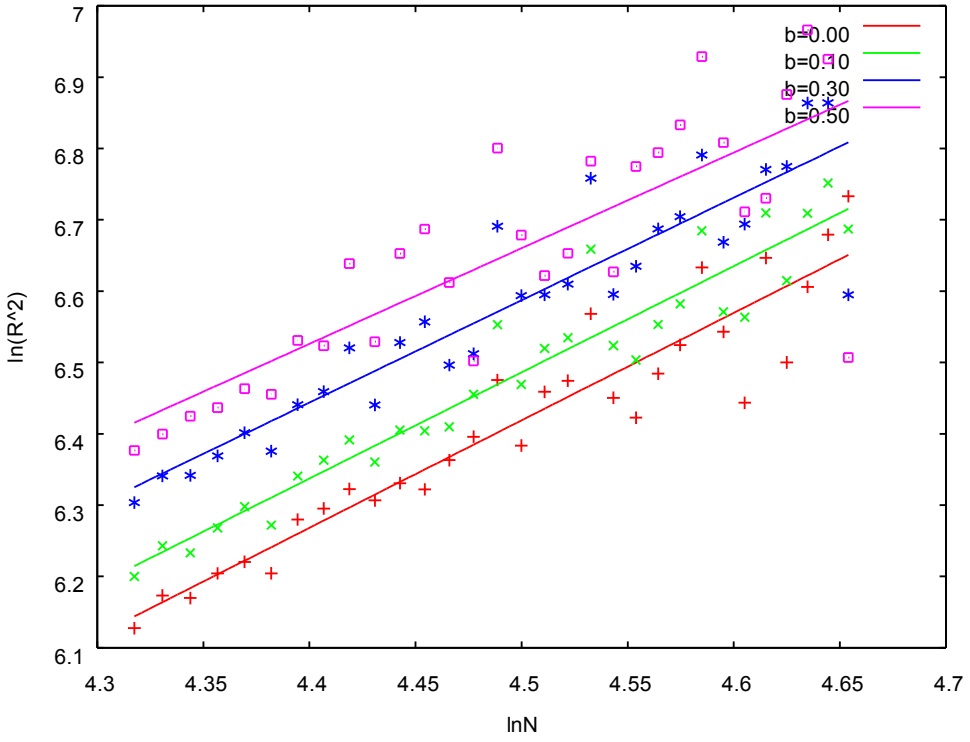
Πίνακας 4.1

N	R^2 (β=0.00)	R^2 (β=0.10)	R^2 (β= 0.30)	R^2 (β= 0.50)
75	458.22	492.67	546.45	587.92
76	479.66	514.36	567.29	601.51
77	478.02	509.26	567.78	616.91
78	494.77	527.35	583.42	624.36
79	502.86	543.38	602.79	641.01
80	494.86	529.46	587.30	636.17
81	533.64	567.16	626.84	686.07
82	541.95	579.90	638.51	680.97
83	556.93	596.75	678.87	763.94
84	548.15	578.32	626.63	684.77
85	561.48	605.06	683.88	774.87
86	556.65	604.20	704.08	801.97
87	579.95	607.66	662.68	744.12
88	599.29	636.14	673.26	666.55
89	648.96	701.30	805.10	898.41
90	592.00	644.82	730.47	795.24
91	638.29	678.25	731.29	751.33
92	648.25	688.58	742.36	775.11
93	712.12	779.66	861.32	881.96
94	632.94	681.05	731.68	755.32
95	615.54	667.39	761.24	875.63
96	654.74	701.44	802.10	892.67
97	681.61	722.06	816.04	927.99
98	759.80	800.10	889.69	1,021.37
99	694.35	713.95	787.39	905.28
100	628.56	708.77	807.51	821.63
101	770.16	820.21	871.74	837.46
102	665.17	746.08	875.43	968.70
103	739.52	819.83	957.01	1,060.21
104	795.61	855.46	957.24	1,017.69
105	839.71	801.97	731.34	669.78
2ν =	*1.50 ± 0.06*	*1.49 ± 0.08*	*1.44 ± 0.11*	*1.34 ± 0.18*

Στον πίνακα 4.1 φαίνονται οι μετρήσεις (β=0) και εκτιμήσεις (β≠0) του R^2 και στο σχήμα 4.5 η γραμμική προσαρμογή για κάθε θερμοκρασία.

Εδώ αξίζει να αναφέρουμε κάποιες λεπτομέρειες που θα μας βοηθήσουν να καταλάβουμε την αποδοτικότητα της μεθόδου. Όπως είπαμε, δημιουργήσαμε 100,000,000 (εκατό εκατομμύρια !!!) περιπάτους. Μόνο 500 από αυτούς κατάφεραν να αποκτήσουν μήκος μεγαλύτερο των 105 βημάτων και 18 μεγαλύτερο των 130 βημάτων. Το μεγαλύτερο μήκος είχε ένας και μοναδικός που κατάφερε να φτάσει τα 159 βήματα. Η προσομοίωση έγινε με επεξεργαστή τεχνολογίας Intel Core 2 Duo

E6400 2.13Ghz με 2MB cache χρησιμοποιώντας τον ένα μόνο πυρήνα και τη συγκεκριμένη υλοποίηση χρειάστηκαν 122m 50s.



Σχήμα 4.5

Η απλή δειγματοληψία δουλεύει για ένα μικρό εύρος μηκών μόνο, και αυτό με μεγάλο υπολογιστικό κόστος. Επιπλέον, ο όγκος των άχρηστων δεδομένων που πρέπει να παράξουμε και που αναφέρονται σε μικρά μήκη περιπάτων που δεν έχουν την αναμενόμενη βάθμωση, είναι κατά πολύ μεγαλύτερος από τα ωφέλημα στα οποία παρατηρείται η αναμενόμενη βάθμωση. Αλλά ακόμη και αυτό συμβαίνει σε ένα περιορισμένο εύρος μηκών (περίπου 75 – 100) αφού για μεγαλύτερα έχουμε το πρόβλημα του θορύβου. Τελικά κρίνουμε πως η απλή δειγματοληψία δεν είναι αποδοτική για $\beta>0.5$ και περιγράφει μόνο τη φάση υψηλής θερμοκρασίας. Αναμένουμε $2\nu = 1$ στη μετάβαση φάσης και $2\nu = 2/3$ στη φάση χαμηλής θερμοκρασίας οι διατάξεις των οποίων δεν εμφανίζονται στο δείγμα μας ώστε να πάρουμε μέτρηση που να έχει νόημα. Για όλους τους παραπάνω λόγους δεν έχει ιδιαίτερη χρησιμότητα σε πρακτικές εφαρμογές αφού ακόμη και στις υψηλές θερμοκρασίες τα μήκη των αλυσίδων που σχηματίζουν τα πολυμερή είναι πολύ

μεγαλύτερα από αυτά που μπορούν να παραχθούν.

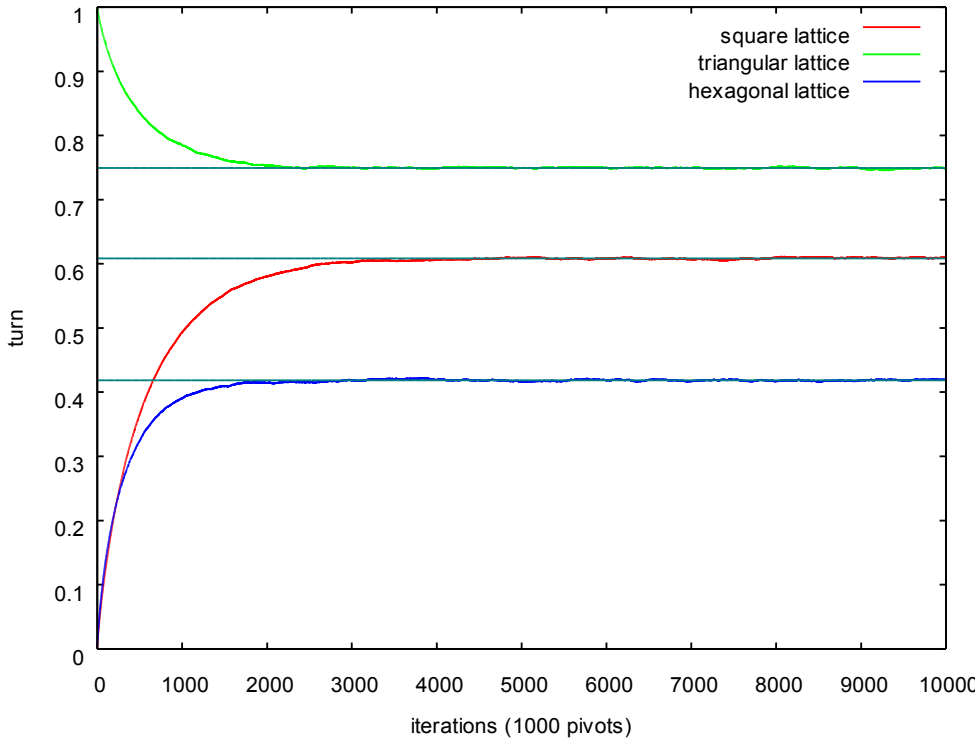
Συγκεντρωτκός πίνακας αποτελεσμάτων

β=1/KT	2ν	λ	μ
0.00	1.50 ± 0.06	0.129 ± 0.002	2.637 ± 0.0005
0.10	1.49 ± 0.08		
0.30	1.44 ± 0.11		
0.50	1.34 ± 0.18		

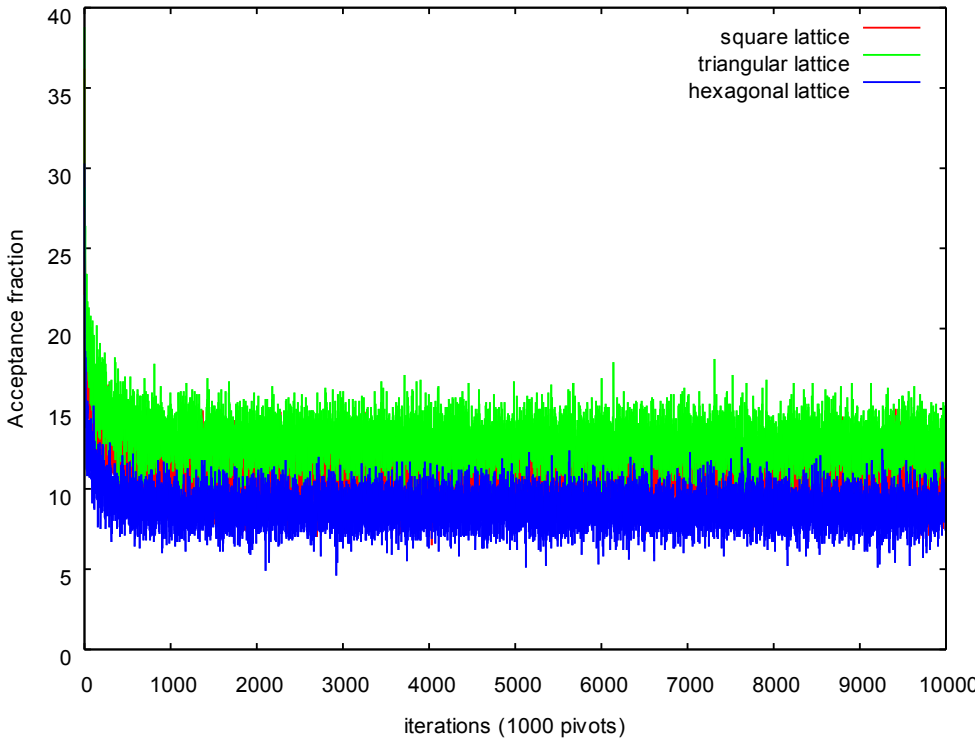
4.2 Δειγματοληψία με κριτήριο σημαντικότητας

Οι προσομοιώσεις μας για τον SAW έγιναν στις δύο διαστάσεις, χρησιμοποιώντας τον αλγόριθμο pivot έτσι όπως έχει υλοποιηθεί από τον Tom Kennedy. Ο αλγόριθμος αποτελείται από δύο κύρια loops, το ένα μέσα στο άλλο. Το outer loop ή αλλιώς niter όπως είναι η παράμετρος μέσα στο κώδικα, είναι ο αριθμός των μετρήσεων που γίνονται κάθε φορά που ολοκληρώνεται ένα inner loop. Το inner_loop είναι ο βρόχος στον οποίο εφαρμόζονται ο αριθμός των pivots που έχουμε ορίσει. Δηλαδή τα συνολικά pivots που λαμβάνουν χώρα είναι niter × inner_loop. Η επιλογή μας ήταν niter = 1000 και inner_loop = 1000. Εφαρμόσαμε τον αλγόριθμο για μήκη από 1,000 έως 100,000 και για τρία διαφορετικά πλέγματα: Το τετράγωνο, το τριγωνικό και το εξαγωνικό. Τα μεγέθη που μετράμε είναι το τετράγωνο της απόστασης από άκρη σε άκρη της αλυσίδας του πολυμερούς (R^2), το ποσοστό των διαδοχικών σημείων του περιπάτου που στρίβουν αντί να συνεχίζουν ευθεία (*turn*) και ο λόγος των pivots που γίνονται αποδεκτά προς αυτά που προτείνονται στο σύνολο (*Acceptance Ratio*). Σημειώστε πως το turn ορίζεται διαφορετικά σε κάθε πλέγμα. Θα υπολογίσουμε τις τιμές των κρίσιμων εκθετών και θα επιβεβαιώσουμε την παγκοσμιότητα όπου υπάρχει.

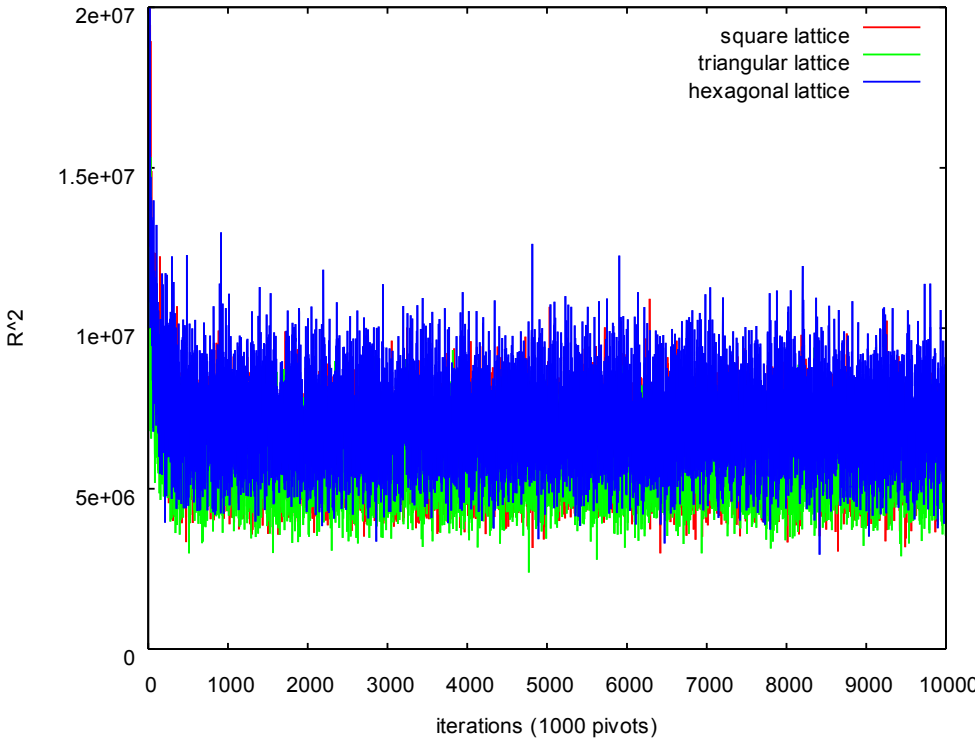
Οι αρχικοί μας περίπατοι δεν είναι τυπικές καταστάσεις περιπάτων και γι' αυτό θα πρέπει κατ' αρχήν να εξετάσουμε τη διαδικασία εξισορρόπησης (thermalization) ώστε να αποφανθούμε για το πότε έρχεται. Για το λόγο αυτό θα κάνουμε τα διαγράμματα των μεγεθών που μετράμε για το μεγαλύτερο μήκος περιπάτου που προσομοιάζουμε (N = 100,000) σαν συνάρτηση του Μόντε Κάρλο χρόνου.



Σχήμα 4.6: Theramalization του turn. Το turn περιγράφει το βαθμό της τοπικής αναδίπλωσης ("τσαλάκωμα") του περιπάτου.



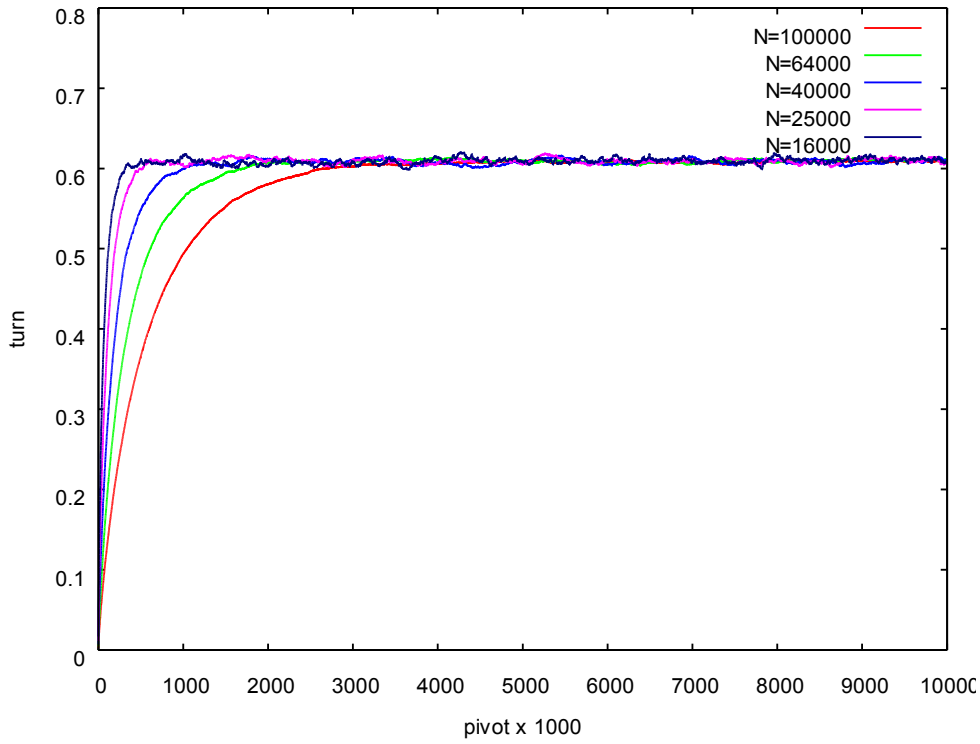
Σχήμα 4.7: Thermalization του λόγου αποδοχής των pivots



Σχήμα 4.8: Thermalization της απόστασης των άκρων

Παρατηρούμε ότι η κατάσταση ισορροπίας για το κάθε μέγεθος έρχεται μετά από διαφορετικό Μόντε Κάρλο χρόνο. Η εκτίμησή μας όμως θα βασιστεί στο turn όπου είναι το μέγεθος που χρειάζεται τον περισσότερο και είναι περίπου 5000 iterations. Πιο συγκεκριμένα παρατηρούμε πως στο τριγωνικό και στο εξαγωνικό πλέγμα, η ισορροπία έρχεται μετά από περίπου 2,500,000 με 3,000,000 pivots, ενώ στο τετράγωνο αρκετά αργότερα, γύρω στα 4,500,000 με 5,000,000. Επίσης παρατηρούμε ότι στο τετράγωνο και στο εξαγωνικό πλέγμα το turn ξεκινάει από το μηδέν. Αντίθετα, στο τριγωνικό ξεκινάει από το ένα που σημαίνει πως ο αρχικός περίπατος δεν έχει κανένα ζευγάρι από διαδοχικά steps που να είναι συγγραμικά.

Στο σχήμα 4.9 φαίνεται το thermalization για διάφορα μήκη περιπάτων στο τετράγωνο πλέγμα έτσι ώστε να έχουμε μια εποπτεία το χρόνο που απαιτείται ώστε να έλθει ένας περίπατος σε ισορροπία σαν συνάρτηση του μήκους του.



Σχήμα 4.9: Thermalization για διάφορα μήκη N στο τετράγωνο πλέγμα

Στους παρακάτω πίνακες παρουσιάζουμε τα αποτελέσματα των προσομοιώσεων που βασίζονται σε 5,000 επαναλήψεις του κώδικα μετά το (thermalization).

ΜΕΤΡΗΣΕΙΣ

Πίνακας 4.2 : Απλό Τετράγωνο Πλέγμα

N	R^2	turn	Acceptance Ratio (%)
1,000	24,332(38)	0.61012(36)	25.353(15)
1,600	4,937(12) x 10	0.60887(52)	23.170(32)
2,500	9,638(19) x 10	0.60824(74)	21.229(32)
4,000	19,542(47) x 10	0.60901(38)	19.390(29)
6,400	39,330(88) x 10	0.61001(65)	17.674(21)
10,000	7,717(31) x 10^2	0.60979(44)	16.264(26)
16,000	15,581(68) x 10^2	0.61018(35)	14.846(30)
25,000	3,055(15) x 10^3	0.60966(64)	13.619(32)
40,000	6,179(21) x 10^3	0.60912(60)	12.481(27)
64,000	12,479(38) x 10^3	0.60919(55)	11.395(22)
100,000	24,463(76) x 10^3	0.60868(35)	10.512(20)

Πίνακας 4.3 Τριγωνικό πλέγμα

N	R^2	turn	Acceptance ration (%)
1,000	22,525(39)	0.74924(48)	27.991(22)
1,600	45,628(84)	0.74862(41)	25.730(34)
2,500	8,910(21) x 10	0.74882(65)	23.845(32)
4,000	18,022(35) x 10	0.74950(61)	21.977(15)
6,400	36,526(98) x 10	0.74970(79)	20.282(29)
10,000	7,103(16) x 10^2	0.74894(62)	18.746(27)
16,000	14,399(25) x 10^2	0.74877(49)	17.327(23)
25,000	28,282(78) x 10^2	0.74899(42)	16.121(19)
40,000	5,682(17) x 10^3	0.74894(53)	14.896(20)
64,000	11,523(35) x 10^3	0.74935(52)	13.787(33)
100,000	22,408(55) x 10^3	0.74942(30)	12.811(25)

Πίνακας 4.4 Εξαγωνικό πλέμα

N	R^2	turn	Acceptace ratio (%)
1,000	28,078(70)	0.41884(24)	23.059(45)
1,600	5,669(11) x 10	0.41848(47)	20.848(25)
2,500	11,108(44) x 10	0.41880(49)	18.982(34)
4,000	22,375(73) x 10	0.41880(33)	17.152(19)
6,400	4,525(12) x 10^2	0.41923(31)	15.539(21)
10,000	8,870(31) x 10^2	0.41916(50)	14.213(17)
16,000	17,930(37) x 10^2	0.41900(46)	12.843(19)
25,000	34,851(88) x 10^2	0.41969(28)	11.656(26)
40,000	7,066(31) x 10^3	0.41933(38)	10.563(17)
64,000	14,335(57) x 10^3	0.41875(49)	9.575(29)
100,000	27,875(95) x 10^3	0.41845(24)	8.689(15)

Κρίσιμος εκθέτης *ν*

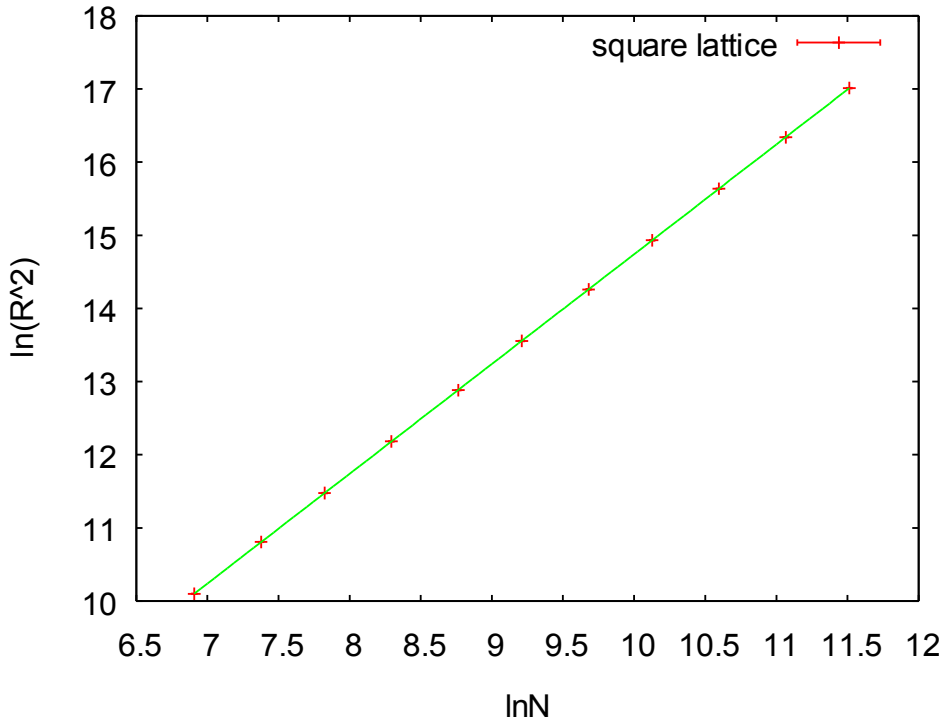
Όπως έχουμε ήδη αναφέρει υπάρχει ο κρίσιμος εκθέτης *ν* ο οποίος παρουσιάζει παγκοσμιότητα. Η σχέση που συνδέει το τετράγωνο της απόσταση με το μήκος του περιπάτου είναι η παρακάτω:

$$< R^2 > \; = \; AN^{2\nu}[1+O(N^{-\Delta})] \qquad N \to \infty$$

επομένως $$< R^2 > \; \sim \; AN^{2\nu} \qquad N \to \infty$$

και η τιμή του ν που προσδιορίζεται στη βιβλιογραφία είναι **3/4**. Σημειώνουμε πάλι ότι η σταθερά A δεν είναι παγκόσμια αλλά εξαρτάται από διάφορους παράγοντες όπως τη θερμοκρασία, το πλέγμα στο οποίο κινείται το πολυμερές και άλλα, και η εξάρτησή του από τα προηγούμενα αναμένεται να είναι ιδιαίτερα πολύπλοκη. Επομένως κάνοντας γραμμική προσαρμογή του τετραγώνου της απόστασης με το μήκος του περιπάτου, μπορούμε να υπολογίσουμε το ν από τη κλίση της ευθείας. Η εξίσωσή της θα είναι:

$$lnR^2 \; = \; lnA + 2\,\nu\, lnN$$

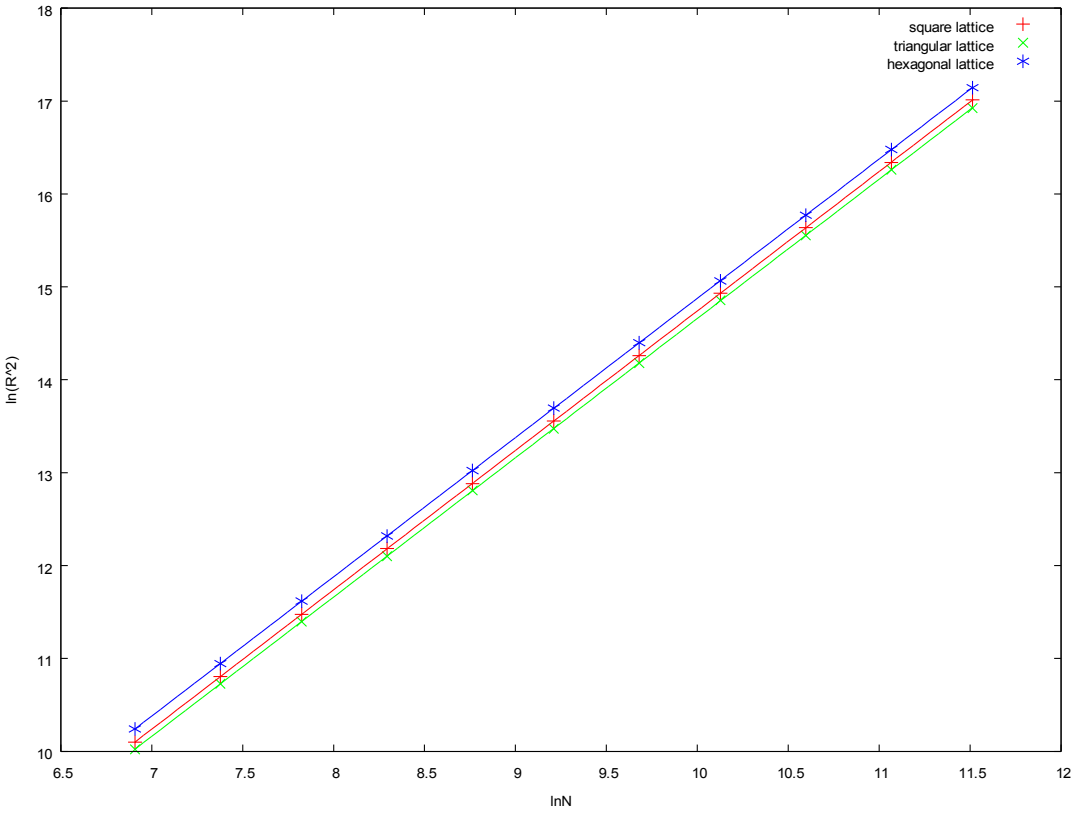


Σχήμα 4.10: Προσαρμογή δεδομένων για να εύρεση του κρίσιμου εκθέτη ν

Η κλίση της ευθείας για το *τετράγωνο* πλέγμα βρέθηκε $2\nu = 1.5007 \pm 0.0004$.

Ομοίως κάνουμε και για τον περιπατητή που κινείται στο τριγωνικό και εξαγωνικό πλέγμα. Για το *τριγωνικό* βρήκαμε $2\nu = 1.4992 \pm 0.0005$ και για το *εξαγωνικό* $2\nu = 1.4988 \pm 0.0004$.

Παρόλο που το είδος του πλέγματος έχει επηρεάσει τις αριθμητικές μας μετρήσεις (λόγω διαφορετικής πλεγματικής σταθεράς) όπως φαίνεται στους πίνακες των αποτελεσμάτων, η βάθμωση του τετραγώνου της απόστασης με το μήκος δεν έχει επηρεαστεί. Άρα μπορούμε να πούμε πως επιβεβαιώνεται και η **παγκοσμιότητα** του συγκεκριμένου κρίσιμου εκθέτη (σχήμα 4.11).



Σχήμα 4.11: Γραμμική προσαρμογή του τετραγώνου της απόστασης με το μήκος και για τα τρία πλέγματα.

Παρατηρούμε επίσης ότι οι τρεις ευθείες είναι παράλληλες, κάτι που έτσι κι αλλιώς το περιμέναμε λόγω της παγκοσμιότητας, και πως είναι έχουν σχετική μετατόπιση στον κατακόρυφο άξονα. Αυτό συμβαίνει διότι το σημείο τομής τους με το άξονα $x = 0$ είναι το lnA, και το A δεν είναι παγκόσμιο αλλά επηρεάζεται από επιμέρους λεπτομέρειες της προσομοίωσης, στη προκειμένη περίπτωση από το πλέγμα πάνω

στο οποίο κινείται ο τυχαίος περιπατητής.

Για το A μετρήσαμε:

$A_{square} = 0.767 \pm 0.003$

$A_{triagular} = 0.717 \pm 0.003$

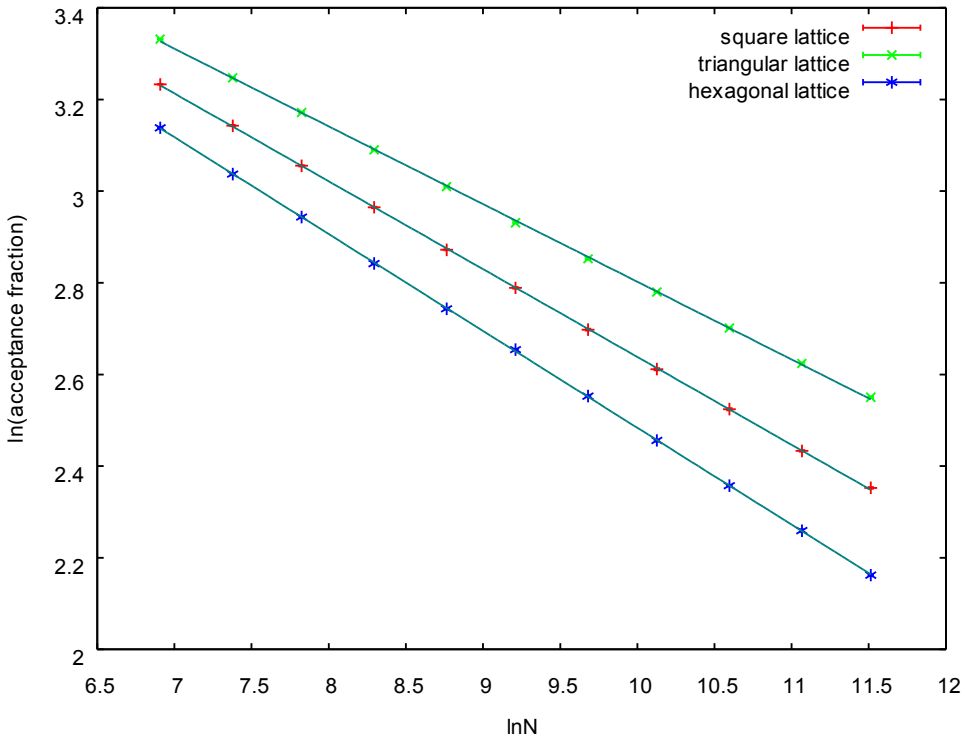
$A_{hexagonal} = 0.895 \pm 0.004$

Κρίσιμος εκθέτης p

Το acceptance fraction τείνει στο μηδέν καθώς το $N \to \infty$ ως N^{-p}. Δηλαδή,

$$f \sim N^{-p}$$

ομοίως λοιπόν για τον υπολογισμό του κρίσιμου εκθέτη p κάνουμε γραμμική προσαρμογή του acceptance fraction με το μήκος.



Σχήμα 4.12: Γραμμική προσαρμογή του acceptance fraction για τα τρία πλέγματα

όπου μετρήσαμε:

$p_{square} = 0.1917 \pm 0.0005$
$p_{triangular} = 0.1695 \pm 0.0007$
$p_{hexagonal} = 0.2114 \pm 0.0004$

Άρα επιβεβαιώνουμε ότι το *p* δεν είναι παγκόσμιο αφού φαίνεται πως επηρεάζεται από το πλέγμα. Αυτό όπως θα δούμε και στη συνέχεια είναι ένα κριτήριο για την επιλογή του πλέγματος αφού το *p* περιγράφει και την αυτοσυσχέτιση των περιπάτων που παράγονται από τις προσομοιώσεις.

Αυτοσυσχέτιση

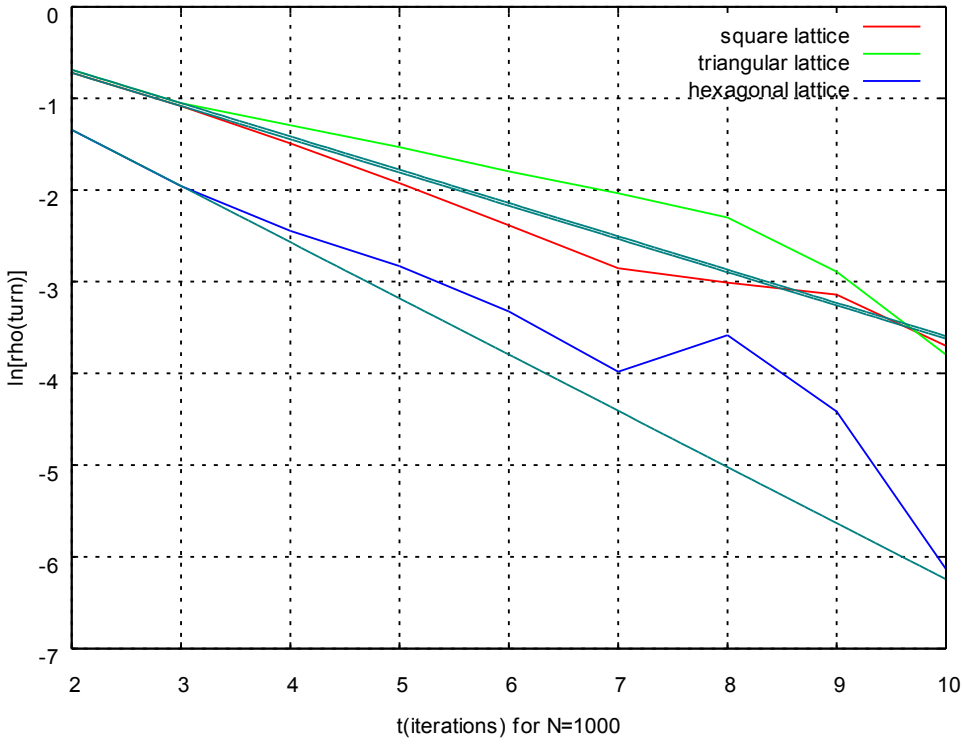
Ο χρόνος αυτοσυσχετισμού *τ* μεγαλώνει καθώς $N \rightarrow \infty$ ως

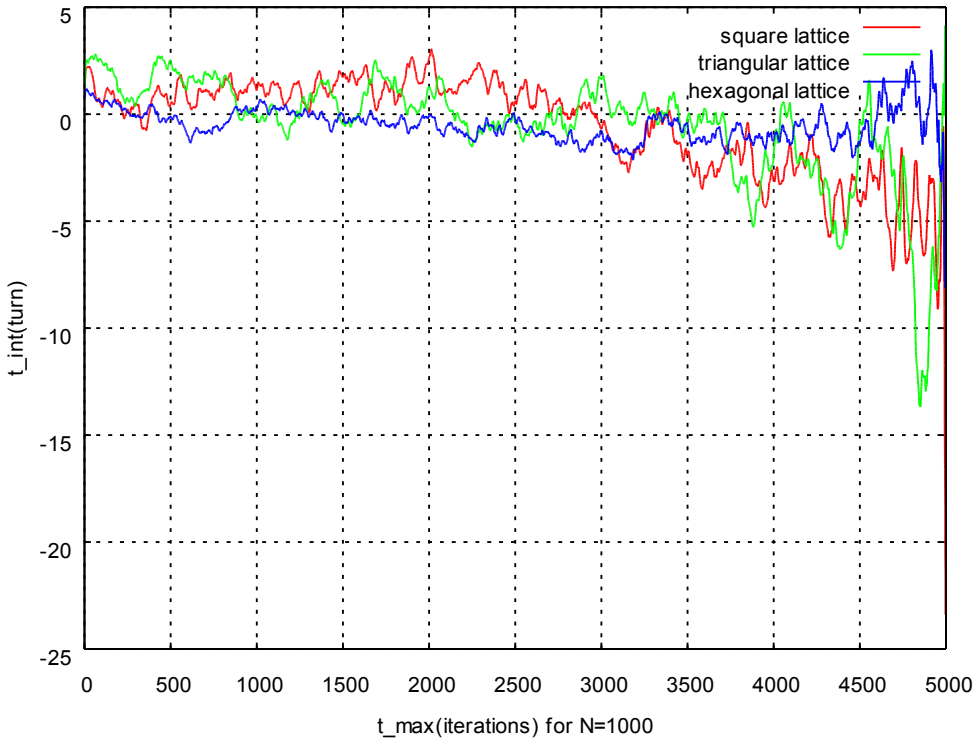
$$\tau \sim N^p$$

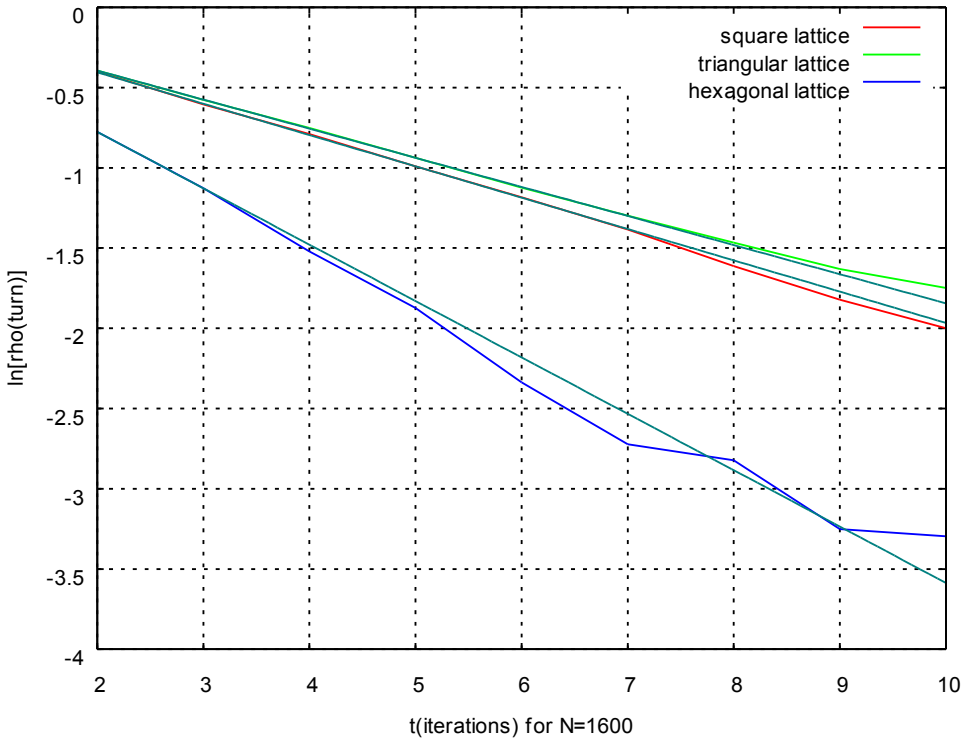
Η εκτίμηση του χρόνου αυτοσυσχετισμού είναι εν γένει μια δύσκολη διαδικασία διότι εκτός των άλλων γίνεται και γραφικά. Εμείς θα κάνουμε μία προσπάθεια να εκτιμήσουμε τους τ και τ_{int} ώστε να δούμε αν ακολουθούν την αναμενόμενη βάθμωση με το N.

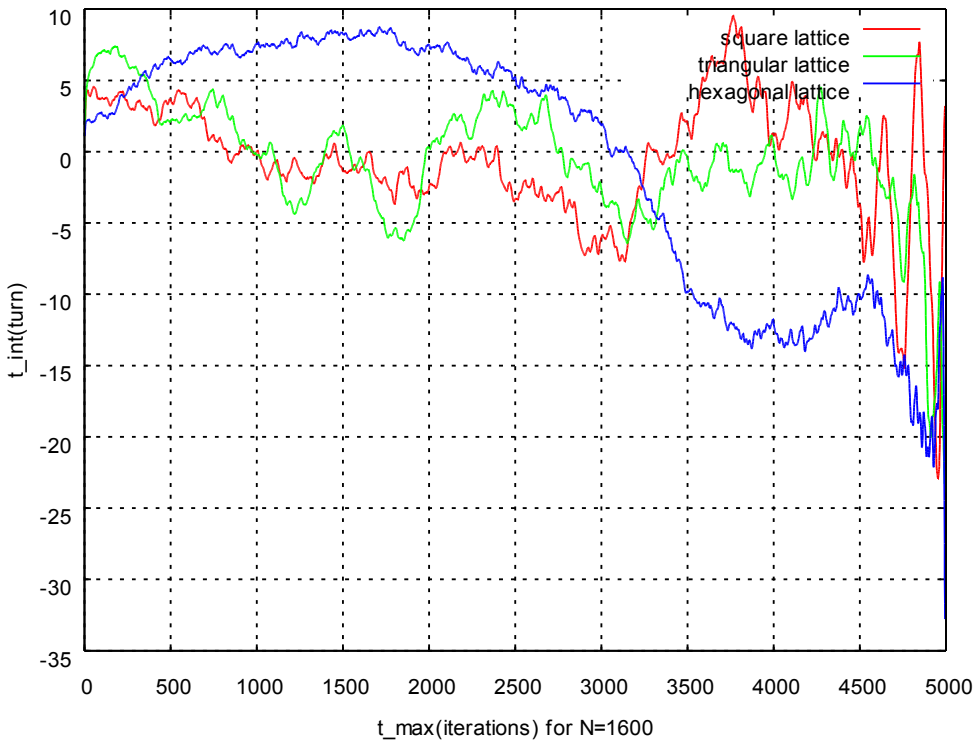
Παρακάτω παραθέτουμε τα διαγράμματα $ln(\rho_{turn})$– t(iterations) ώστε με γραμμική παρεμβολή στο εκθετικό μέρος να προσδιορίσουμε το τ (τ=-1/slope) και τα διαγράμματα τ_{int}(turn) – t(iterations) από τα οποία προσδιορίσουμε μια ελάχιστη τιμή για τον ολοκληρωμένο χρόνο αυτοσυσχετισμού.

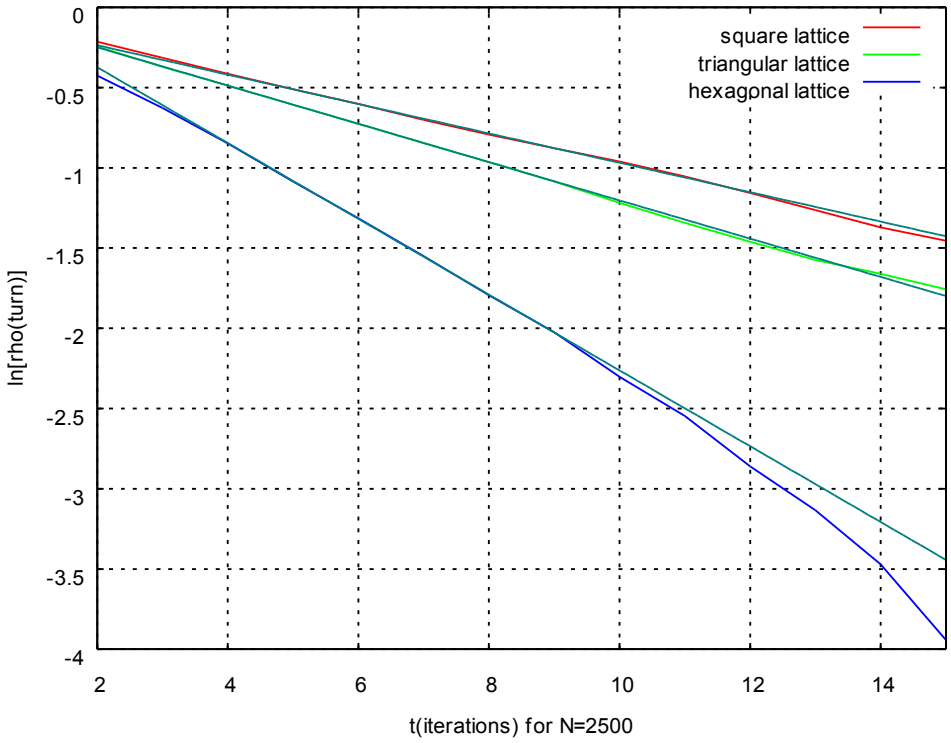
Στον πίνακα 4.5 φαίνονται οι εκτιμήσεις μας

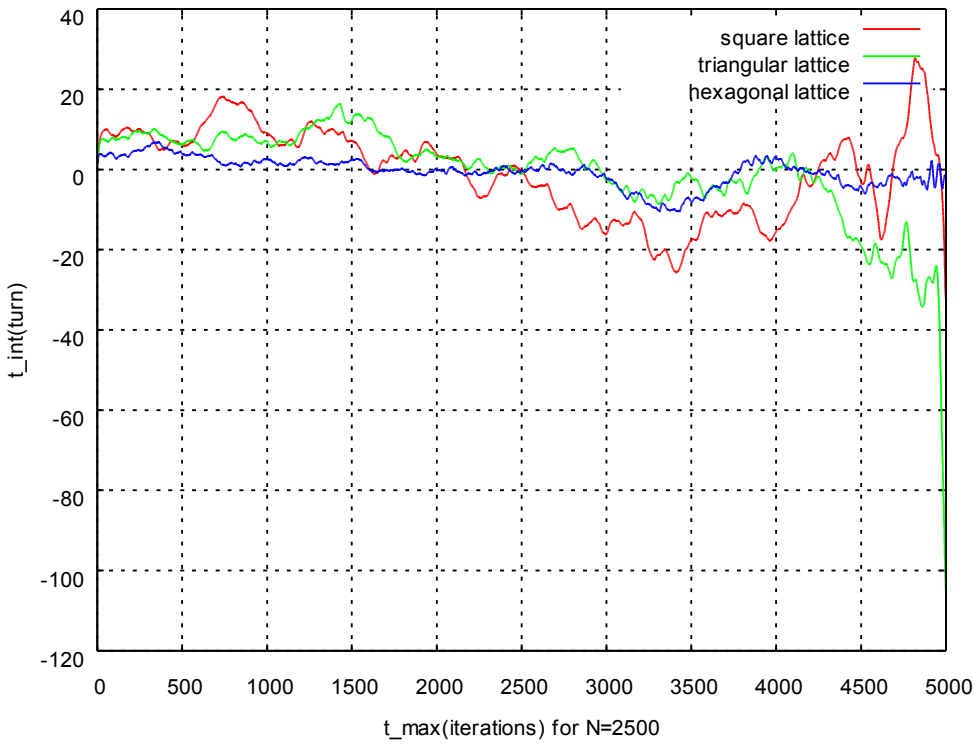

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=1000
0
-1
-2
-3
-4
-5
-6
-7
2
3
4
5
6
7
8
9
10

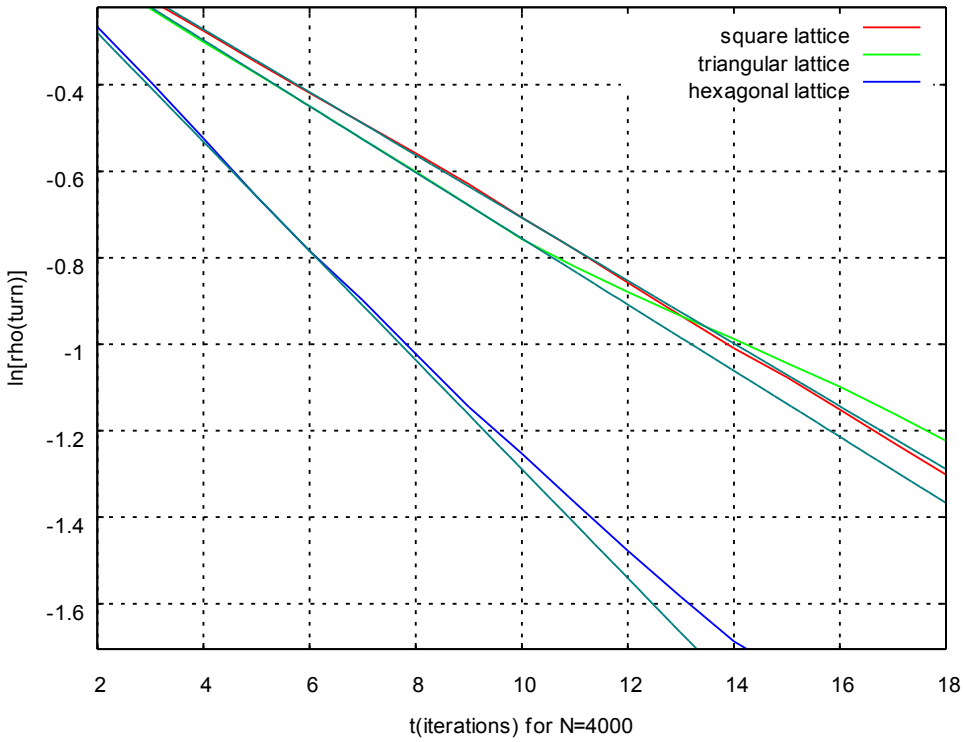

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=1000
5
0
-5
-10
-15
-20
-25
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

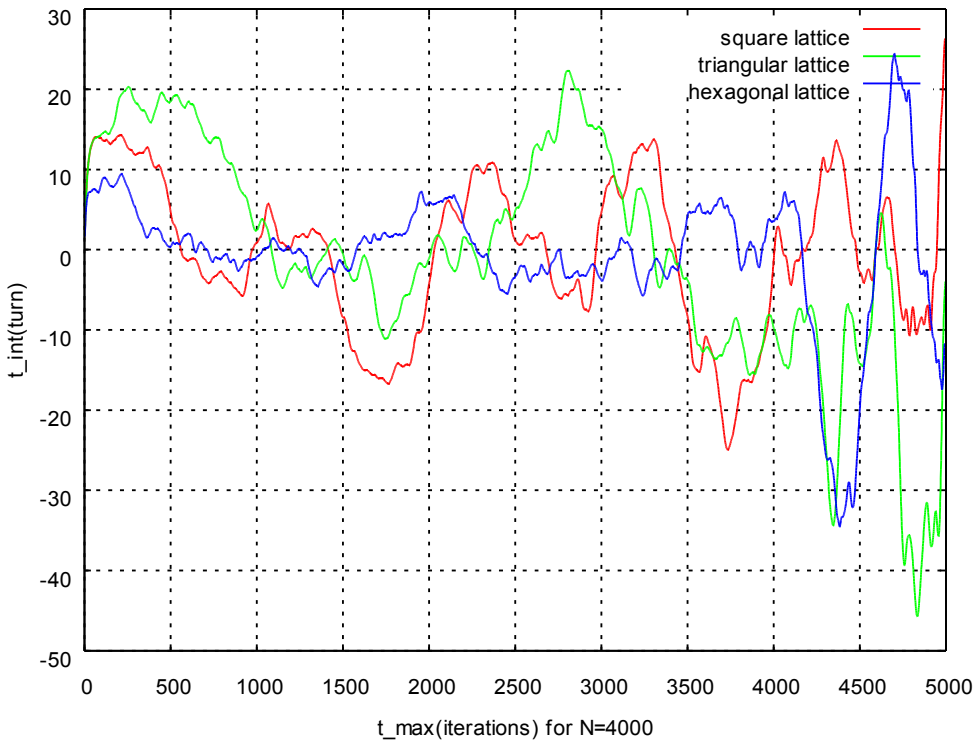

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=1600
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
2
3
4
5
6
7
8
9
10

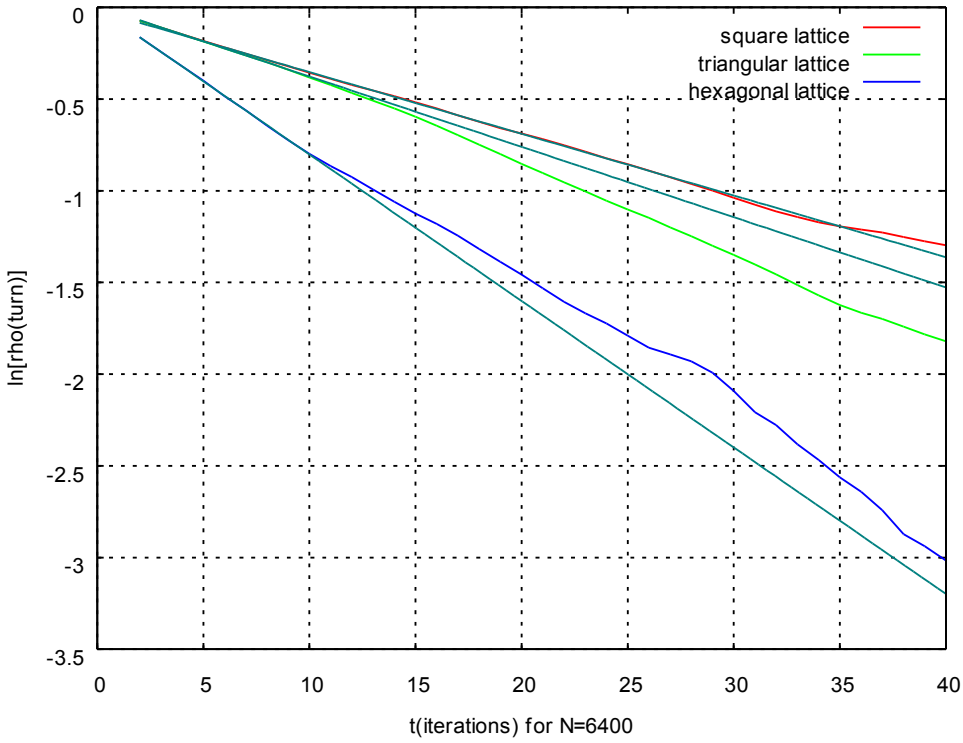

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=1600
10
5
0
-5
-10
-15
-20
-25
-30
-35
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

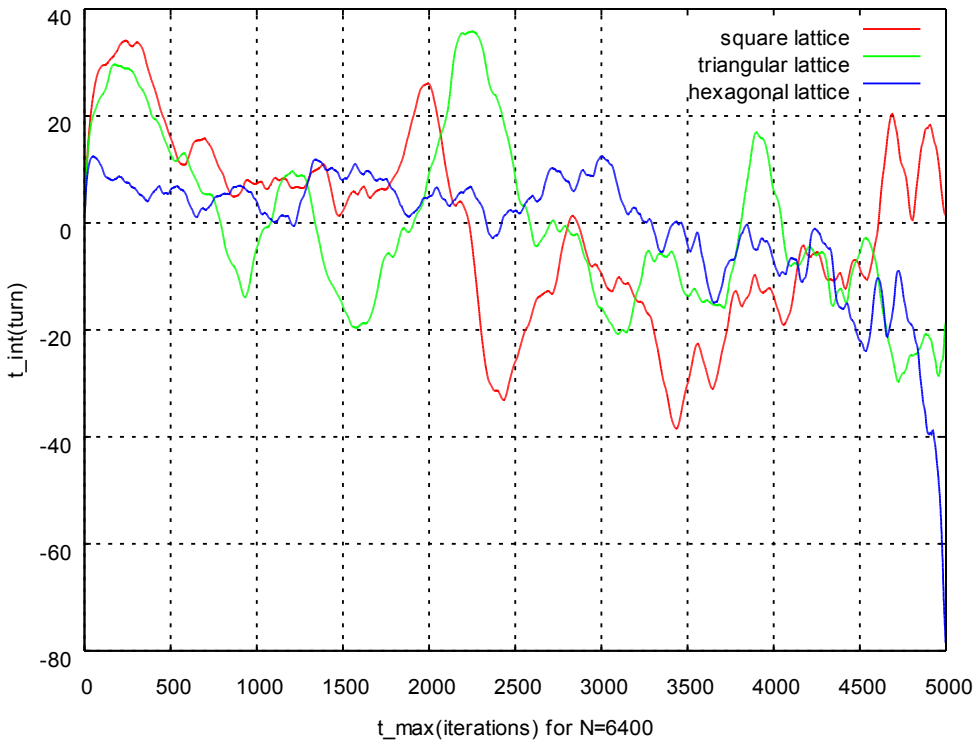

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=2500
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
2
4
6
8
10
12
14

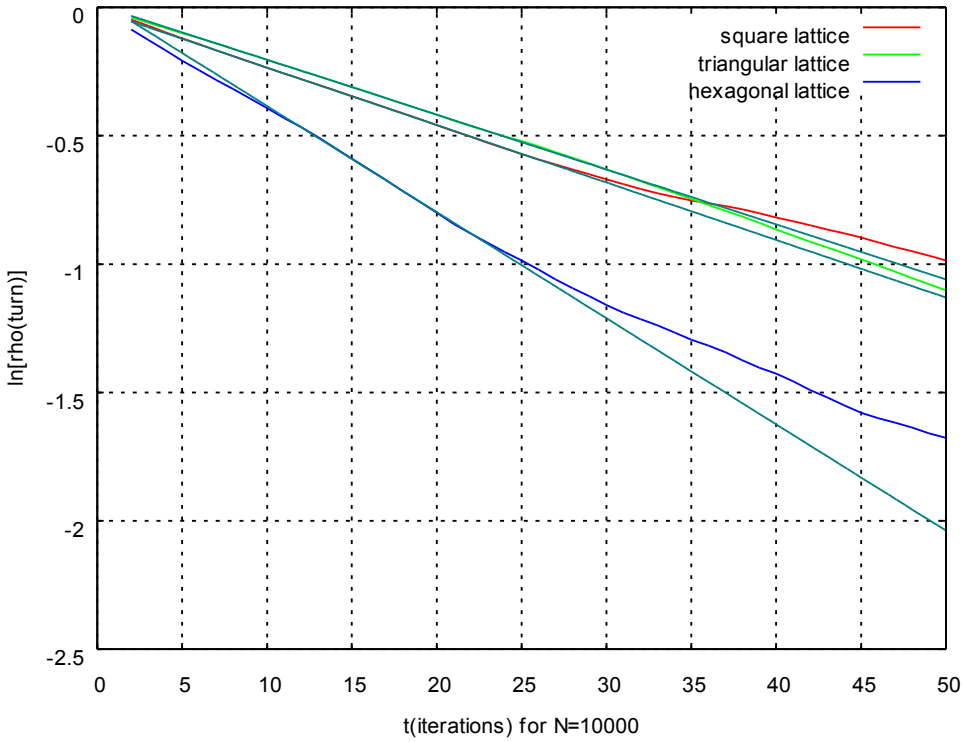

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=2500
40
20
0
-20
-40
-60
-80
-100
-120
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

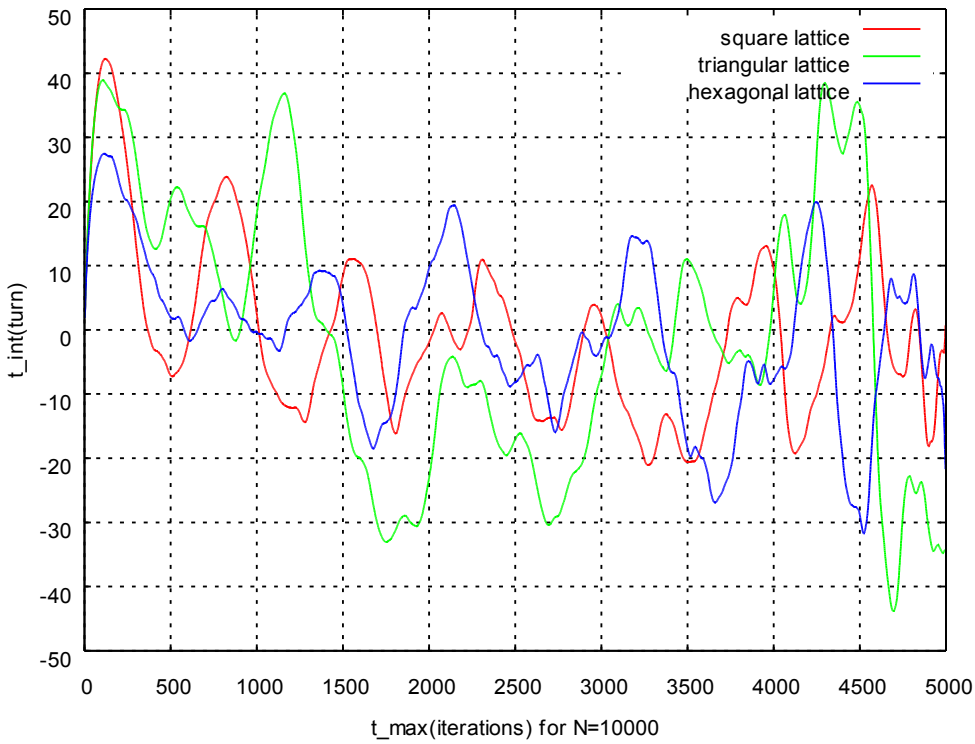

square lattice
triangular lattice
hexagonal lattice
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
ln[rho(turn)]
2
4
6
8
10
12
14
16
18
t(iterations) for N=4000

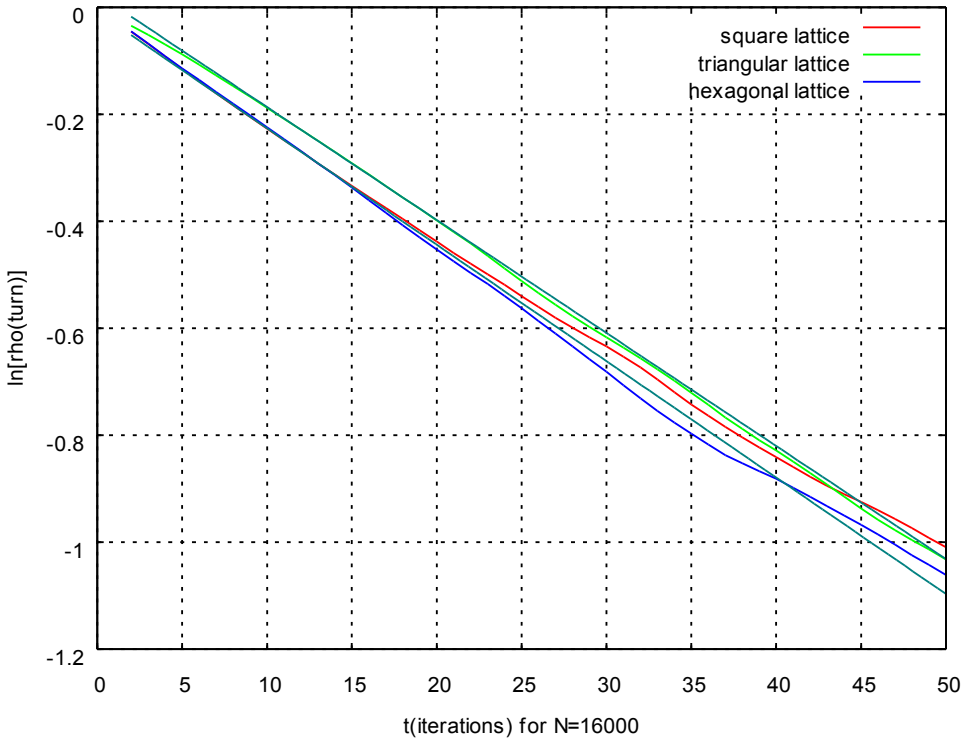

square lattice
triangular lattice
hexagonal lattice
30
20
10
0
-10
-20
-30
-40
-50
t_int(turn)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
t_max(iterations) for N=4000

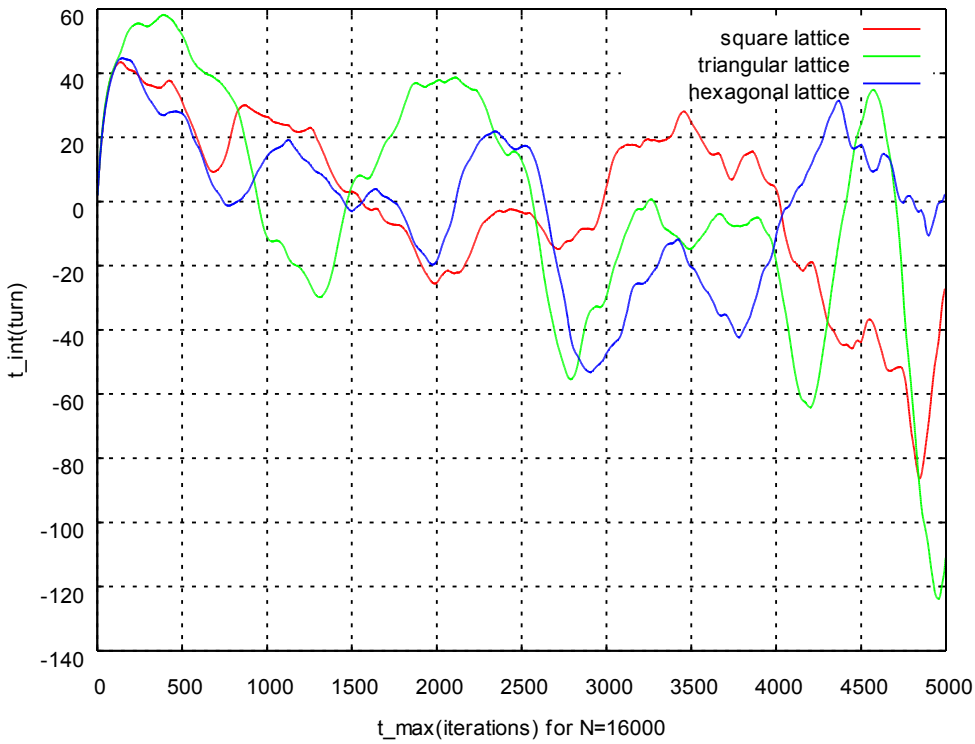

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=6400
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
5
10
15
20
25
30
35
40

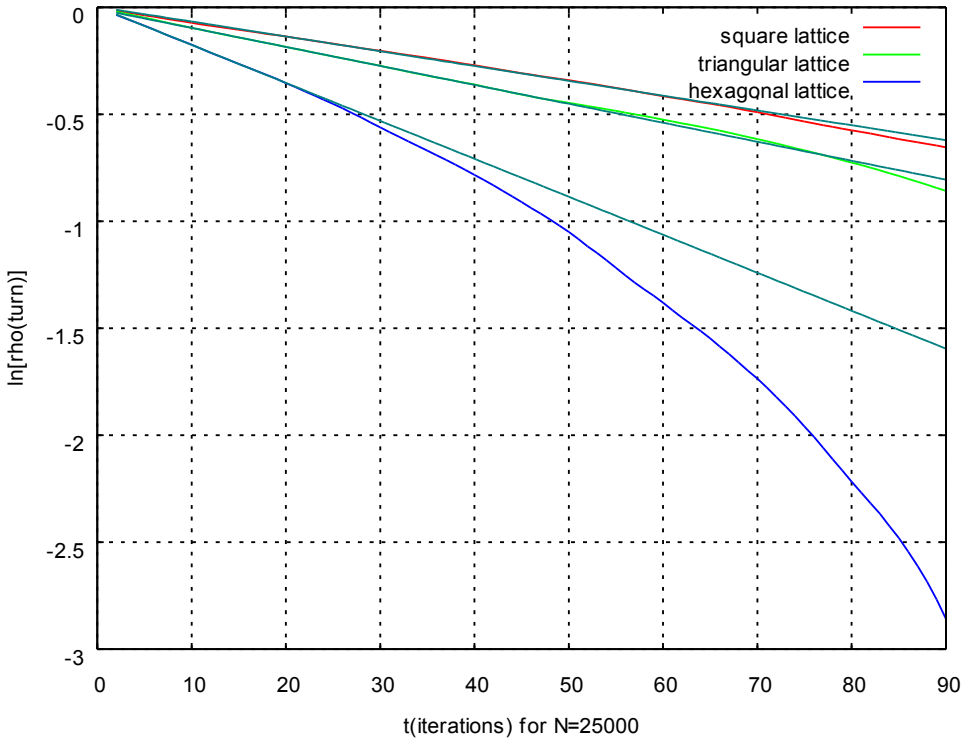

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=6400
40
20
0
-20
-40
-60
-80
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

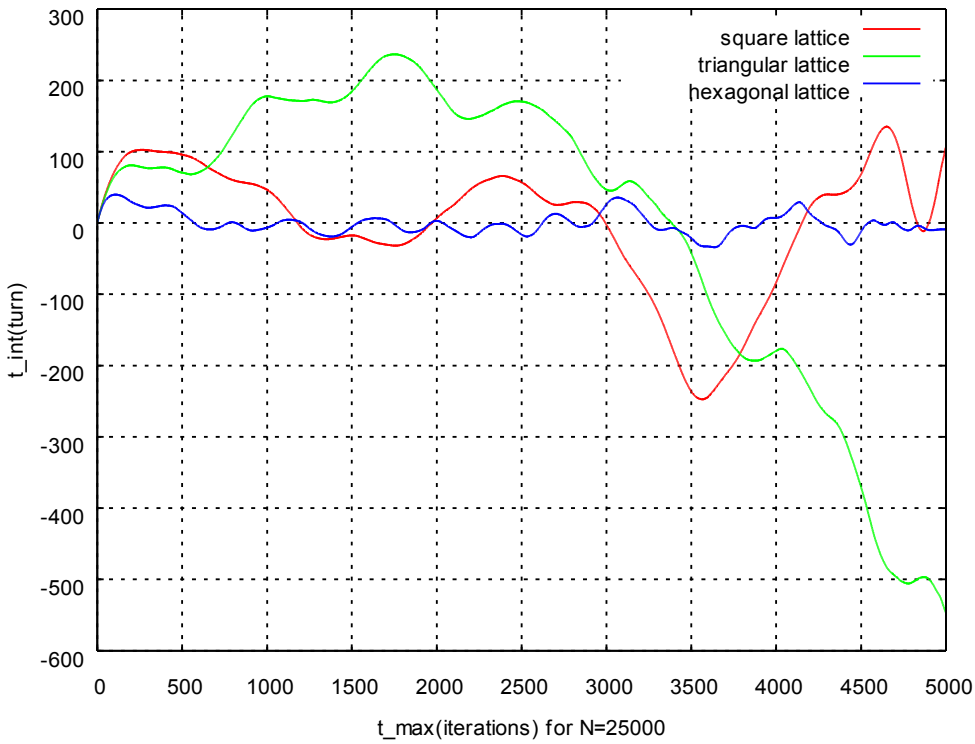

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=10000
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30
35
40
45
50

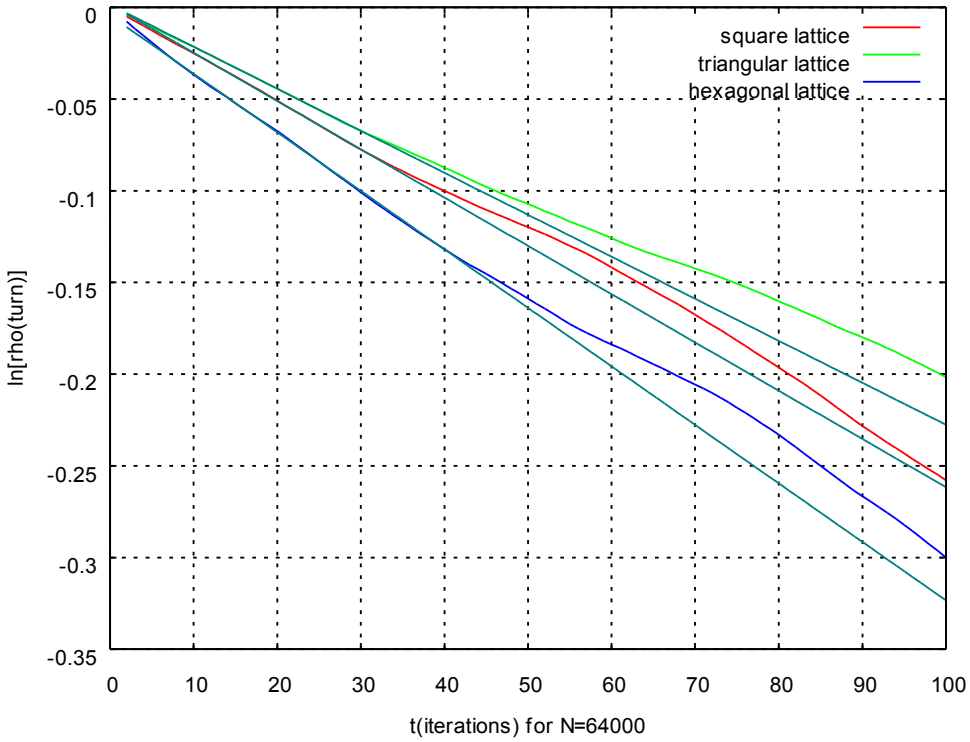

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=10000
50
40
30
20
10
0
-10
-20
-30
-40
-50
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

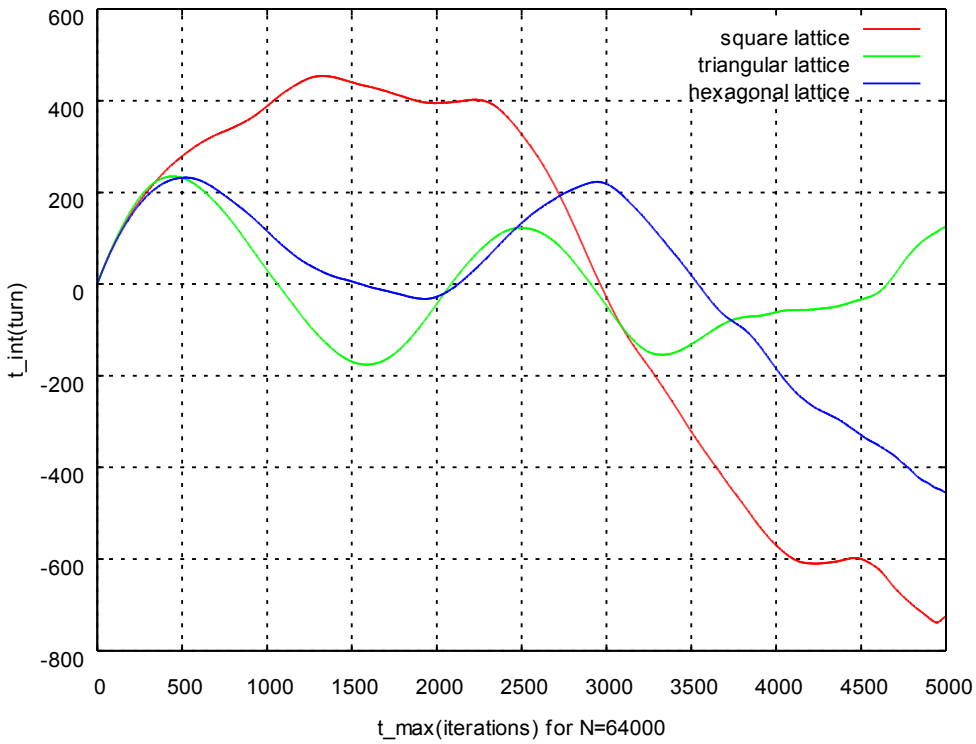

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=16000
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
5
10
15
20
25
30
35
40
45
50

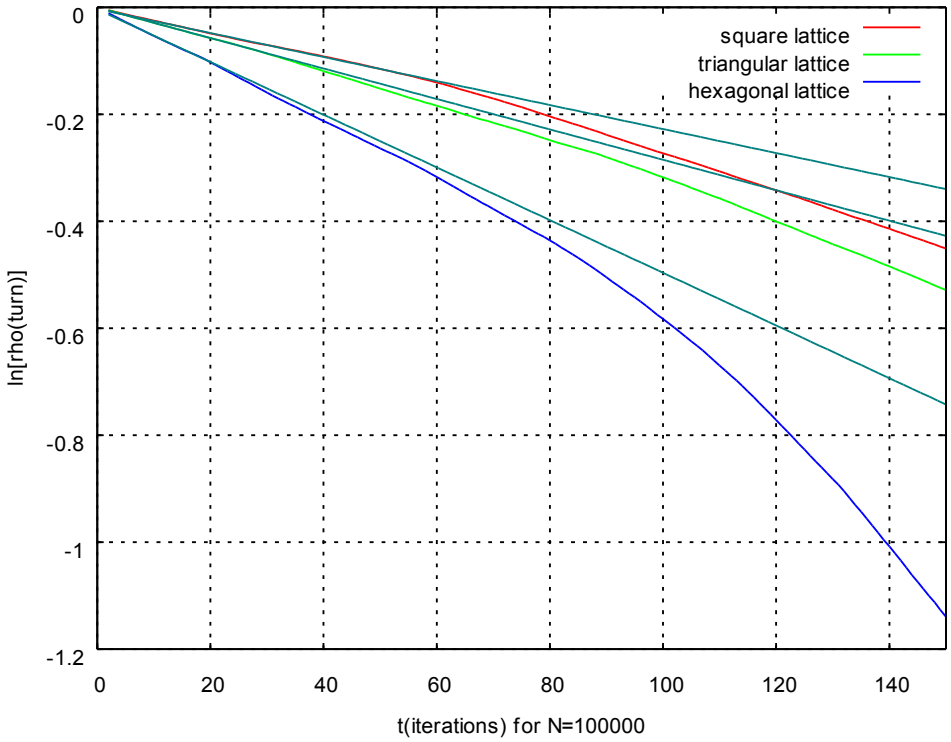

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=16000
60
40
20
0
-20
-40
-60
-80
-100
-120
-140
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

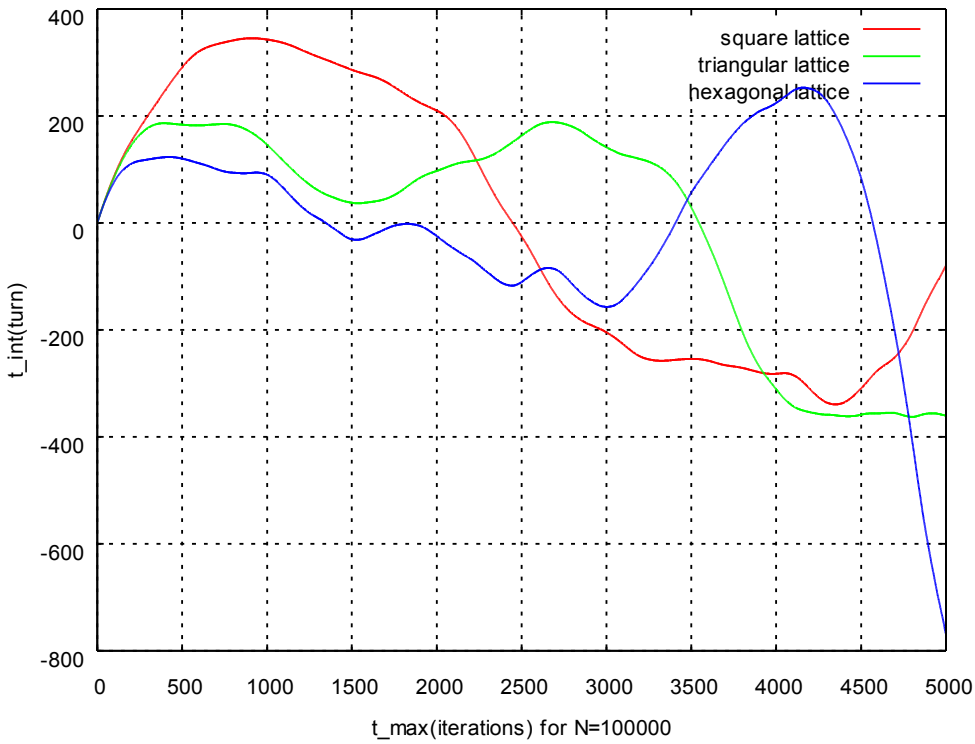

square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=25000
0
-0.5
-1
-1.5
-2
-2.5
-3
0
10
20
30
40
50
60
70
80
90


square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=25000
300
200
100
0
-100
-200
-300
-400
-500
-600
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=64000
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
0
10
20
30
40
50
60
70
80
90
100


square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=64000
600
400
200
0
-200
-400
-600
-800
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


square lattice
triangular lattice
hexagonal lattice
ln[rho(turn)]
t(iterations) for N=100000
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
20
40
60
80
100
120
140

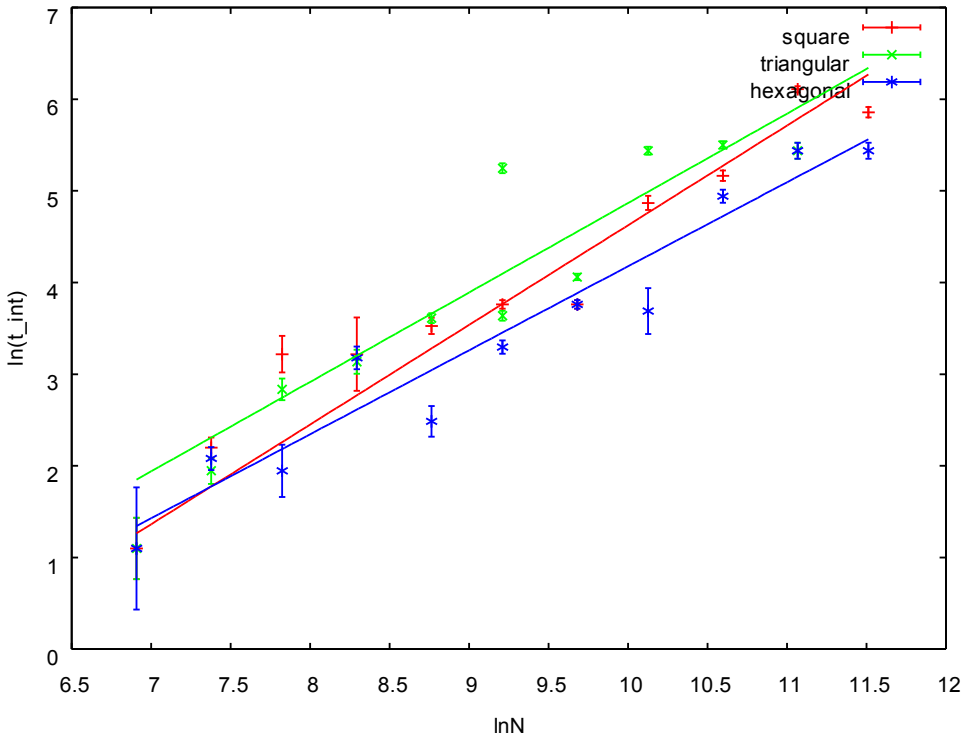

square lattice
triangular lattice
hexagonal lattice
t_int(turn)
t_max(iterations) for N=100000
400
200
0
-200
-400
-600
-800
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Οι εκτιμήσεις μας για τους χρόνους αυτοσυσχετισμού είναι:

Πίνακας 4.5

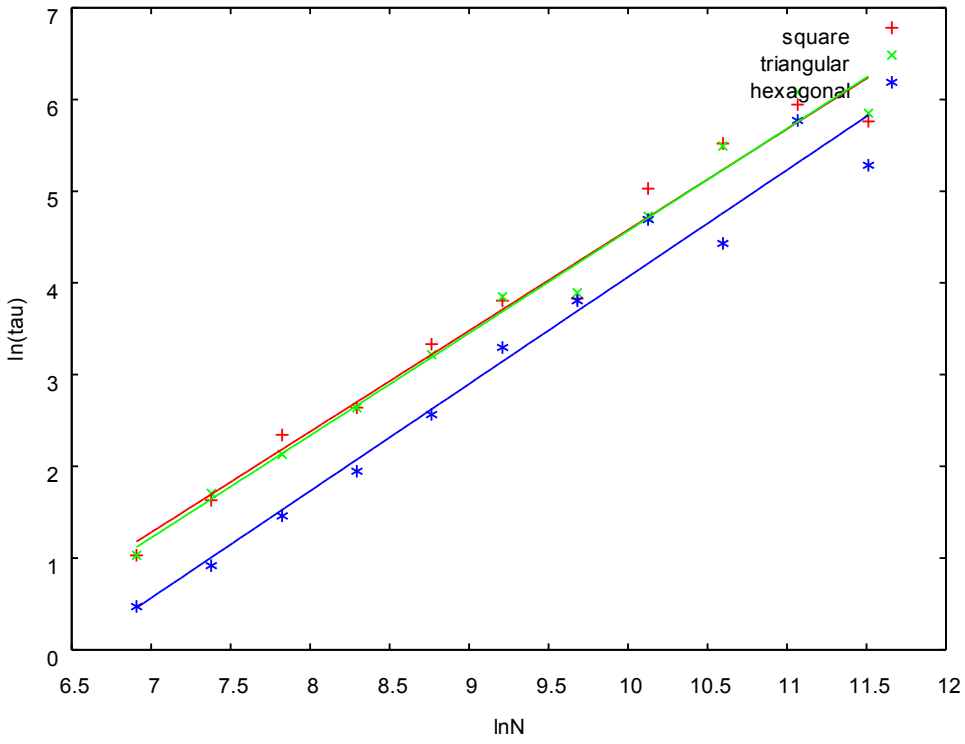
	square		triangular		hexagonal	
N	τ	τ_{int}	τ	τ_{int}	τ	τ_{int}
1,000	2.8	3(1)	2.8	3(1)	1.6	3(2)
1,600	5.1	9(1)	5.5	7(1)	2.5	8(1)
2,500	10.4	25(5)	8.4	17(2)	4.3	7(2)
4,000	14	25(10)	14	23(3)	7	24(3)
6,400	28	34(3)	25	37(2)	13	12(2)
10,000	45	43(2)	47	38(2)	27	27(2)
16,000	46	43(2)	49	58(2)	45	43(2)
25,000	153	130(10)	113	230(10)	109	40(10)
40,000	250	175(10)	243	245(10)	84	140(10)
64,000	382	450(15)	437	230(20)	321	230(20)
100,000	318	350(20)	347	190(10)	197	230(20)

Στο σχήμα 4.13 φαίνεται η γραμμική προσαρμογή για τον ολοκληρωμένο χρόνο αυτοσυσχετισμού τ_{int}.



Σχήμα 4.13

Στο παρακάτω σχήμα φαίνεται η γραμμική προσαρμογή για το χρόνο αυτοσυσχετισμού τ.



Σχήμα 4.14

Τελικά τα αποτελέσματά μας είναι:

Πίνακας 4.6

lattice	p = f(τ)	p = f(τint)
square	1.10(6)	1.09(2)
triangular	1.05(6)	0.98(2)
hexagonal	1.17(7)	0.92(3)

Παρατηρούμε ότι οι τιμές που υπολογίζουμε δεν είναι οι αναμενόμενες και ουσιαστικά έχουμε αποτύχει να επιβεβαιώσουμε την αναμενόμενη βάθμωση του χρόνου αυτοσυσχετισμού με το μήκος N. Ακόμη παρατηρούμε πως οι τιμές που υπολογίζουμε βρίσκονται όλες γύρω από το 1, ενώ οι τιμές που αναμένουμε βρίσκονται όλες γύρω από το 0.2 (και για τα τρία πλέγματα). Αυτό μας υποψιάζει για την ύπαρξη συστηματικών σφαλμάτων. Ο κυριότερος όμως λόγος για την αποτυχία

εκτιμάται πως είναι ο αριθμός των μετρήσεων. Ενδεχομένως με περισσότερες μετρήσεις και σε περισσότερα μήκη να είχαμε μια καλύτερη συμφωνία των αποτελεσμάτων μας και των αναμενόμενων τιμών

5. ΣΥΓΚΡΙΣΗ ΜΕΘΟΔΩΝ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Τελικά μελετώντας θεωρητικά τις δύο μεθόδους αλλά και εκτιμώντας τα αποτελέσματα που μας δίνουν καταλήγουμε στο συμπέρασμα ότι ο αλγόριθμος pivot είναι μακράν ένας καλύτερος, αποδοτικότερος και γρηγορότερος τρόπος για να προσομοιάσουμε το μοντέλο του αυτοαποφεύγοντα τυχαίου περιπατητή. Τα βασικότερα σημεία στα οποία υπερτερεί συντριπτικά έναντι της απλής δειγματοληψίας είναι τα μεγάλα μήκη περιπάτων που μπορεί να περιγράψει και που ουσιαστικά αποτελούν ρεαλιστικά μήκη πολυμερών, ο μικρός και εύκολα διαχειρίσιμος όγκος δεδομένων που παράγονται από τις προσομοιώσεις και τέλος η αξιοσημείωτη ταχύτητά του από την υλοποίηση του Tom Kennedy.

ΠΑΡΑΡΤΗΜΑ

Ο Αλγόριθμος για την μελέτη των self avoiding walks με τη μέθοδο της απλής δειγματοληψίας

```
/*********************  saw2.c ********************************/
/*
 Compile with:
 gcc -O2     saw2.c -o saw2
 icc -O3 -xK saw2.c -o saw2 (PIII - gcc is faster)
 icc -O3 -xW saw2.c -o saw2 (PIV)
 */

#include <libgen.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define  XNN 1
#define  YNN L
/*
   Nwalk: Number of random walks generated.
   N    : Length of random walks
   L    : Linear size of lattice and L2 = L*L
 */
int    N, L, L2;
int    *h; /* The frequency function array h[L2]: How many times each
              point is visited by the random walker */
long   Nwalk,iwalk; /*Random walk counter          */
int    R,i2,MA;
void   init(int, char**),end();
void   generate_a_walk(),measure();
double measA1(); /*Area and length functions       */
void   measA2l1(double *,double *),measl2l3(double *,double *);
double drandom();
```

```
void   init_random();
void   energy();


/*********************  main   () ****************************/
int main(int argc, char **argv){
  init(argc,argv); /*initialize program*/
  for(iwalk=1;iwalk<=Nwalk;iwalk++){
    generate_a_walk();
    if (i2>14) energy();
    if (i2>14) measure();
  }
  end();
} //main()


/*********************  init   () ****************************/
void init(int argc, char **argv){
  int i;
  if( argc != 4 ){
    fprintf(stderr,"Usage: %s <Nwalk> <N> <L>\n",basename(argv[0]));
    exit(1);
  }
  Nwalk       = atol(argv[1]);
  N           = atoi(argv[2]);
  L           = atoi(argv[3]);
  if( L%2 == 0 ) L++;  //we accept odd L only
  L2    = L*L;
  printf("# Size of lattice: L= %d  L2= %d\n",L,L2);
  printf("# Nwalk= %ld SAW random walks with Nstep= %d steps\n",Nwalk,N);
  /*Allocate memory for the h[L2] array */
  h   = (int *)malloc((size_t) (L2*sizeof(int)));
  if(h  ==NULL){fprintf(stderr,"saw2: Not enough memory!\n");exit(1);}
  for(i=0;i<L2;i++) {h[i] = 0;} /*Init to zero*/
  init_random();/*Initialize random number generator*/
```

```
}

/*********************  generate_a_walk() ********************/
void generate_a_walk(){
  int i,ir,r,r0;
  int dir;
  int boing; //count hits at the edge of lattice
  int nret ; //count no. of comeback times
  boing=0; nret=0;
  for(i=0;i<L2;i++) h[i] = 0;
  /*r is the position of the walker on the lattice*/
  r = L2/2; /*We place walker at the center of lattice (L2 is odd!)*/
  r0= r;
  dir=(int)(drandom()*4);/*dir is the stored direction of the previous step*/
  for(i=0;i<N;i++){ /*N steps foreach walk*/
    /*3 possible choices: (dir-1)%4,(dir)%4,(dir+1)%4*/
    ir=dir+3+(int)(drandom()*3);//ir always >0 for % operation, we add 4
    ir %= 4;
    dir = ir;//Stored for next move
    //The four directions are now stored counterclockwise:
    switch(ir){
    case 0: // Move in +X direction
      if((++r )>=L2){r -= L2;}//carefull ++r not r++!
      if(h[r]++){return;}
      break;
    case 2: // Move in -X direction
      if((--r )< 0 ){r += L2;}//carefull --r not r--!
      if(h[r]++){return;}
      break;
    case 1: // Move in +Y direction
      if((r+=L)>=L2){r -= L2;}
      if(h[r]++){return;}
      break;
    case 3: // Move in -Y direction
      if((r-=L)< 0 ){r += L2;}
```

```
    if(h[r]++){return;}
    break;
   }//switch(ir)
   R = r; i2 = i; //0(zero) does not count as a step, just choose the first site
                  //that the chain occupies!!!!!!!!!
                  //-> so not i2=i+1
                  //and h[center] = 0 !!!!!!!!!!!!!!!!!
   //if(i>55){printf("w %ld %d %d\n",iwalk,i,r);}//if you wish to print the path
  }//for(i=0;i<N;i++)
}//generate_a_walk
/************************  end()     ****************************/
void end(){
  exit(0);
}//end
/************************  drandom() ****************************/
#define a 16807
#define m 2147483647
#define q 127773
#define r 2836
#define conv (1.0/(m-1))

long seed;

void init_random(){

  seed = (long) time( (time_t) 0);
  printf("# Initiated drandom() with seed= %ld\n",seed);
  printf("#iwalk N     R^2     x  y R(table) MA\n");
}

double drandom(){
  long l;

  l    = seed/q;
  seed = a*(seed-q*l) - r*l;
```

```
  if(seed < 0) seed +=m;
  return conv*(seed-1);
}
/************************  measure() ****************************/
void measure(){
  int x,y; //translate r-position to x,y-position on the lattice
  int i,Lh;
  double xr,yr;
  double l0,l1,l2,l3,A0,A1,A2;
  double R2;

  Lh = L/2;
  /* Final position x,y,R2:        */
  y  = R/L  ; x  = R - y*L;
  y -= Lh   ; x -= Lh;//origin at the center of lattice
  yr = (double)y; xr = (double)x;
  R2 = xr*xr+yr*yr;
  printf  ("R %ld %d %f %d %d %d %d\n",iwalk,i2,R2,x,y,R,MA);

  /* Area and length measurements: */
  //A0= (double)N; l0 = R2;
  //A1 = measA1(); measA2l1(&A2,&l1);measl2l3(&l2,&l3);
  //printf("Al %d %f %f %f %f %f %f %f\n",iwalk,A0,A1,A2,l0,l1,l2,l3);
  //printf("Al %d %f %f %f %f %f %f %f\n",iwalk,A0,A1,A2,l0,l1,-1.,-1.);

  /* Average histogram             */
  /*   for(i=0;i<L2;i++){
    hav[i] += h[i];
    if(iwalk <= 10 ){
      y=i/L;x=i-y*L;x-=Lh;y-=Lh;//if you wish to print all hs'
      printf("hst %d %d %d %d\n",iwalk,x,y,h[i]);
    }//if(iwalk <= 10 )
    }//for(i=0;i<L2;i++)*/
  fflush(stdout);
}//measure
```

```
/************************energy*************************************/

void energy(){
  int k,nn;

  MA=0;
    for (k=0;k<L2;k++){
      if (h[k]==1){
            if ((nn=k+XNN)>=L2) nn -= L2; MA+=h[nn];
            if ((nn=k-XNN)<  0) nn += L2; MA+=h[nn];
            if ((nn=k+YNN)>=L2) nn -= L2; MA+=h[nn];
            if ((nn=k-YNN)<  0) nn += L2; MA+=h[nn];
            }//endif
    }//for(k=0...
    MA/=2;
    MA-=i2; // i2 =(N-1) => chain's links | notice that h[center] remains 0 at the first step
}//energy
```

Βιβλιογραφία

[1] "Monte Carlo Methods in Statistical Physics" M. E. J. Newman, G.T. Barkema 1998

[2] "Σημειώσεις Υπολογιστικής Φυσικής ΙΙ" Κωνσταντίνος Αναγνωστόπουλος 2005

[3] "Monte Carlo Methods fot the Self Avoiding Walk" ALAN D. SOKAL 1994

[4] "A Faster Implementation of the Pivot Algorithm" Tom Kennedy 2002

[5] "Monte Carlo Simulation in Statistical Physics: An Introduction (Kindle Edition)" Kurt Binder and Dieter W. Heermann 1988