pySecDec: A Toolbox for the Numerical Evaluation of multi-scale Integrals

Stephan Jahn[1]

[1]Max-Planck-Institut für Physik, München

Corfu Summer Institute
Workshop on Particle Physics and Cosmology TOOLS

September 13, 2017

[1]sjahn@mpp.mpg.de

What is pySecDec?

successor of SecDec-3

S. Borowka, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1502.06595]

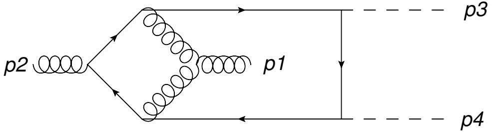
Numerically computes regulated parameter integrals of the form

$$\mathcal{I} \equiv \int_0^1 \mathrm{d}x_1 \ldots \int_0^1 \mathrm{d}x_N \prod_{i=1}^{m} f_i\left(\vec{x}, \vec{a}\right)^{b_i + \sum_k c_{ik}\epsilon_k}$$

where the f_i are polynomials. Typically: $\mathcal{I}\big|_{\epsilon_k=0} = \infty$.

Example:
Loop integrals

after Feynman parametrization



The SecDec collaboration

Sophia Borowka

Gudrun Heinrich

Stephan Jahn

Stephen Jones

Matthias Kerner

Johannes Schlenk

former members

Thomas Binoth

Jonathon Carter

Tom Zirke

Paper

[1703.09692] submitted to CPC

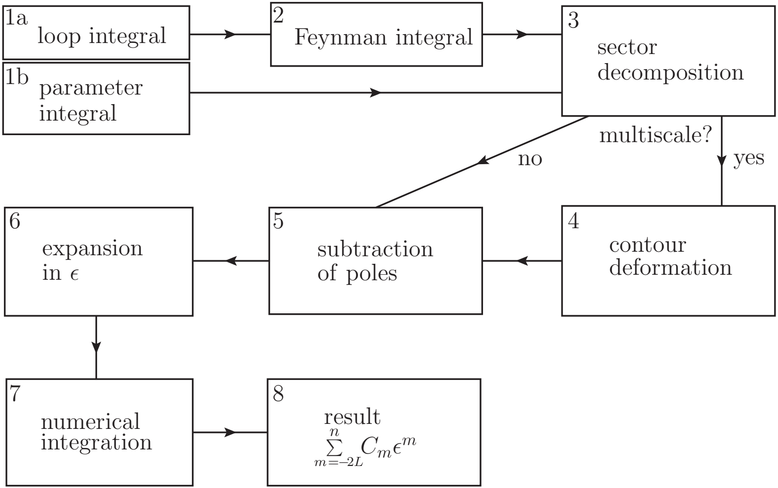
Homepage

`http://secdec.hepforge.org/`

Other Implementations

- *C. Bogner, S. Weinzierl*
  Sector decoposition
  [0709.4092]
- *A.V. Smirnov*
  FIESTA 4
  [1511.03614]

Flowchart


1a loop integral
1b parameter integral
2 Feynman integral
3 sector decomposition
multiscale?
no
yes
4 contour deformation
5 subtraction of poles
6 expansion in ϵ
7 numerical integration
8 result
$\sum_{m=-2L}^{n} C_m \epsilon^m$

Loop Integral - Momentum Representation

$$\mathcal{I} = \int d^D k_1 \cdot \ldots \cdot d^D k_L \frac{1}{P_1^{\nu_1} \cdot \ldots \cdot P_N^{\nu_N}}$$

D: dimensionality

L: number of loops

N: number of propagators

P_i: propagators ($< momentum >^2 \left[- < mass >^2\right] + i\delta$)

ν_i: propagator powers

Loop Integral - Feynman Representation

$$\mathcal{I} = (-1)^{N_\nu} \frac{\Gamma(N_\nu - LD/2)}{\prod_{j=1}^{N} \Gamma(\nu_j)} \int_0^1 \prod_{j=1}^{N} dx_j x_j^{\nu_i - 1} \delta\left(1 - \sum_{n=1}^{N} x_n\right) \frac{\mathcal{U}^{N_\nu - (L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}}$$

D: dimensionality

L: number of loops

N: number of propagators

ν_i: propagator powers

$N_\nu = \sum_{i=1}^{N} \nu_i$

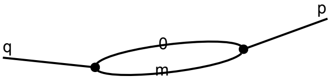
$\mathcal{U} = \mathcal{U}(\vec{x})$: 1st Symanzik polynomial

$\mathcal{F} = \mathcal{F}(\vec{x}, p_i \cdot p_j, m_i^2)$: 2nd Symanzik polynomial

Feynman Parametrization with pySecDec

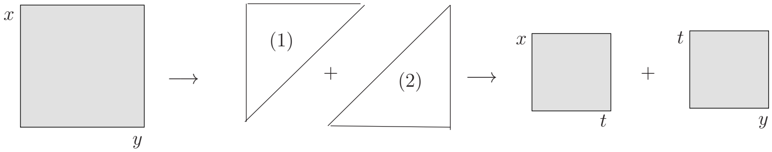
```
>>> from pySecDec.loop_integral import
 ↪    LoopIntegralFromPropagators

>>> one_loop_bubble =
 ↪    LoopIntegralFromPropagators(
...    propagators=['k^2-m^2', '(k-p)^2'],
...    loop_momenta=['k']
...)

>>> one_loop_bubble.exponentiated_U
( + (1)*x0 + (1)*x1)**(2*eps - 2)

>>> one_loop_bubble.exponentiated_F
( + (m**2 - p**2)*x0*x1 +
 ↪     (m**2)*x0**2)**(-eps)

>>> one_loop_bubble.Gamma_factor
gamma(eps)
```

```
>>> from pySecDec.loop_integral import
 ↪    LoopIntegralFromGraph, plot_diagram

>>> one_loop_bubble = LoopIntegralFromGraph(
...   internal_lines=(['m',(1,2)], [0,(1,2)]),
...   external_lines=[('p',1), ('q',2)],
...   replacement_rules=[('q','p')]
...)

>>> one_loop_bubble.exponentiated_U
( + (1)*x0 + (1)*x1)**(2*eps - 2)

>>> one_loop_bubble.exponentiated_F
( + (m**2 - p**2)*x0*x1 +
 ↪     (m**2)*x0**2)**(-eps)

>>> one_loop_bubble.Gamma_factor
gamma(eps)

>>> plot_diagram(
...     one_loop_bubble.internal_lines,
...     one_loop_bubble.external_lines,
...     filename='bubble1L',
...     Gstart=986089
... )
```



Sector Decomposition

or: Resolution of Overlapping Singularities



$$\int_0^1 dx \int_0^1 dy \ (x+y)^{a+b\epsilon}\, f(x,y)$$

$$= \int_0^1 dx \int_0^1 dy \ (x+y)^{a+b\epsilon}\, f(x,y)[\underbrace{\Theta(x-y)}_{(1)} + \underbrace{\Theta(y-x)}_{(2)}]$$

$$= \int_0^1 dx \int_0^1 dt\, x\, x^{a+b\epsilon}\,(1+t)^{a+b\epsilon}\, f(x,xt) + \int_0^1 dt \int_0^1 dy\, y\, y^{a+b\epsilon}\,(t+1)^{a+b\epsilon}\, f(yt,y)$$

Sector Decomposition

or: Resolution of Overlapping Singularities

```
>>> import pySecDec as psd

>>> # define the integration variables
>>> integration_variables = ['x','y']

>>> # define the polynomial to be decomposed
>>> poly = psd.algebra.Expression('(x+y) ** (a+b*eps)', integration_variables)

>>> # keep track of variable transformations
>>> x = psd.algebra.Expression('x', integration_variables)
>>> y = psd.algebra.Expression('y', integration_variables)

>>> # initialize the decomposition
>>> initial_sector = psd.decomposition.Sector([poly], [x,y])
>>> print(initial_sector)
Sector:
Jacobian= + (1)
cast=[(( + (1))**( + (a + b*eps))) * (( + (1)*x + (1)*y)**( + (a + b*eps)))]
other=[ + (1)*x,  + (1)*y]

>>> # perform the decomposition
>>> for sector in psd.decomposition.iterative.iterative_decomposition(initial_sector):
...     print(sector)
...     print()
Sector:
Jacobian= + (1)*x
cast=[(( + (1)*x)**( + (a + b*eps))) * (( + (1) + (1)*y)**( + (a + b*eps)))]
other=[ + (1)*x,  + (1)*x*y]

Sector:
Jacobian= + (1)*y
cast=[(( + (1)*y)**( + (a + b*eps))) * (( + (1)*x + (1))**( + (a + b*eps)))]
other=[ + (1)*x*y,  + (1)*y]
```

Subtraction of Poles

$$\int_0^1 dt\, t^{-1+b\epsilon} g(t)$$

$$= \int_0^1 dt\, t^{-1+b\epsilon} \left(g(0) + g(t) - g(0)\right)$$

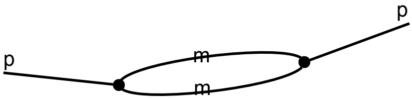
$$= \underbrace{\int_0^1 dt\, t^{-1+b\epsilon} g(0)}_{=\frac{1}{b\epsilon} g(0)} + \underbrace{\int_0^1 dt\, t^{-1+b\epsilon} \left(g(t) - g(0)\right)}_{\text{finite for } \epsilon \to 0 \text{, expand integrand in } \epsilon}$$

Subtraction of Poles

```
>>> import pySecDec as psd
>>> import sympy as sp

>>> # define the essential symbols
>>> integration_variables = ['t']
>>> regulators = ['eps']
>>> symbols = integration_variables + regulators

>>> # define "t^(-1 + b*eps)" and "g(t)"
>>> t_monomial = psd.algebra.Expression('t**(-1 + b*eps)', symbols)
>>> g = psd.algebra.Expression('g(t)', symbols)

>>> # pack the monomial (can have more than one in general)
>>> monomials = psd.algebra.Product(t_monomial)

>>> # need an initializer for the pole part
>>> polynomial_one = psd.algebra.Polynomial.from_expression(1, symbols)
>>> pole_part_initializer = psd.algebra.Pow(polynomial_one, -polynomial_one)

>>> # perform the subtraction
>>> subtraction_initializer = psd.algebra.Product(monomials, pole_part_initializer, g)
>>> subtracted = psd.subtraction.integrate_pole_part(subtraction_initializer, 0)

>>> # pretty representation
>>> for term in subtracted:
...     print(sp.sympify(term))
g(0)/(b*eps)
t**(b*eps - 1)*(-g(0) + g(t))

>>> # internal representation
>>> subtracted
[((( + (1))**( + (b)*eps + (-1)))) * (( + (b)*eps) ** ( + (-1))) * ((g( + (0)))),
 ((( + (1)*t)**( + (b)*eps + (-1)))) * (( + (1)) ** ( + (-1))) * ((g( + (1)*t)) + (( + (-1)) * (g( + (0)))))]
```

Contour Deformation - A Simple Example

consider the massive one loop bubble



$$\int d^D k \frac{1}{(k^2 - m^2)\left((k-p)^2 - m^2\right)}$$

taking $\delta\left(1 - x_1 - x_2\right)$ into account:

$$\begin{aligned}\mathcal{F}\left(\vec{x}, p_i \cdot p_j, m_i^2\right) &= \left[m^2\left(x_1 + x_2\right)^2 - p^2 x_1\, x_2\right]_{x_2 = 1 - x_1} \\ &= m^2 - p^2 x_1 (1 - x_1)\end{aligned}$$

- physical threshold $p^2 \geq 4m^2$
- can have $\mathcal{F} = 0$ although $x_1 \neq 0$

Contour Deformation

or: Satisfying the Feynman Prescription

$$\mathcal{I} = \int d^D k_1 \cdot \ldots \cdot d^D k_L \frac{1}{P_1^{\nu_1} \cdot \ldots \cdot P_N^{\nu_N}}$$

$$= (-1)^{N_\nu} \frac{\Gamma(N_\nu - LD/2)}{\prod_{j=1}^N \Gamma(\nu_j)} \int_0^1 \prod_{j=1}^N dx_j x_j^{\nu_i - 1} \delta\left(1 - \sum_{n=1}^N x_n\right) \frac{\mathcal{U}^{N_\nu - (L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}}$$

P_i: propagators $(< momentum >^2 \; [- < mass >^2] + \boxed{i\delta})$

$\mathcal{F} = \mathcal{F}(\vec{x}, p_i \cdot p_j, m_i^2) - \boxed{i\delta}$: 2^{nd} Symanzik polynomial

Contour Deformation

or: Satisfying the Feynman Prescription

$$\mathcal{F}(\vec{x}, p_i \cdot p_j, m_i^2) - \boxed{i\delta}$$

move integration contour to the complex plane

$$\begin{aligned} \vec{z}: \quad [0,1]^n &\longrightarrow \mathbb{C}^n \\ x_k &\to z_k(\vec{x}) \\ &\equiv x_k - i\lambda_k\, x_k(1-x_k)\frac{\partial\,\mathrm{Re}[\mathcal{F}(\vec{x})]}{\partial x_k} \end{aligned}$$

such that

$$\mathrm{Im}[\mathcal{F}(\vec{z}(\vec{x}))] \leq \mathrm{Im}[\mathcal{F}(\vec{x})] \quad \forall\, \vec{x} \in [0,1]^n$$

Contour Deformation

or: Satisfying the Feynman Prescription

$$\begin{aligned}\mathcal{F}\left(\vec{z}(\vec{x})\right) = &+ \operatorname{Re}\mathcal{F}(\vec{x}) + i\operatorname{Im}\mathcal{F}(\vec{x}) \\ &+ \sum_k \lambda_k \left[\boxed{-i\left(\frac{\partial \operatorname{Re}\mathcal{F}(\vec{x})}{\partial x_k}\right)^2} + \frac{\partial \operatorname{Re}\mathcal{F}(\vec{x})}{\partial x_k}\frac{\partial \operatorname{Im}\mathcal{F}(\vec{x})}{\partial x_k}\right] x_k(1-x_k) \\ &+ \sum_{k,l} \frac{\lambda_k\lambda_l}{2}\left[-\frac{\partial^2 \operatorname{Re}\mathcal{F}(\vec{x})}{\partial x_k \partial x_l} \boxed{-i\frac{\partial^2 \operatorname{Im}\mathcal{F}(\vec{x})}{\partial x_k \partial x_l}}\right] \prod_{i=k,l} \frac{\partial \operatorname{Re}\mathcal{F}(\vec{x})}{\partial x_i} x_i(1-x_i) \\ &+ \mathcal{O}(\lambda^3)\end{aligned}$$

- correct sign at $\mathcal{O}(\lambda)$
- indeterminate sign at $\mathcal{O}(\lambda^2)$

correct sign if λ_k "small enough"

Expansion in the Regulator(s) ϵ

- conceptually easy
- Taylor expansion and series multiplication

  poles in ϵ explicitly factorize by construction
- cumbersome bookkeeping

  individual factors must be expanded to different orders

Numerical Integration

- uses CUBA integrator library by default (except 1D)

  T. Hahn [hep-ph/0404043]
- `gsl_integration_cquad` (GNU scientific library) for 1D
- can link any numerical integrator using the C++ interface

Basic Usage

$$\int_0^1 dx \int_0^1 dy (x+y)^{-2+\epsilon} = \frac{1}{\epsilon} + (1 - \log(2)) + O(\epsilon) \approx \frac{1}{\epsilon} + 0.306853 + O(\epsilon)$$

Step 1: write input files

generate_easy.py

```
from pySecDec import make_package

make_package(

name = 'easy',
integration_variables = ['x','y'],
regulators = ['eps'],

requested_orders = [0],
polynomials_to_decompose = ['(x+y)^(-2+eps)'],

)
```

integrate_easy.py

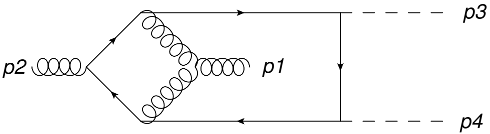
```
from pySecDec.integral_interface \
    import IntegralLibrary

# load c++ library
easy_integral = \
    IntegralLibrary('easy/easy_pylink.so')

# integrate
_, _, result = easy_integral()

# print result
print('Numerical Result:')
print(result)
```

Step 2: run pySecDec

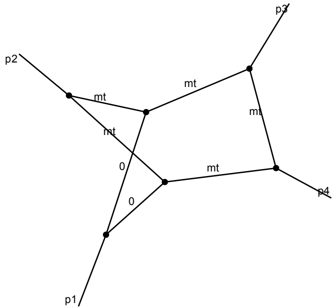
```
$ python generate_easy.py && make -C easy && python integrate_easy.py
<skipped some output>
Numerical Result:
 + (1.00015897181235158e+00 +/- 4.03392522752491021e-03)*eps^-1 + (3.06903035514056399e-01 +/-
 ↪     2.82319349818329918e-03) + O(eps)
```

Application: Higgs Boson Pair Production

S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1608.04798]



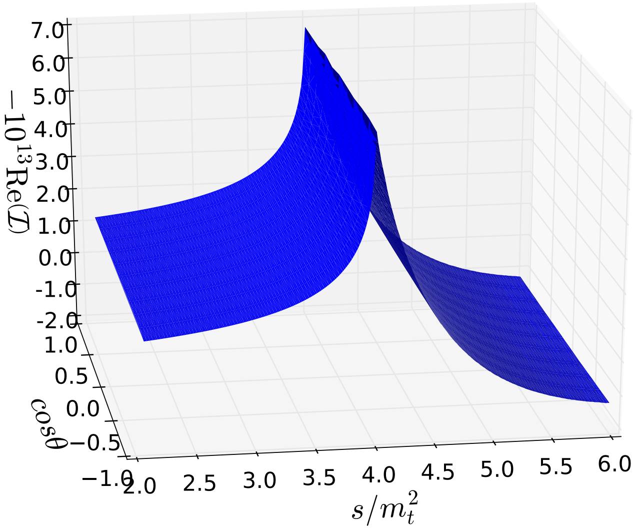
$$\mu^{-6-4\epsilon}\,\mathcal{I} \equiv \text{finite} \left[\vphantom{\int} \right.$$



$$\left. \vphantom{\int} \right], \quad \mu = 1\text{GeV}$$

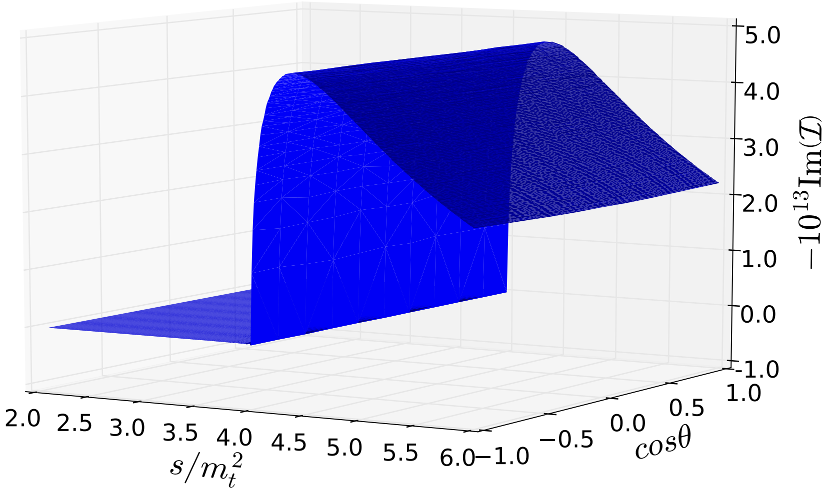
Application: Higgs Boson Pair Production

S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1608.04798]



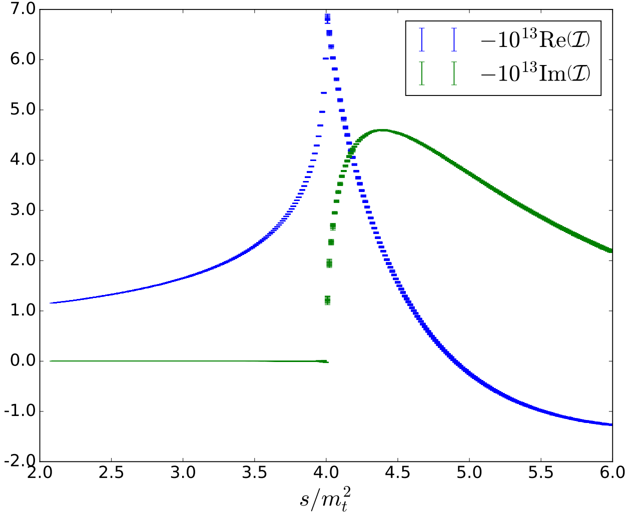
Application: Higgs Boson Pair Production

S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1608.04798]



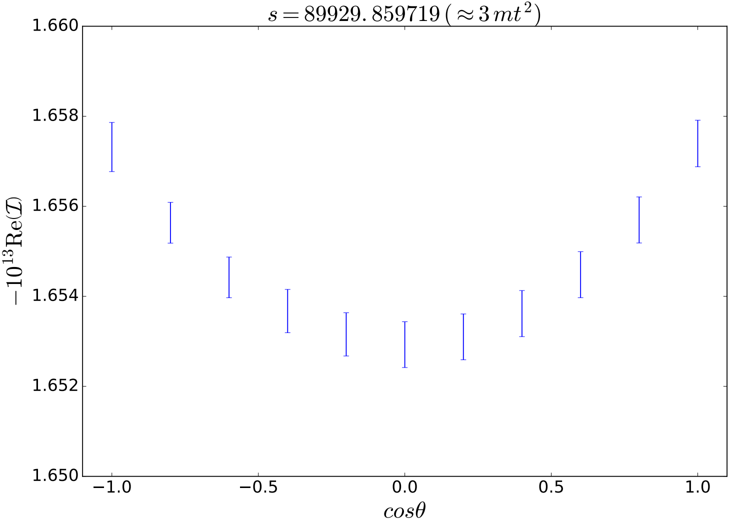
Application: Higgs Boson Pair Production

S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1608.04798]



Application: Higgs Boson Pair Production

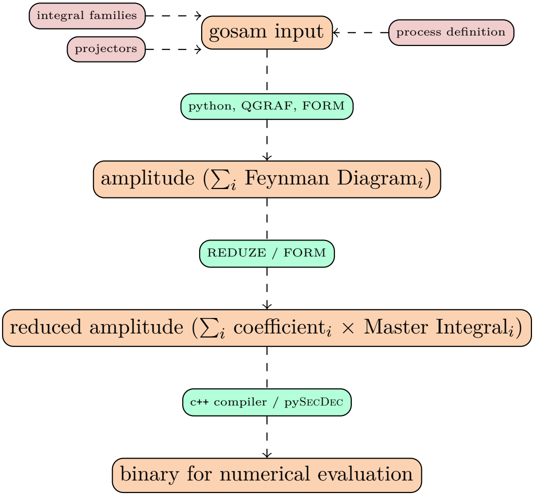
S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke [1608.04798]



Application: GoSam-Xloop

The GoSam-Xloop collaboration

Nicolas Greiner
Gudrun Heinrich
Stephan Jahn
Stephen Jones
Matthias Kerner
et al.



New Features in pySecDec

- fully relying on open-source software
- generates a C++ library suitable for amplitude calculations
- supports more general tensor numerators
- supports multiple regulators
- can handle integrals without Euclidean region
- faster numerics due to **Code Optimization in FORM**

  J. Kuipers, T. Ueda, J.A.M. Vermaseren [1310.7007]
- can handle integrals without Euclidean region
- extended checks of the deformed integration contour

Coming Soon

- quasi monte carlo (qmc) integrator
- numerical integration on GPUs using CUDA

Summary

introduction to the Sector Decomposition approach
as implemented in pySecDec (`http://secdec.hepforge.org/`)

- description of the method
- pySecDec code examples
- application in gg $\rightarrow$ HH [1608.04798]
- application in amplitude generator GoSam-Xloop (unpublished)

BACKUP

Timings

Table 5
Comparison of timings (algebraic, numerical) using pySecDec, SecDec 3 and Fiesta 4.1.

	pySecDec time (s)	SecDec 3 time (s)	Fiesta 4.1 time (s)
triangle2L	(40.5, 9.6)	(56.9, 28.5)	(211.4, 10.8)
triangle3L	(110.1, 0.5)	(131.6, 1.5)	(48.9, 2.5)
elliptic2L_euclidean	(8.2, 0.2)	(4.2, 0.1)	(4.9, 0.04)
elliptic2L_physical	(21.5, 1.8)	(26.9, 4.5)	(115.3, 4.4)
box2L_invprop	(345.7, 2.8)	(150.4, 6.3)	(21.5, 8.8)

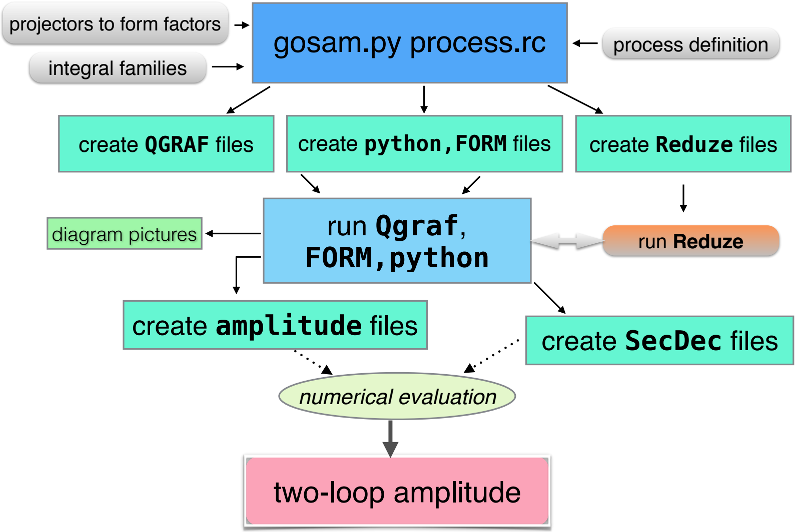
Basic Usage - Analytical Calculation

$$\int_0^1 dx \int_0^1 dy (x+y)^{-2+\epsilon}$$

$$= 2 \int_0^1 dx\, x^{-1+\epsilon} \int_0^1 dt\ (1+t)^{-2+\epsilon}$$

$$= \frac{2}{\epsilon} \left[\int_0^1 dt\ (1+t)^{-2} + \epsilon \int_0^1 dt\ (1+t)^{-2} \log(1+t) + O(\epsilon^2) \right]$$

$$= \frac{2}{\epsilon} \left[\frac{1}{2} + \epsilon \frac{1}{2} (1 - \log(2)) + O(\epsilon^2) \right] = \frac{1}{\epsilon} + (1 - \log(2)) + O(\epsilon)$$


projectors to form factors
integral families
gosam.py process.rc
process definition
create QGRAF files
create python,FORM files
create Reduze files
diagram pictures
run Qgraf, FORM,python
run Reduze
create amplitude files
create SecDec files
numerical evaluation
two-loop amplitude