QCD and precision calculations
Lecture 4: beyond one loop

Gudrun Heinrich

Max Planck Institute for Physics, Munich

PREFIT School, March 4, 2020

MAX-PLANCK-GESELLSCHAFT

cost
EUROPEAN COOPERATION
IN SCIENCE & TECHNOLOGY

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

Contents

- typical workflow to calculate multi-loop amplitudes

- calculation of master integrals:

    - overview on methods

    - multi-loop integrals in Feynman parameter space

    - classification of singularities

    - numerical evaluation of multi-loop integrals

    - sector decomposition: algorithm

    - sector decomposition: hands on

Calculation of loop amplitudes

typical workflow:

- amplitude generation:

$$\mathcal{A}^{\mu_1\cdots\mu_n} = \sum \text{Feynman diagrams}$$ (algebraic expressions)

to saturate Lorentz and spinor indices: project onto helicity amplitudes or decompose into linearly independent Lorentz structures $T_j^{\mu_1\cdots\mu_n}$ multiplying **form factors** F_j : $\mathcal{A}^{\mu_1\cdots\mu_n} = \sum_j F_j \, T_j^{\mu_1\cdots\mu_n}$

construct projectors onto the form factors:

$$F_j = P^{(j)}_{\mu_1\ldots\mu_n} \mathcal{A}^{\mu_1\cdots\mu_n}$$

- reduction to coefficients times master integrals: $F_j = \sum_k c_k^{(j)} I_k^{(j)}$

  use automated tools like Air, Fire, Reduze, LiteRed, Kira, ...

- calculation of master integrals

Calculation of loop amplitudes

typical workflow:

- amplitude generation:

$$\mathcal{A}^{\mu_1 \cdots \mu_n} = \sum \text{Feynman diagrams}$$ (algebraic expressions)

to saturate Lorentz and spinor indices: project onto helicity amplitudes or decompose into linearly independent Lorentz structures $T_j^{\mu_1 \cdots \mu_n}$ multiplying **form factors** F_j : $\mathcal{A}^{\mu_1 \cdots \mu_n} = \sum_j F_j \, T_j^{\mu_1 \cdots \mu_n}$

construct projectors onto the form factors:

$$F_j = P^{(j)}_{\mu_1 \ldots \mu_n} \mathcal{A}^{\mu_1 \cdots \mu_n}$$

- reduction to coefficients times master integrals: $F_j = \sum_k c_k^{(j)} \, I_k^{(j)}$

  use automated tools like Air, Fire, Reduze, LiteRed, Kira, ...

- calculation of master integrals ← focus on this part in the following

2-loop automation: GoSam-Xloop

process definition

projectors to form factors

gosam.py process.rc

integral families

create **QGRAF** files

create **python,FORM** files

create **Reduze** files

diagram pictures

run **Qgraf, FORM,python**

run **Reduze**

create **amplitude** files

create **pySecDec** files

numerical integration

two-loop amplitude

Calculation of master integrals

many different methods:

- differential equations *very successful for analytic evaluations*

  Kotikov ’91; Remiddi ’97, Gehrmann, Remiddi ’00, Henn ’13 ...

- integration of Mellin-Barnes representation

  analytic as well as numerical prominent examples

  V.A. Smirnov ’99; Tausk ’99; Czakon ’05; Dubovyk, Freitas, Gluza, Riemann, Usovitsch ’16, ‘19

- direct evaluation in momentum space

  only possible numerically, needs cancellation of singularities in integrand

  Soper ’99; Gong, Soper, Nagy ‘09; Weinzierl, Reuschle et al. ’10-’20;

  Rodrigo, Sborlini, Driencourt-Mangin, Torres-Bobadilla et al. ’08-’20; Capatti, Hirschi, Kernmaschah, Ruijl ’19

- direct evaluation in Feynman parameter space

  for more complicated integrals analytic solution impractical

  Passarino, Uccirati et al ’01-’18; De Doncker, Fujimoto, Kurihara, et al. ’04-’19, Volkov ‘19

  sector decomposition: Hepp ’66; Denner, Roth ’96; Binoth, GH ’00; Bogner, Weinzierl ’07; Smirnov et al. ’08-’16;
  Anastasiou et al ’08; Carter, GH ’10; Borowka, GH, Jahn, Jones, Kerner, Schlenk, Zirke ’12-‘20

public programs for sector decomposition

- **sector_decomposition** (uses Ginac) (only Euclidean region)

  [Bogner, Weinzierl ’07]

  + **CSectors** for construction of integrand in terms of Feynman parameters

  [Gluza, Kajda, Riemann, Yundin ’10]

- FIESTA (versions 1,2,3,4) (uses Mathematica, C++)

  [A.Smirnov, V.Smirnov, Tentyukov, ’08,’09,’13,‘15]

- (py)SecDec (uses python, C++)

  [J. Carter, GH, ’10 (SecDec-1.0)]

  [Borowka, GH, Jahn, Jones, Kerner, Schlenk, Zirke ’17 (pySecDec)]

  [Borowka, GH, Jahn, Jones, Kerner, Schlenk ’19 (pySecDec on GPU, QMC)]

- (not public) FORM implementation

  [Fujimoto, Kaneko, Ueda ’08,’10]

analytic versus numerical: pro's and con's

	analytic	numerical
pole cancellation	exact	with numerical uncertainty
fast evaluation	☑ (mostly)	depends
control of integrable singularities	☑ control of analytic regions	difficult
extension to more scales	difficult	less difficult
automation	difficult	less difficult

Multi-loop integrals in Feynman parameter space

scalar *L*-loop integral in momentum space:

$$G(\nu_1 \dots \nu_N) = \int \prod_{l=1}^{L} \frac{d^D k_l}{i\pi^{\frac{D}{2}}} \prod_{j=1}^{N} \frac{1}{P_j^{\nu_j}(\{k\},\{p\},m_j^2)}$$

D: spacetime dimension, *N:* number of propagators

p: external momenta, k: loop momenta, example $P_1 = (k_1 + p_1)^2 - m_1^2$

ν_i : propagator powers (also called *indices)*

Feynman parametrisation:

$$\frac{1}{P_1^{\nu_1} P_2^{\nu_2} \dots P_N^{\nu_N}} = \frac{\Gamma(N_\nu)}{\prod_{i=1}^{N} \Gamma(\nu_i)} \int_0^\infty \prod_{i=1}^{N} dx_i \, x_i^{\nu_i - 1} \frac{\delta(1 - \sum_{j=1}^{N} x_j)}{[x_1 P_1 + x_2 P_2 + \dots + x_N P_N]^{N_\nu}}$$

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

turns product of propagators into a single sum

Multi-loop integrals in Feynman parameter space

after Feynman parametrisation:

$$G = \Gamma(N_\nu) \int \prod_{j=1}^{N} dx_j \, x_j^{\nu_j - 1} \, \delta(1 - \sum_{i=1}^{N} x_i) \int d\bar{k}_1 \ldots d\bar{k}_L \left[\sum_{j,l=1}^{L} k_j \cdot k_l \, M_{jl} - 2 \sum_{j=1}^{L} k_j \cdot Q_j + J \right]^{-N_\nu}$$

$$= (-1)^{N_\nu} \frac{\Gamma(N_\nu - LD/2)}{\prod_{j=1}^{N} \Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} dx_j \, x_j^{\nu_j - 1} \delta(1 - \sum_{i=1}^{N} x_i) \, \frac{\mathcal{U}^{N_\nu - (L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}}$$

$$\mathcal{U} = \det(M) \quad , \quad N_\nu = \sum_{j=1}^{N} \nu_j \, , \qquad d\bar{k} = \frac{d^D k}{i\pi^{\frac{D}{2}}}$$

$$\mathcal{F} = \det(M) \left[\sum_{i,j=1}^{L} Q_i M_{ij}^{-1} Q_j - J - i\delta \right] .$$

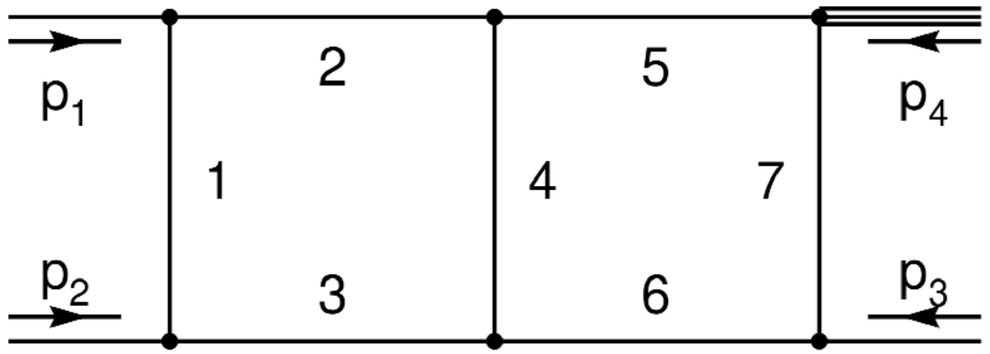
Q_i contains Feynman parameters and external momenta

$\mathcal{U}$: polynomial in Feynman parameters, positive semi-definite

$\mathcal{F}$: depends on x_i and kinematic invariants s_{ij}, m_i^2

Example

two-loop box integral, one off-shell leg



$$\mathcal{U} = x_{123}x_{567} + x_4 x_{123567}$$

$$\mathcal{F} = (-s_{12})(x_2x_3x_{4567} + x_5x_6x_{1234} + x_2x_4x_6 + x_3x_4x_5) + (-s_{23})x_1x_4x_7 + (-p_4^2)x_7(x_2x_4 + x_5x_{1234}) \,,$$

$$x_{ijk\ldots} = x_i + x_j + x_k + \ldots \,, \quad s_{ij} = (p_i + p_j)^2$$

$$G = -\Gamma(3+2\epsilon) \int \mathrm{d}x_1 \ldots \mathrm{d}x_7 \, \delta(1 - \sum_{i=1}^{7} x_i) \, \mathcal{U}^{1+3\epsilon} \, \mathcal{F}^{-3-2\epsilon}$$

can lead to singularities if

$$\mathcal{F} = 0$$

singularity structure

3 types of singularities can arise

1) UV singularities: *show up as poles* $1/\epsilon^{\alpha}$

two types: (a) overall UV poles: $\Gamma(N_\nu - LD/2) \sim \Gamma(n\epsilon)$

(b) UV subdivergences:

come from $\mathcal{U}(x) = 0$ for some $x_i = 0$ with $\mathcal{U}$ having negative exponent

2) IR singularities (soft and collinear): *show up as poles* $1/\epsilon^{\alpha}$

come from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ for some $x_i = 0$ with expo $\mathcal{F}$ negative

3) threshold-type singularities:

$\mathcal{F}(x, s_{ij}, m_i^2) = 0$ inside integration domain for some values of invariants and $\mathbf{x}$

usually integrable, avoid zeros on real axis by integration along contour in complex plane

singularity structure

example 1-loop bubble

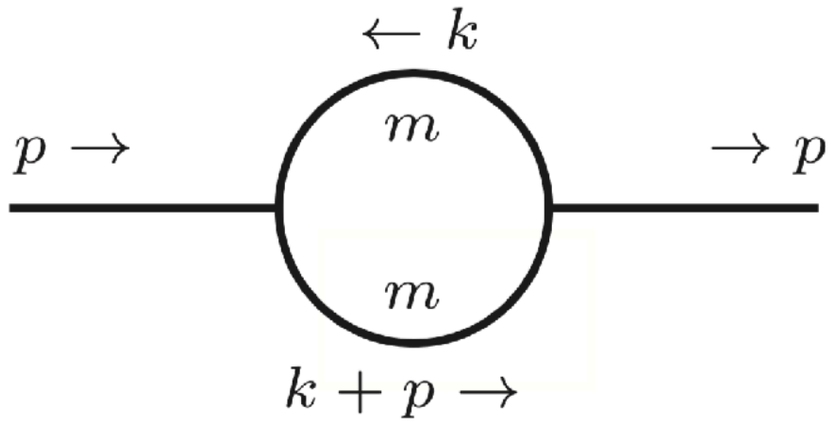


figure: Stephan Jahn

$$s = p^2$$

$$G = \Gamma(\epsilon) \int dx_1 \int dx_2 \, \delta(1 - x_1 - x_2) \underbrace{(x_1 + x_2)^{-2+2\epsilon}}_{\mathcal{U}^{\mathrm{eu}}} \underbrace{[-s x_1 x_2 + m^2 - i\delta]^{-\epsilon}}_{\mathcal{F}^{\mathrm{ef}}}$$

$$= \Gamma(\epsilon) \int_0^1 dx \, [-s\, x(1-x) + m^2 - i\delta]^{-\epsilon}$$

overall UV pole

bracket vanishes at $s = 4m^2, x = \frac{1}{2}$

threshold for production of 2 on-shell particles with mass *m*

$i\delta$ - prescription how to move integration contour away from poles on real x-axis

numerical evaluation of multi-loop integrals

- any numerical method first requires the isolation/subtraction of UV and IR singularities

- methods working in 4 dimensions in momentum space need
  - subtraction of UV divergences at integrand level
  - cancellation of IR divergences:
    KLN theorem, needs careful combination of virtual/real contributions

- methods working in D dimensions need procedure to isolate poles in $1/\epsilon$

one of these methods is sector decomposition

Sector Decomposition algorithm

two strategies: (a) iterative ← outlined below

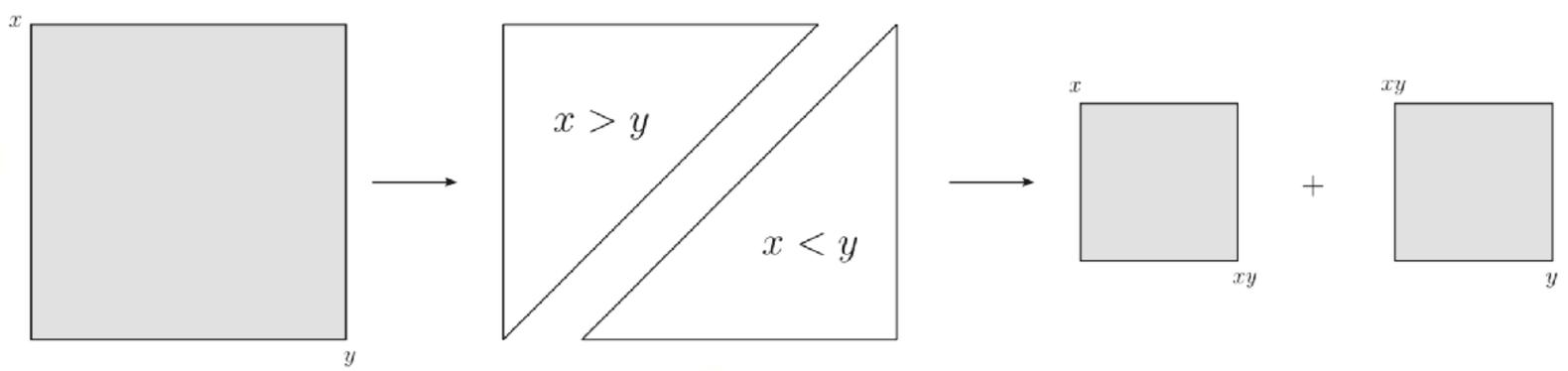
(b) based on algebraic geometry

Bogner, Weinzierl ’08; Fujimoto, Kaneko, Ueda ’10; Schlenk ’15

basic idea:

disentangle overlapping singularities at origin of integration parameter space by dividing into sectors, where one parameter is smallest in each sector

$$\mathcal{I} = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y \; (x+y)^{-2+\varepsilon} \, f(x,y) [\underbrace{\Theta(x-y)}_{x>y} + \underbrace{\Theta(y-x)}_{x<y}]$$



Sector Decomposition algorithm

$$\mathcal{I} = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y\ (x+y)^{-2+\varepsilon}\, f(x,y)[\underbrace{\Theta(x-y)}_{x>y} + \underbrace{\Theta(y-x)}_{x<y}]$$

$$\equiv \underbrace{\int_0^1 \mathrm{d}x \int_0^x \mathrm{d}y\ (x+y)^{-2+\varepsilon}\, f(x,y)}_{\mathcal{I}_{x>y}} + \underbrace{\int_0^1 \mathrm{d}y \int_0^y \mathrm{d}x\ (x+y)^{-2+\varepsilon}\, f(x,y)}_{\mathcal{I}_{x<y}}.$$

substitute $\quad y \to xy \qquad\qquad x \to yx$

result:

$$\mathcal{I}_{x>y} = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y\, x^{-1+\varepsilon}\, (1+y)^{-2+\varepsilon}\, f(x,xy)$$

$$\mathcal{I}_{x<y} = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y\, y^{-1+\varepsilon}\, (x+1)^{-2+\varepsilon}\, f(yx,y).$$

singularities factorised

non-singular at x=0

Sector Decomposition algorithm

remarks:

- functions with more integration variables may need iteration of this procedure
- it is not guaranteed that the iteration stops; geometric decomposition is proven to stop

subtraction of the poles:

in each sector, integrals are of the form (consider just one parameter at the time)

$$\mathcal{G} = \int_0^1 dt\, t^{-1-\epsilon} g(t)$$

use

$$\begin{aligned}\mathcal{G} &= \int_0^1 \mathrm{d}t\, t^{-1+\varepsilon} \left(g(0) + g(t) - g(0)\right) \\ &= \underbrace{\int_0^1 \mathrm{d}t\, t^{-1+\varepsilon} g(0)}_{=\frac{1}{\varepsilon} g(0)} + \underbrace{\int_0^1 \mathrm{d}t\, t^{-1+\varepsilon} \left(g(t) - g(0)\right),}_{\text{finite for } \varepsilon \to 0,\ \text{expand integrand in } \varepsilon}\end{aligned}$$

Sector Decomposition algorithm

after subtractions and expansion in ϵ integral is of the form

$$G = \sum_{j=-2L}^{n} C_j \, \epsilon^j$$

C_j are finite, but complicated parameter integrals

$\rightarrow$ integrate numerically

if all kinematic invariants s_{ij} are negative ("Euclidean kinematics") (and all masses are positive):

$\mathcal{F}(x, s_{ij}, m^2)$ will be ≥ 0

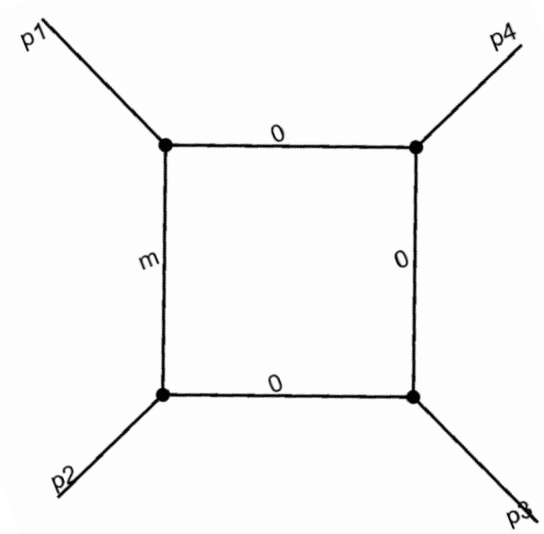
$\Rightarrow$ numerical integration straightforward

(Monte Carlo or quasi-Monte Carlo)

Contour deformation

however usually kinematic invariants have different signs

example box integral (2 to 2 scattering)



for physical kinematics

$$s = (p_1 + p_2)^2 > 0, t = (p_2 - p_3)^2 < 0$$

$$\mathcal{F} = -s\, x_1 x_3 - t\, x_0 x_2 - i\delta$$

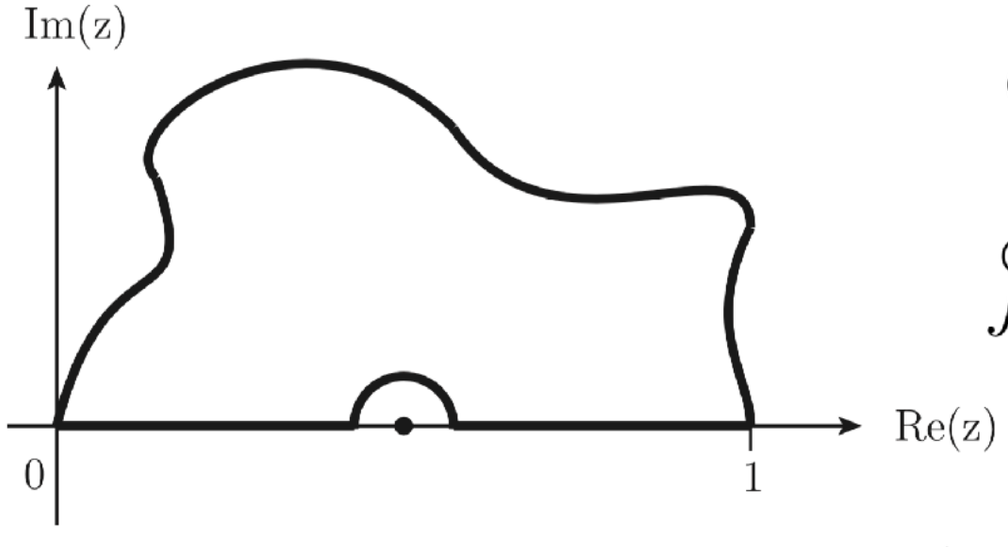
will vanish on a hyperplane inside integration domain

$-i\delta$ saves the day!

→ make integration variables complex

preserve sign of imaginary part to ensure poles are outside contour

Contour deformation



Cauchy:

$$\oint_c \mathrm{d}z I(z) = \int_0^1 \mathrm{d}x I(x) + \int_1^0 \mathrm{d}z I(z) = 0$$

transformation $x_k \to z_k(\vec{x}) = x_k - i\,\tau_k(\vec{x})$

$$\tau_k(\vec{x}) = \lambda_k\, x_k(1-x_k)\, \frac{\partial \mathcal{F}(\vec{x})}{\partial x_k}$$

λ_k : deformation parameter

expand $\mathcal{F}(\vec{z})$ around $\vec{z} = \vec{x}$

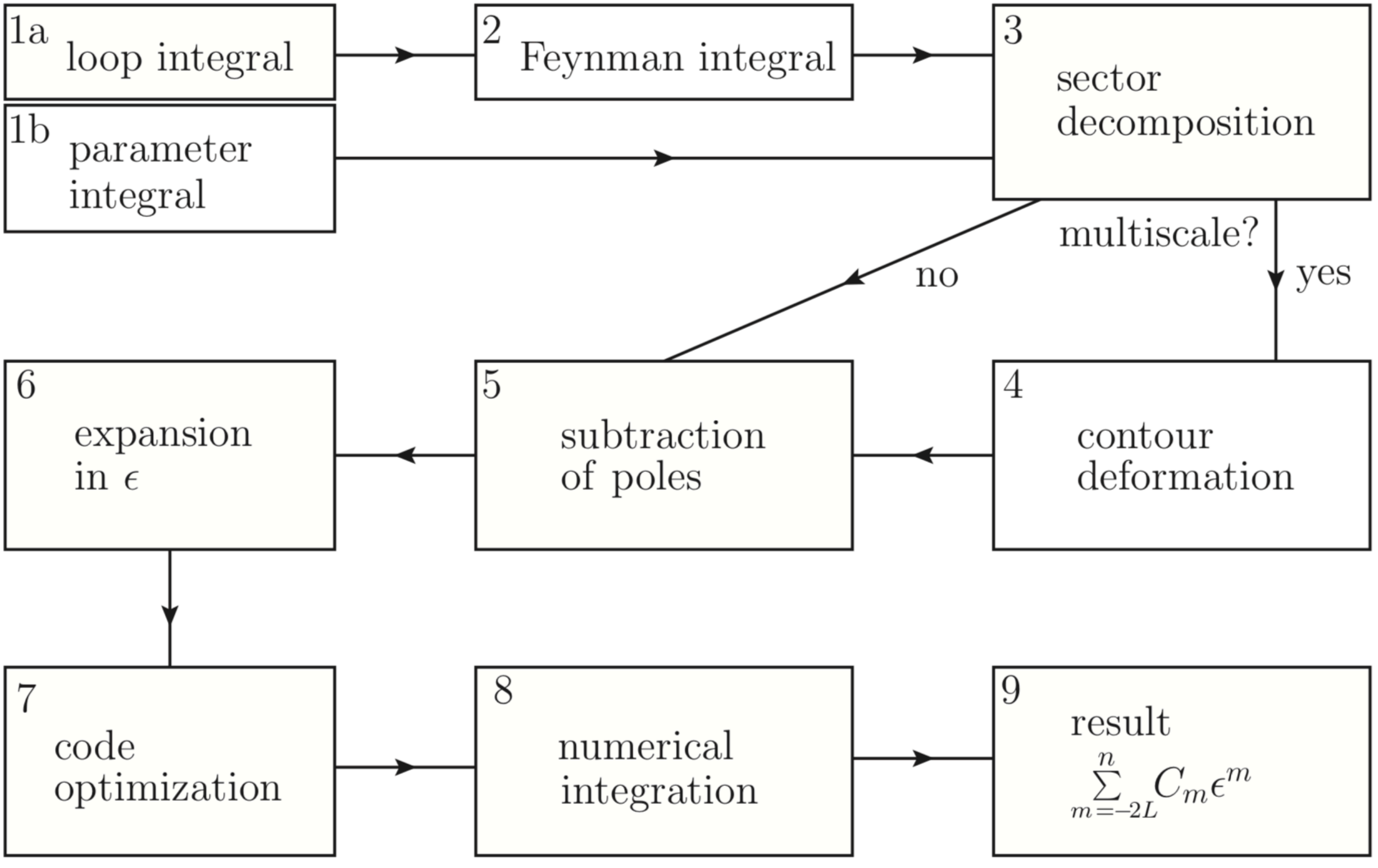
always positive

$$\mathcal{F}(\vec{z}) = \mathcal{F}(\vec{x}) - i \sum_k \lambda_k x_k(1-x_k) \left(\frac{\partial \mathcal{F}(\vec{x})}{\partial x_k}\right)^2 - \frac{1}{2}\sum_j \sum_k \tau_j \tau_k \left(\frac{\partial^2 \mathcal{F}(\vec{x})}{\partial x_j x_k}\right)$$

$$+ \frac{i}{6}\sum_j \sum_k \sum_l \tau_j \tau_k \tau_l \left(\frac{\partial^3 \mathcal{F}(\vec{x})}{\partial x_j x_k x_l}\right) + \ldots$$

does not spoil sign for λ small enough

pySecDec: workflow



pySecDec hands on: download and installation

download the code at

https://github.com/mppmu/secdec/releases

YouTube installation video:

https://www.youtube.com/watch?v=Ughnr3EKgkk

documentation (installation and usage):

https://secdec.readthedocs.io/en/stable/

pySecDec: hands on

after successful installation, go to the folder

examples/easy

this example calculates

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

input file:

generate_easy.py

run the sector decomposition:

python generate_easy.py

```
1  from pySecDec import make_package
2
3  make_package(
4
5  name = 'easy',
6  integration_variables = ['x','y'],
7  regulators = ['eps'],
8
9  requested_orders = [0],
10 polynomials_to_decompose = ['(x+y)
       ^(-2+eps)'],
11
12 )
```

pySecDec: hands on

examples/easy

compile:

```
make -C easy
```

numerical integration file:

integrate_easy.py

run integration:

```
python integrate_easy.py
```

result should be

$$\frac{1}{\epsilon} + 0.306853 + O(\epsilon)$$

```
1  from pySecDec.integral_interface
       import IntegralLibrary
2  from math import log
3
4  # load c++ library
5  easy = IntegralLibrary('easy/
       easy_pylink.so')
6
7  # choose integrator
8  easy.use_Vegas(epsrel=1e-3,epsabs
       =1e-10)
9
10 # integrate
11 _, _, result = easy()
12
13 # print result
14 print('Numerical Result:' + result
       )
15 print('Analytic Result:' + ' +
       (%.15g)*eps^-1 + (%.15g) + O(
       eps)' % (1.0,1.0-log(2.0)))
```

pySecDec: hands on

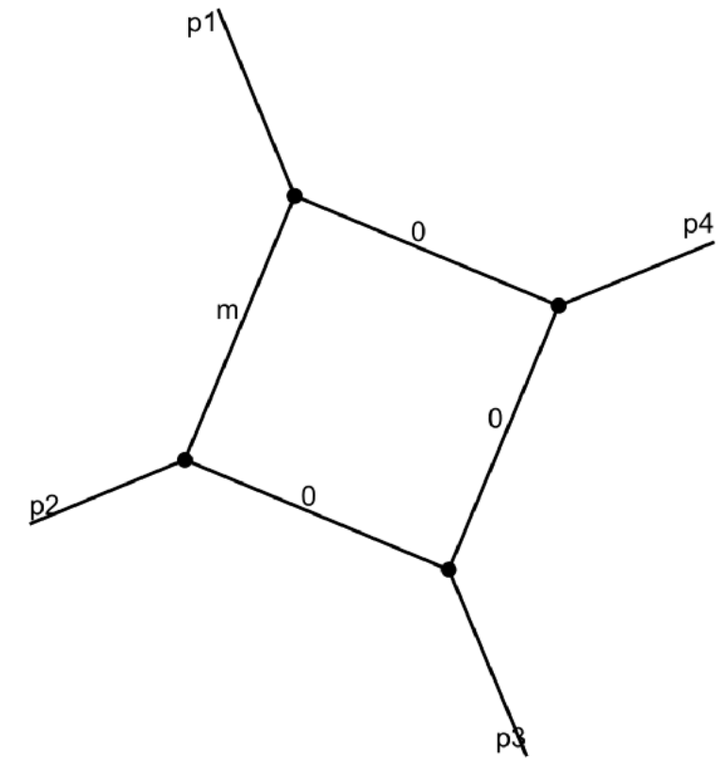
go to the folder `examples/box1L`

- run the sector decomposition:

```
python generate_box1L.py
```

- compile and run integration:

```
make -C box1L
python integrate_box1L.py
```



- look at diagram:

```
cd box1L
acroread box1L.pdf
```

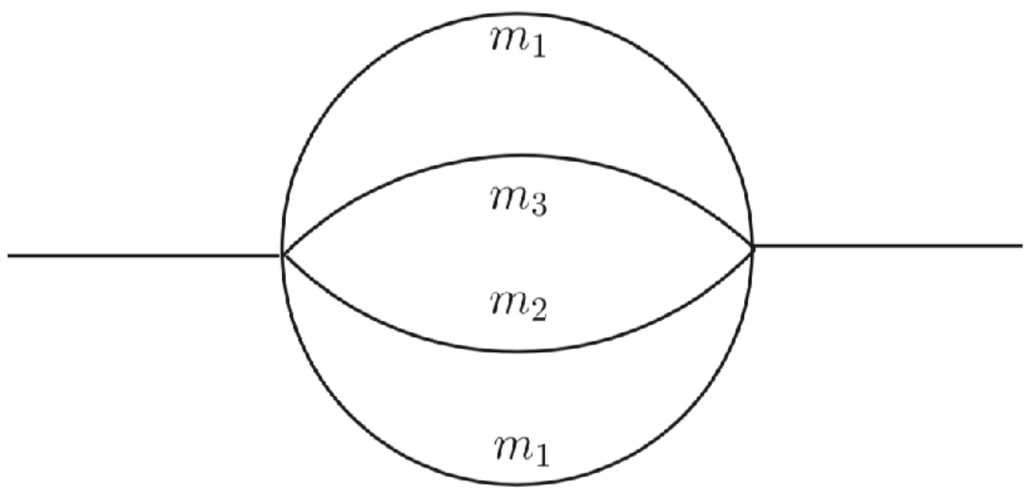
- look at polynomials F and U:

copy definition of `li` from `generate_box1L.py` into python

call `li.F` and `li.U`

pySecDec: hands on

examples/banana_3mass



3 loops, 3 different masses

analytic result unknown

```
python generate_banana_3mass.py
make -C banana_3mass
python integrate_banana_3mass.py
```

use both Vegas and QMC, compare accuracy and timings

(see Table 3 of arXiv:1811.11720)