Numerical evaluation of multi-loop integrals

Stephan Jahn[1]

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

Loops and Legs in Quantum Field Theory 2018

April 30, 2018

[1]sjahn@mpp.mpg.de

Outline

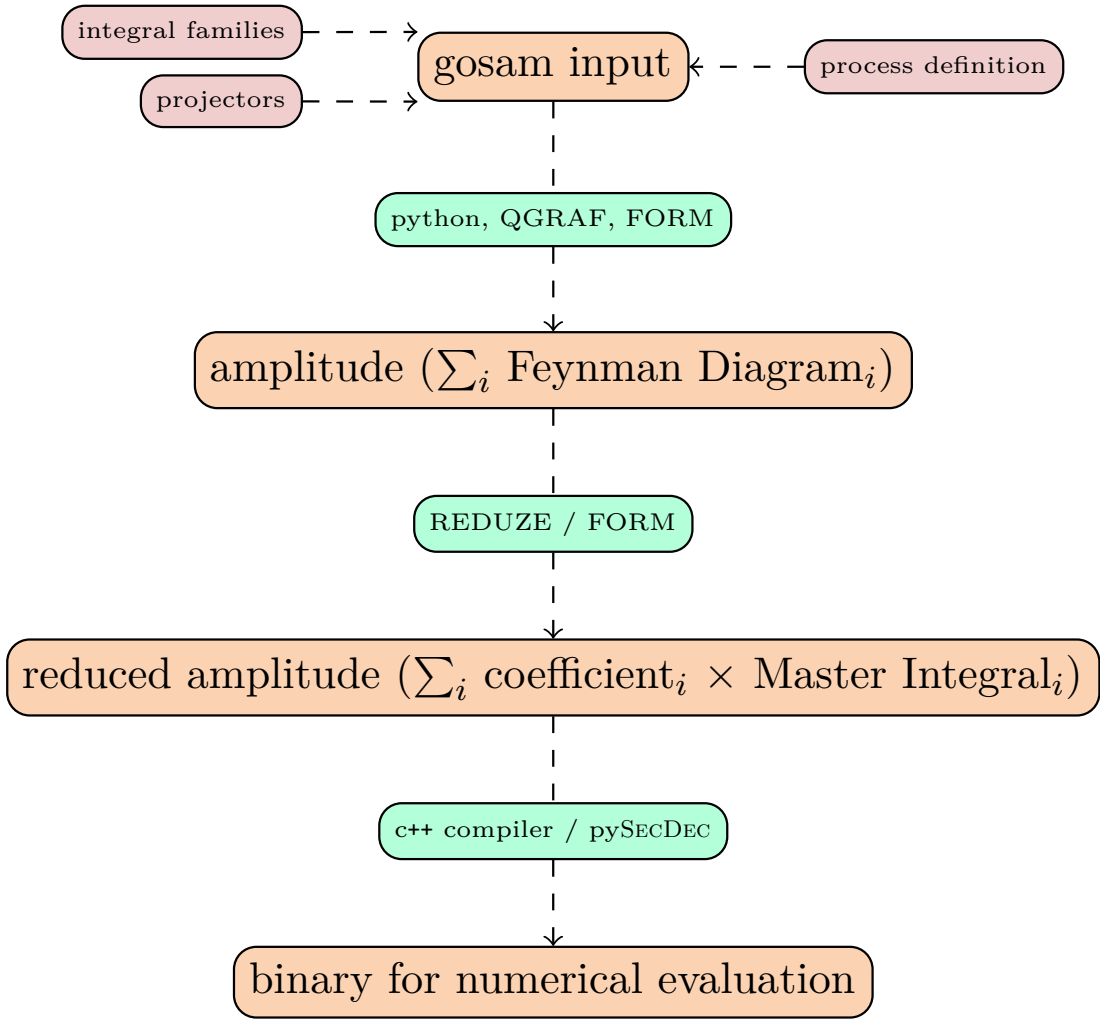
- sector decomposition and pySecDec
- phenomenological applications

- ongoing development of pySecDec

GoSam-Xloop

The GoSam-Xloop collaboration

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



pySecDec

successor of SecDec-3

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

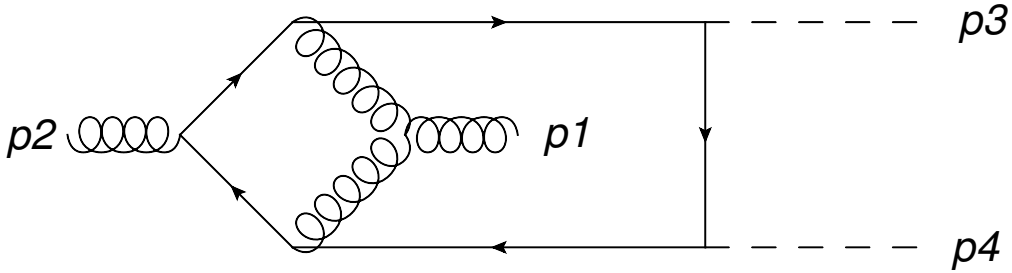
Numerically computes regulated parameter integrals of the form

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

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

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

CPC 222 (2018) 313
[1703.09692]

Homepage

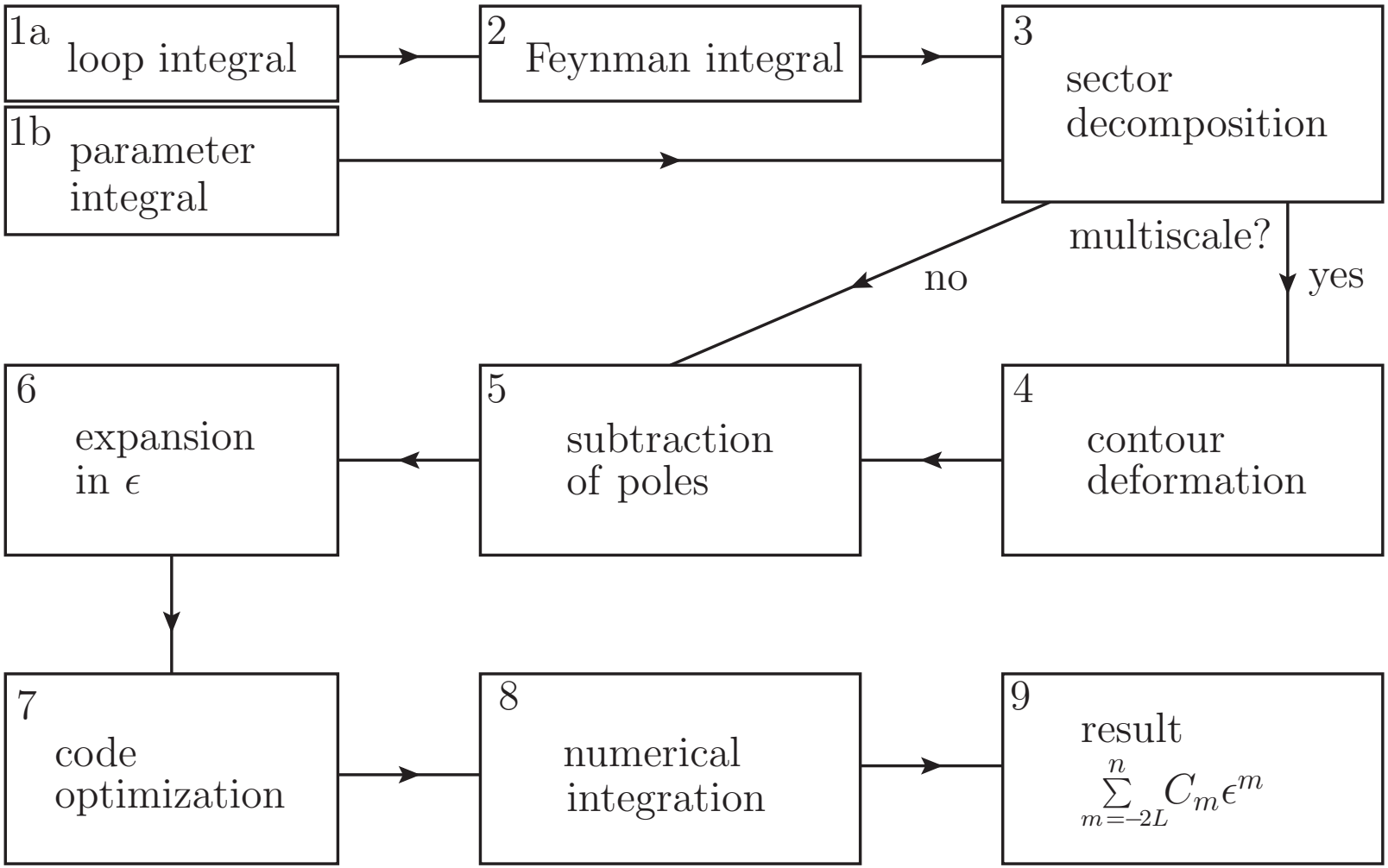
`http://secdec.hepforge.org/`

Other public Implementations

- *C. Bogner, S. Weinzierl*
  Sector decomposition
  [0709.4092]

  supplemented by:
  J. Gluza, K. Kajda, T. Riemann, V. Yundin
  CSectors
  [1010.1667]
- *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 code optimization
8 numerical integration
9 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

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

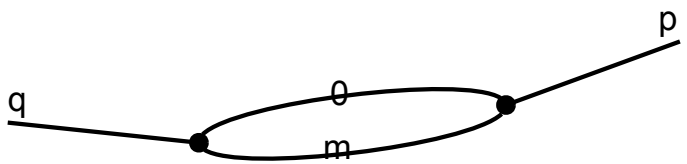
$\mathcal{U} = \mathcal{U}(\vec{x})$: 1st Symanzik polynomial

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

Feynman Parametrization with pySecDec

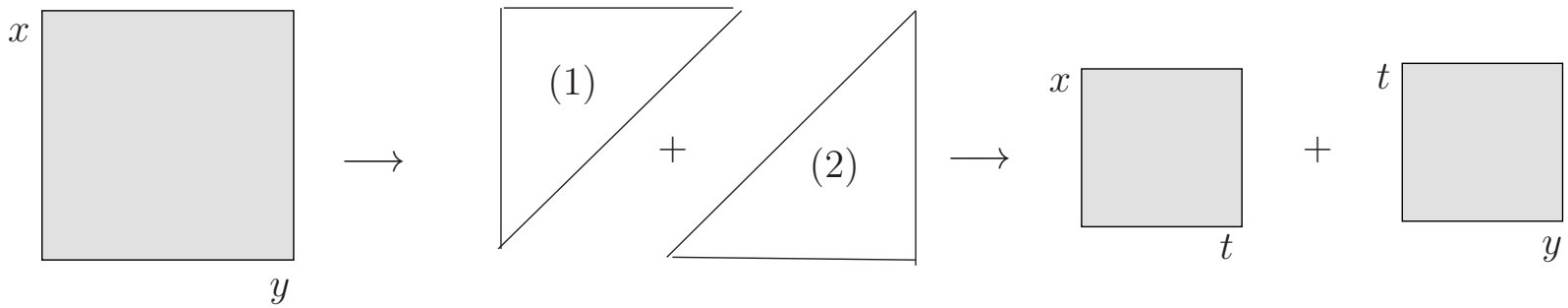
```
>>> from pySecDec.loop_integral import
  ↪    LoopIntegralFromPropagators

>>> 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



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

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

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

Subtraction of Poles

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

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

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

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

```
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 )
```

integrate_easy.py

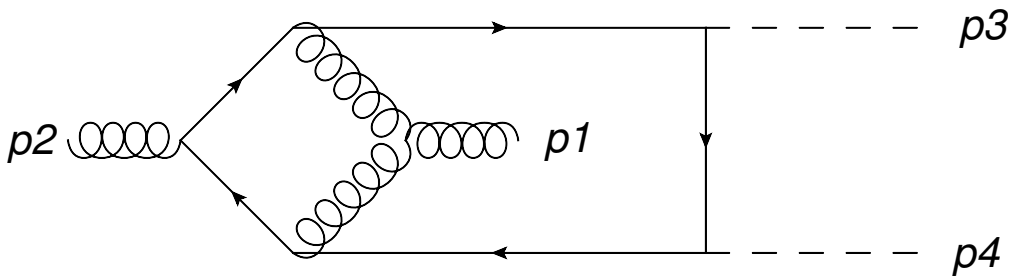
```
1  from pySecDec.integral_interface \
2      import IntegralLibrary
3
4  # load c++ library
5  easy_integral = \
6      IntegralLibrary('easy/easy_pylink.so')
7
8  # integrate
9  _, _, result = easy_integral()
10
11 # print result
12 print('Numerical Result:')
13 print(result)
```

Step 2: run pySecDec

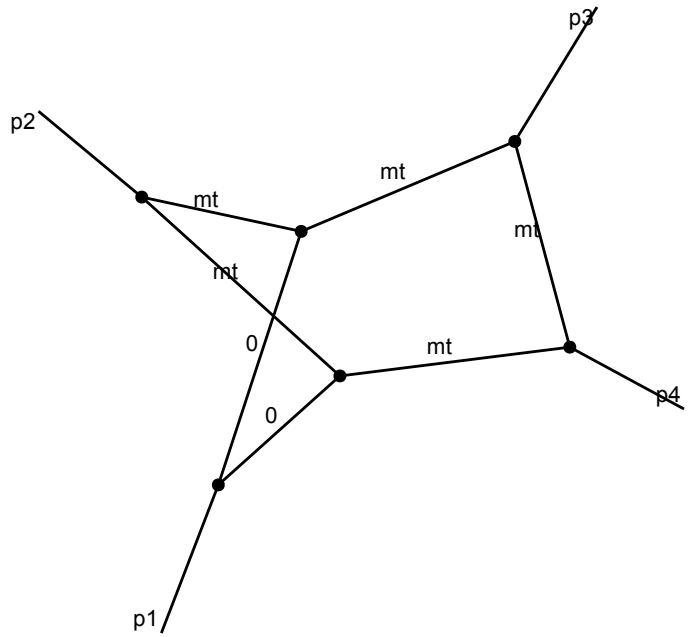
```
1  $ python generate_easy.py && make -C easy && python integrate_easy.py
2  <skipped some output>
3  Numerical Result:
4   + (1.0001589718123515 8e+00 +/- 4.0339252275249102 1e-03)*eps^-1 + (3.0690303551405639 9e-01 +/-
     ↪     2.8231934981832991 8e-03) + O(eps)
```

Higgs Boson Pair Production

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

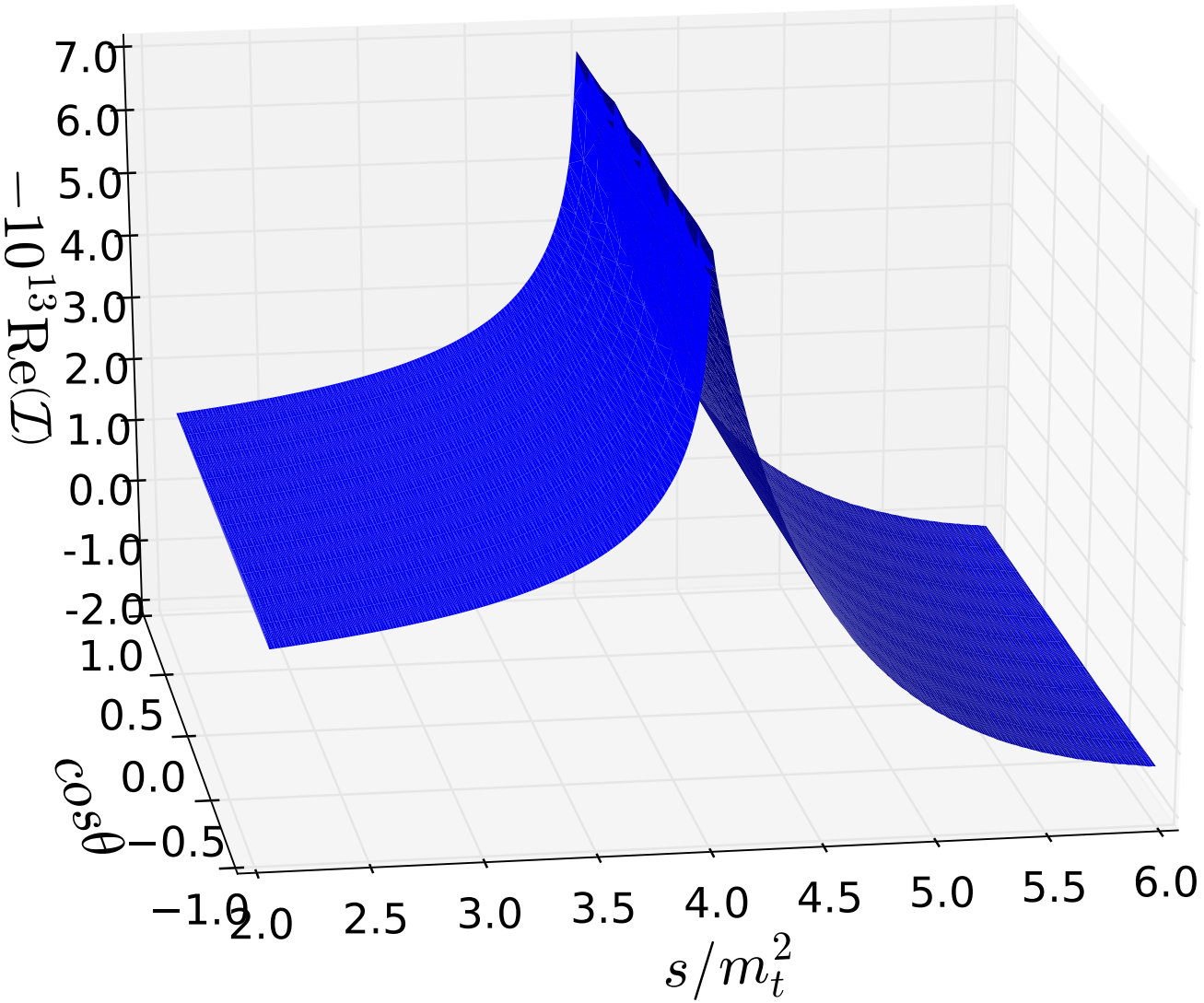


$$\mu^{-6-4\epsilon}\,\mathcal{I} \equiv \text{finite}\left[\;\;\right], \;\mu = 1\text{GeV}$$



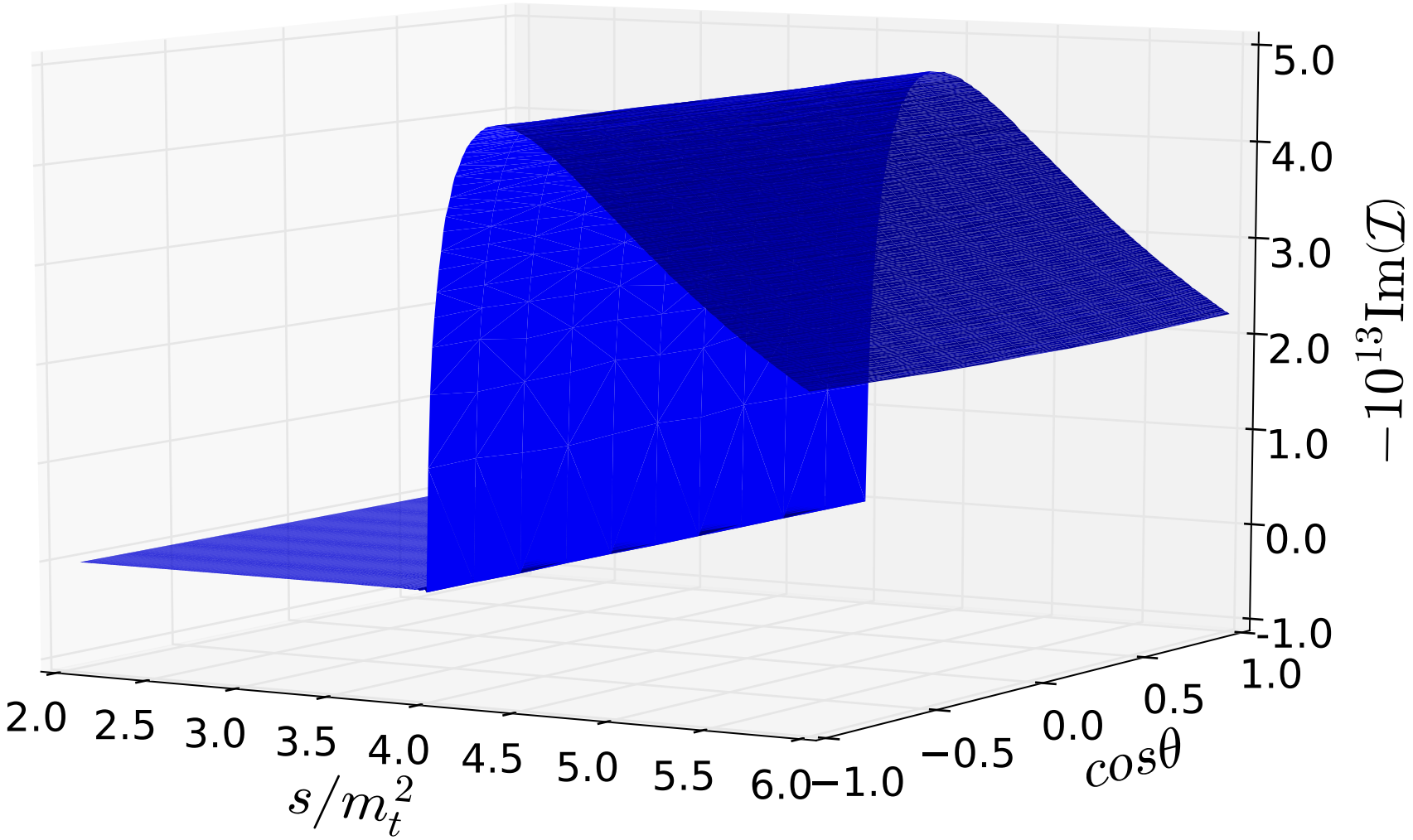
Higgs Boson Pair Production

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



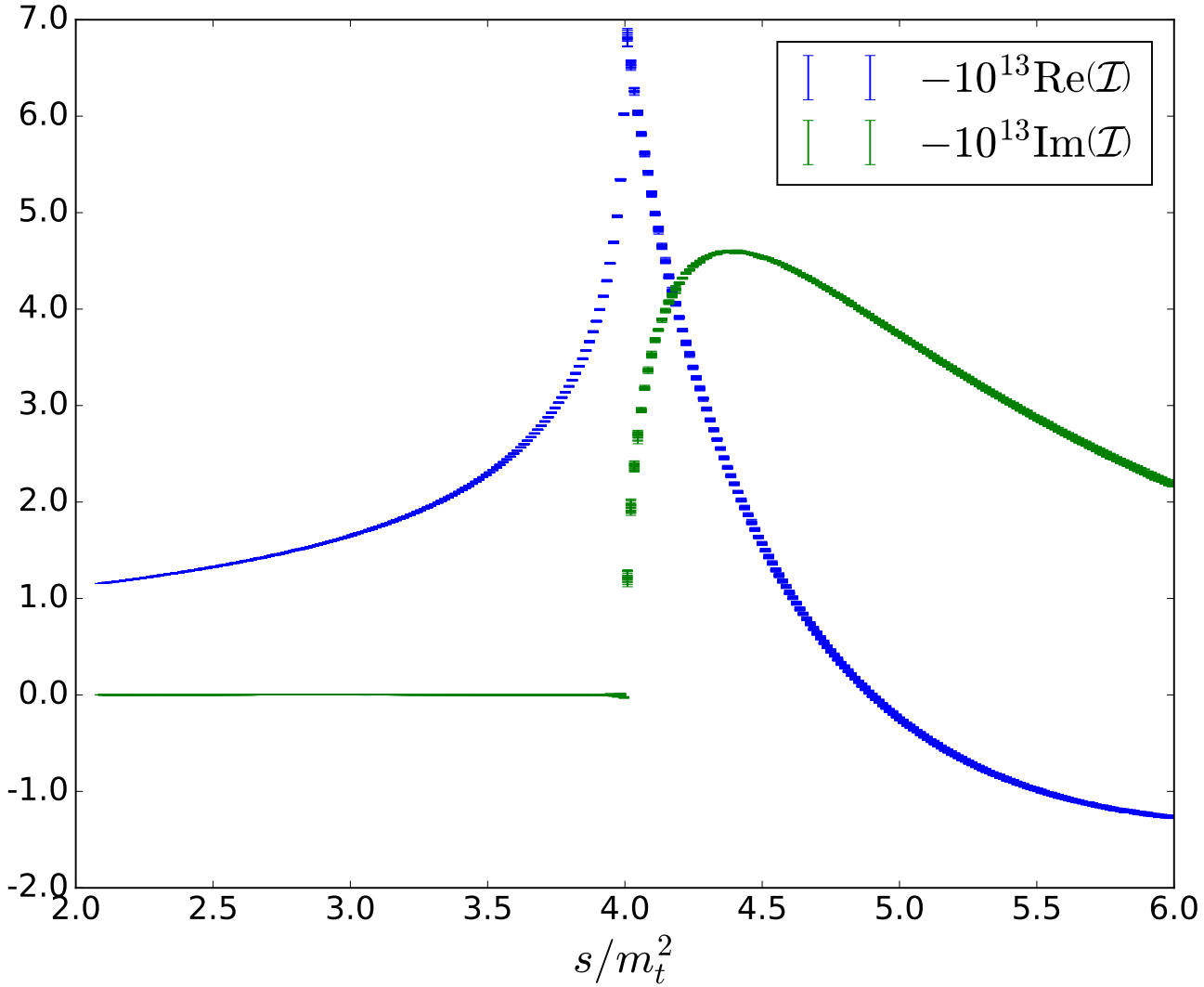
Higgs Boson Pair Production

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



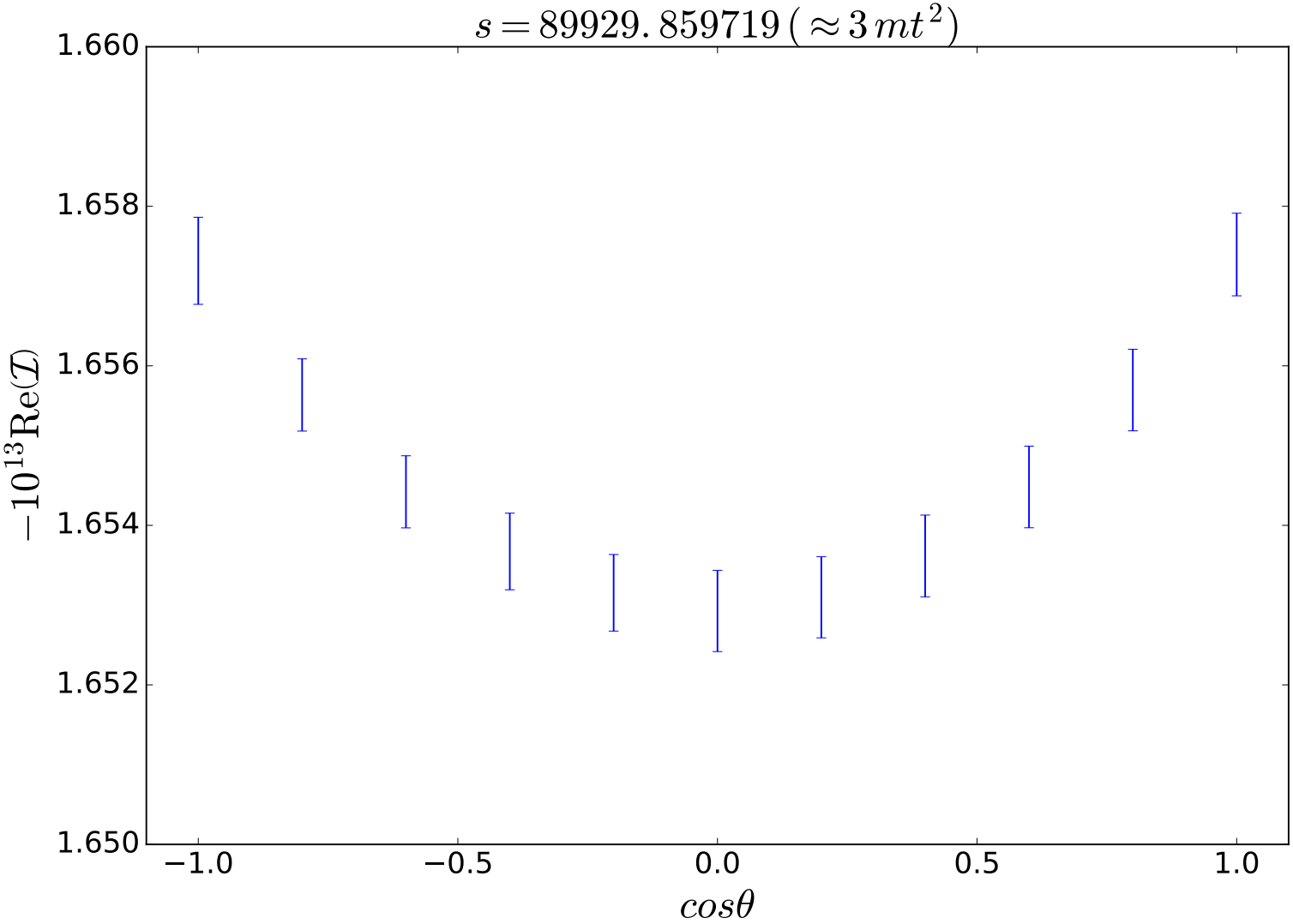
Higgs Boson Pair Production

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



Higgs Boson Pair Production

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



Z Boson Pair Production

G. Heinrich, SJ, S. P. Jones, M. Kerner, J. Pires [1710.06294]

- N-jettiness subtraction scheme

  R. Boughezal et al. [1504.02131], *J. Gaunt et al.* [1505.04794]

- 2 loop virtual amplitude from qqvvamp (massless QCD)

  T. Gehrmann et al. [1503.04812]

- cross check with calculations in q_T-subtraction scheme

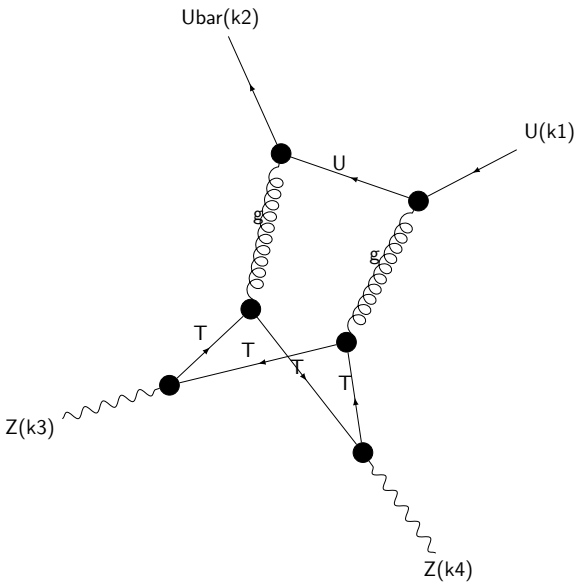
  F. Cascioli et al. [1405.2219], *M. Grazzini et al.* [1507.06257]

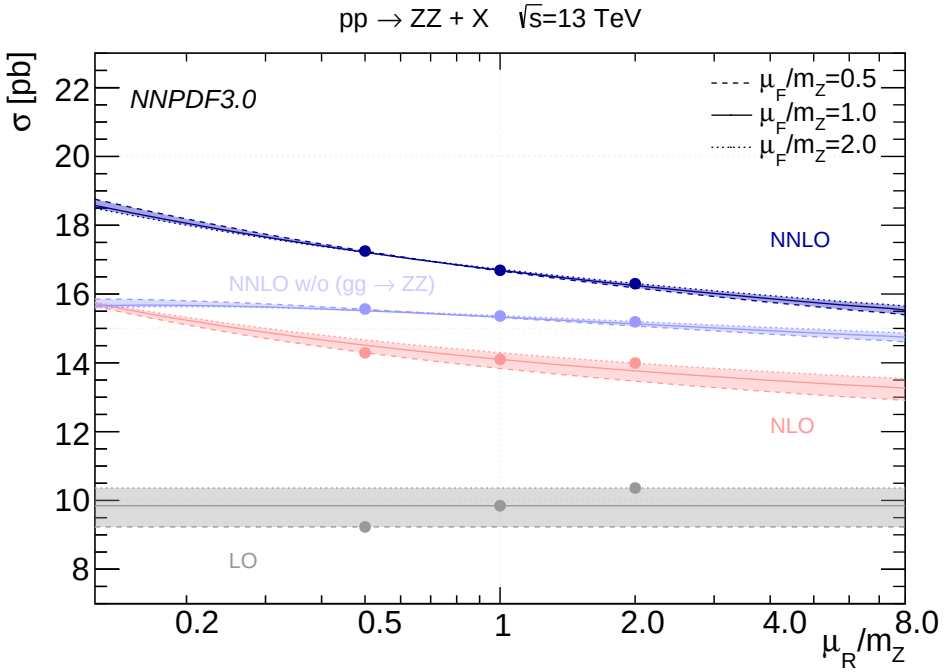
Outlook: top mass effects in pp $\to$ ZZ

Outlook: Top Mass Effects in pp → ZZ

- last missing piece for NNLO(QCD) cross-section: $q\bar{q} \to ZZ$
- largest scale uncertainty from: $gg \to ZZ$ (loop-induced)
- first occurrence of top-quark propagator at two-loop level
- diagrams (thus integrals) similar to those in HH
- automated calculation with GoSam-Xloop
  - Larin-scheme for γ_5 treatment

S. A. Larin [hep-ph/9302240]





Symmetry Finder

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

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

same up to renaming $x \leftrightarrow y$

Symmetry Finder

- $n!$ possible column permutations with n variables
- two optimized algorithms implemented by Ben Ruijl and Stephen Jones:

  B. Ruijl, S. P. Jones [to appear in the proceedings of ACAT 2017]

  - based on permutations as suggested by Alexey Pak

    A. Pak [1111.0868]
  - based on finding graph isomorphisms with `dreadnaut`

    B. D. McKay, A. Piperno [1301.1493]

- outlook: identify matroid symmetries in Loopedia

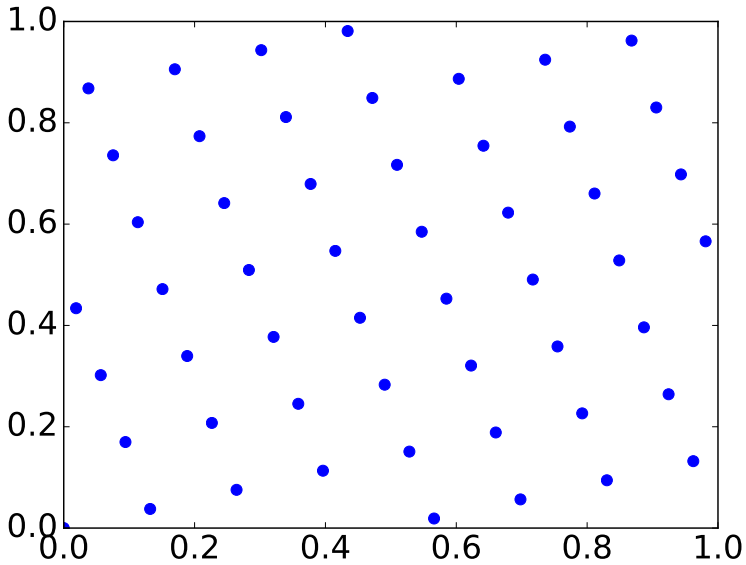
  C. Bogner, S. Borowka, T. Hahn, G. Heinrich, S. P. Jones, M. Kerner, A. von Manteuffel, M. Michel, E. Panzer, V. Papara [1709.01266]

  → talk by *Thomas Hahn*

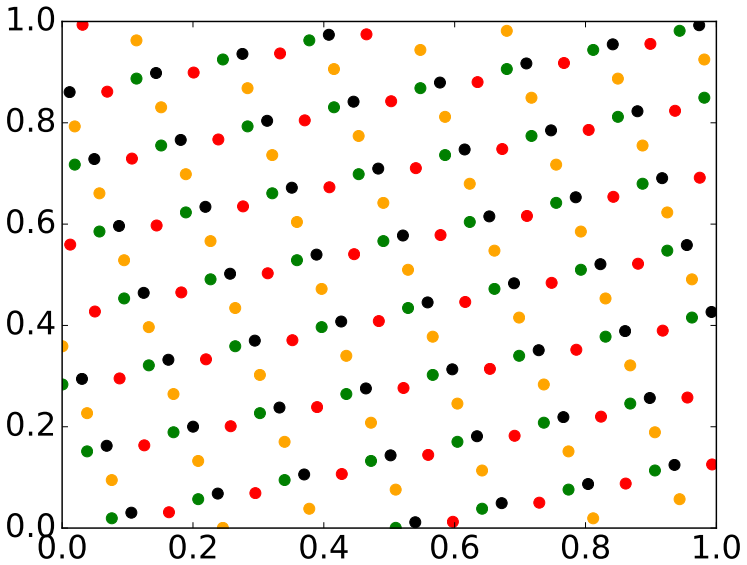
Quasi Monte Carlo (QMC) on GPUs

D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

$$I_s[f] \approx \bar{Q}_{s,n,m}[f] \equiv \frac{1}{m}\sum_{k=0}^{m-1} Q_{s,n}^{(k)}[f], \qquad I_s[f] \equiv \int_{[0,1]^s} \mathrm{d}^s x\, f(\mathbf{x}), \qquad Q_{s,n}^{(k)}[f] \equiv \frac{1}{n}\sum_{i=0}^{n-1} f\left(\left\{\frac{i\mathbf{z}}{n} + \mathbf{\Delta}_k\right\}\right)$$



integral error between
$\sim \frac{1}{n}$ and $\sim \frac{1}{\sqrt{n}}$



determined from
m random shifts

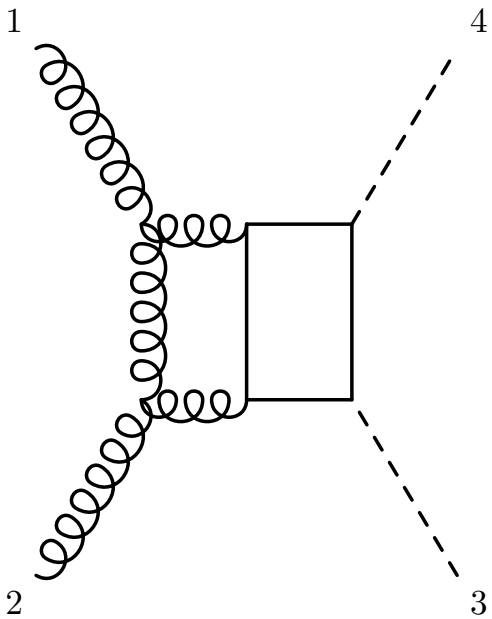
with $\{\cdot\}$ the fractional part, $\mathbf{z}$ the generating vector, $\mathbf{\Delta}_k$ the random shift vector, and f the integrand

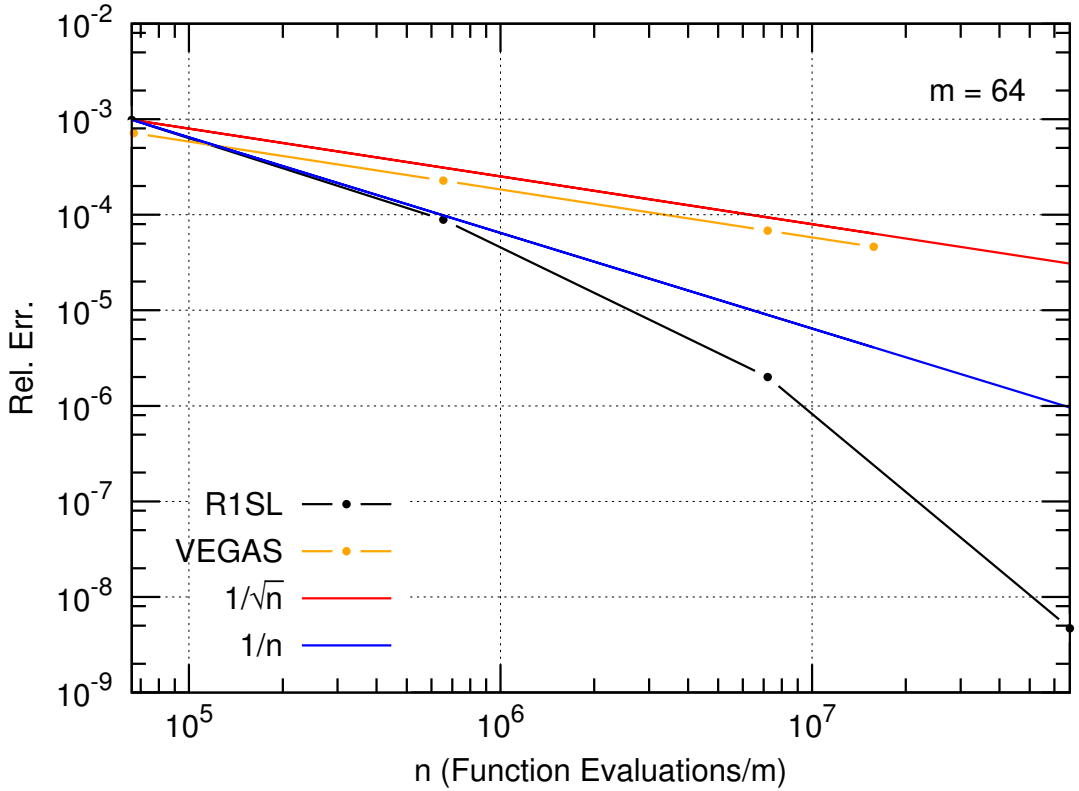
Quasi Monte Carlo (QMC) on GPUs

D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

implementation (within pySecDec and standalone) to be published shortly

figures by Stephen Jones



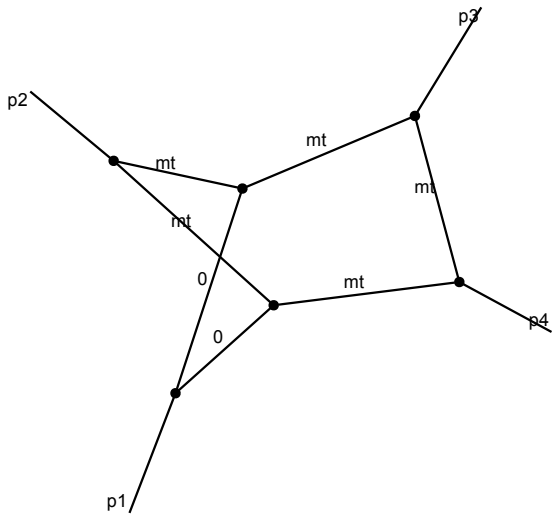


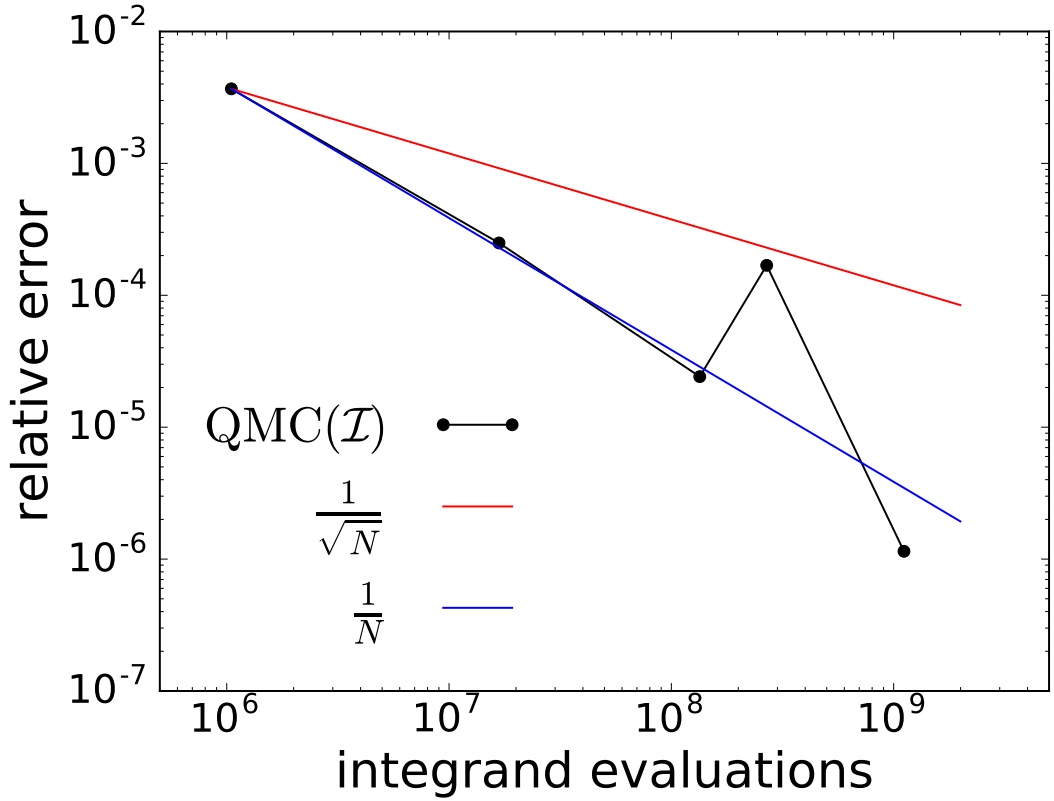
Quasi Monte Carlo (QMC) on GPUs

D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

implementation (within pySecDec and standalone) to be published shortly

- scaling can fluctuate





Quasi Monte Carlo (QMC) on GPUs

D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

implementation (within pySecDec and standalone) to be published shortly

number of samples	algorithm	CPU	GPU	rel. error
$32 \times 16{,}777{,}213$	QMC	19m	2m	3×10^{-4}
$28{,}754{,}314$	Divonne	14m	-	2×10^{-4}
$32 \times 134{,}217{,}689$	QMC	2h 30m	15m	2×10^{-5}
$84{,}536{,}102$	Divonne	45m	-	4×10^{-5}

- CPU: 4 $\times$ Intel Xeon Gold 6140
- GPU: 1 $\times$ Tesla V 100
- comparison against Divonne integrator ("adaptive QMC") from CUBA library

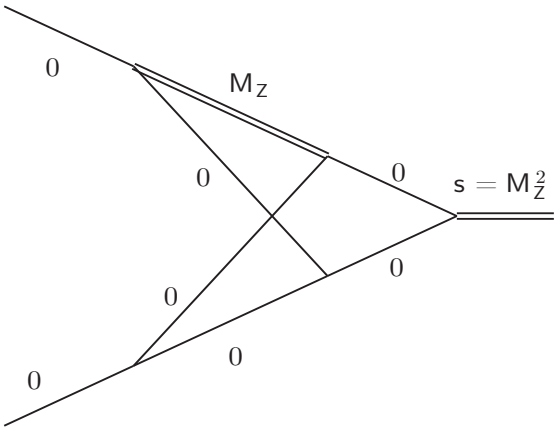
  T. Hahn [hep-ph/0404043]

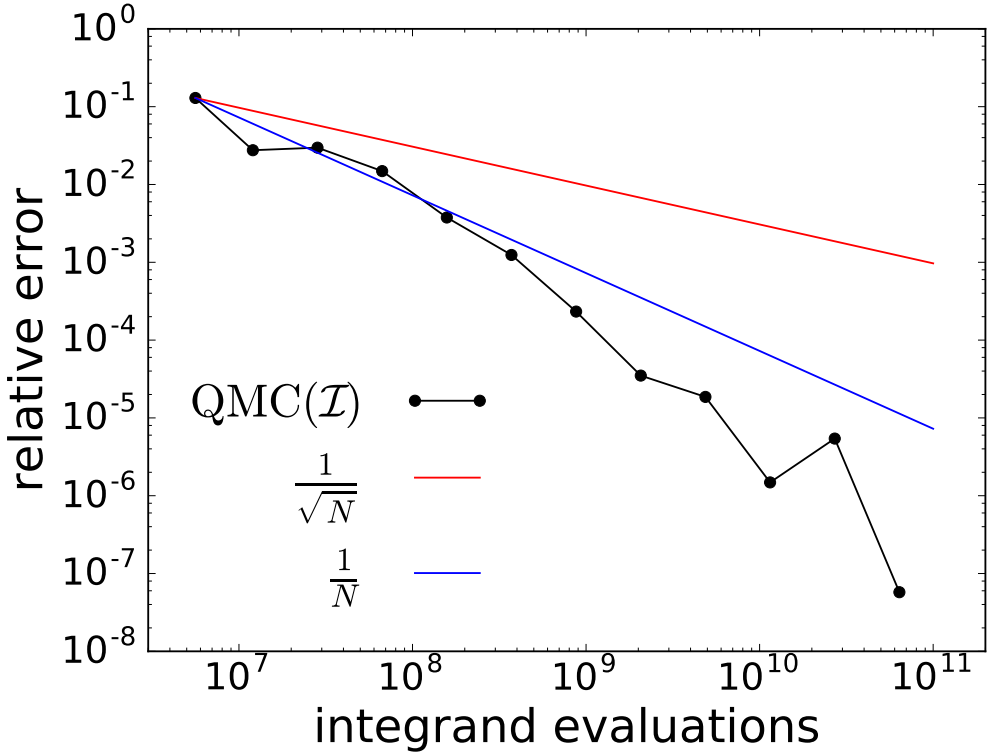
Quasi Monte Carlo (QMC) on GPUs

D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

implementation (within pySecDec and standalone) to be published shortly

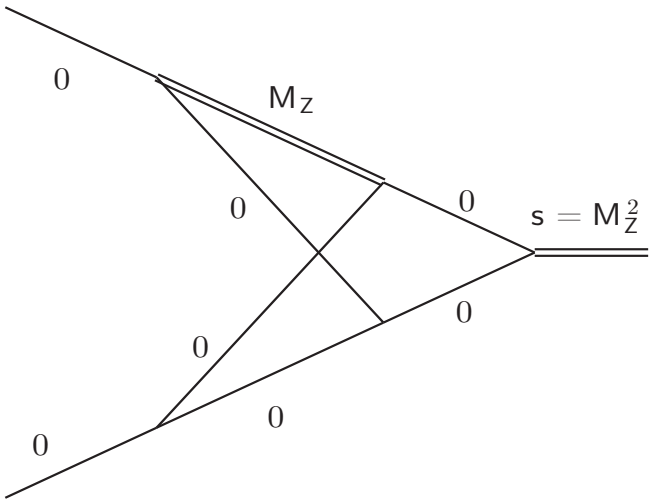
- scaling can fluctuate





2 loop Vertex Diagram

I. Dubovyk, A. Freitas, J. Gluza, T. Riemann, J. Usovitsch [1610.07059]

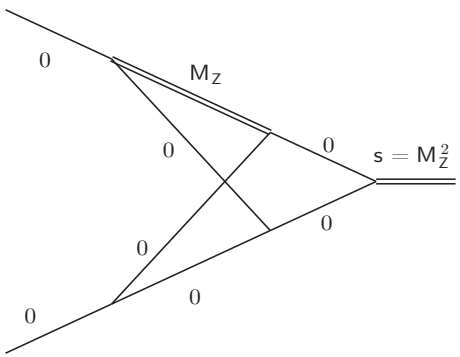


- did not work in SecDec-3
- fixed in pySecDec
- 1 hour integration time on
  4x Intel Xeon Gold 6140
  4x Tesla V 100
  in parallel

```
Numerical Result
eps^-2: 1.23370055013616997 - 2.28018137283890867e-18*I +/- ( 1.53371963073975744e-17 + 1.28037129931719179e-17*I )
eps^-1: 2.89025451824806323 + 3.87578459645767159*I +/- ( 2.12654623448814062e-8 + 1.81276217174111614e-8*I )
eps^0: 0.778599104098807615 + 4.12351279954185834*I +/- ( 5.59405711265899905e-7 + 6.09682594018807232e-7*I )

Reference Result
eps^-2: 1.23370055013617    - 6.20475892887384  * 10^-13 * I
eps^-1: 2.8902545096591976  + 3.875784585038738          * I
eps^0:  0.7785996083247692  + 4.123512600516016          * I
```

2 loop Vertex Diagram

I. Dubovyk, A. Freitas, J. Gluza, T. Riemann, J. Usovitsch [1610.07059]



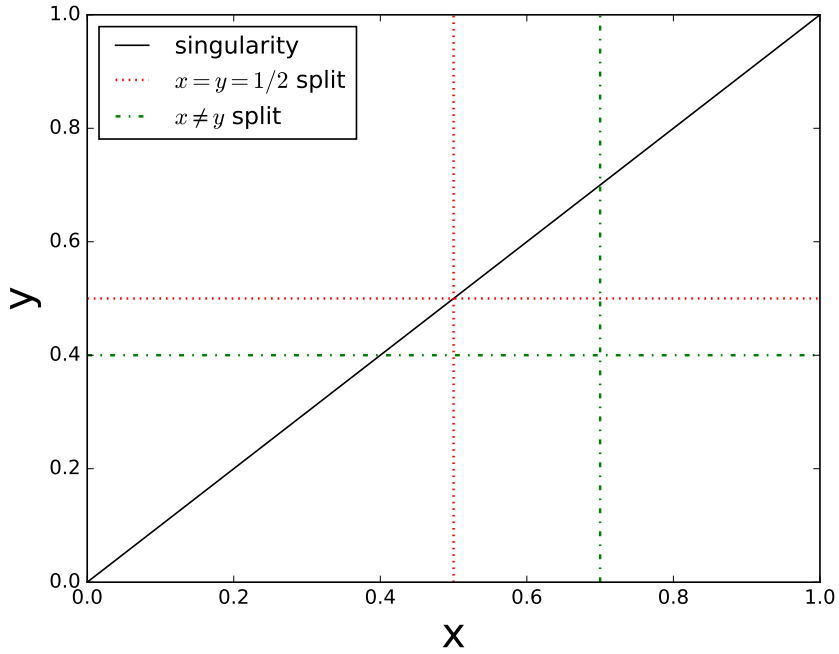
2^{nd} Symanzik polynomial:

$$\begin{aligned}\mathcal{F}/m_Z^2 = {} & x_3^2 x_5 + x_3^2 x_4 + x_2 x_3 x_5 \\ & + x_2 x_3 x_4 + x_1 x_3 x_5 + x_1 x_3 x_4 \\ & + x_1 x_3^2 + x_1 x_2 x_3 + x_0 x_3 x_4 \\ & + x_0 x_3^2 + x_0 x_2 x_3 \\ & - x_1 x_2 x_4 - x_0 x_1 x_5 - x_0 x_1 x_4 \\ & - x_0 x_1 x_2 - x_0 x_1 x_3\end{aligned}$$

- did not work in SecDec-3
- essence of the problem:
  - terms of opposite sign
  - with the same scale

The "x − y" problem

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

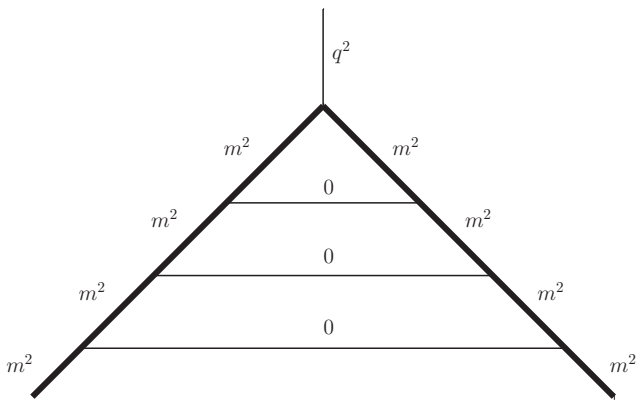


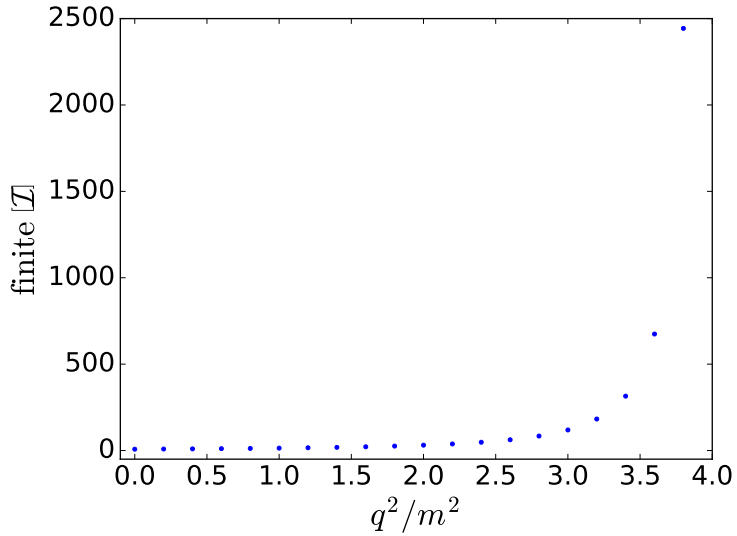
- overlapping singularity along the line $x = y$
- singularity at $x = y = 1$
  - remap singularities at 1 to 0 by split
- SecDec-3 implemented split at $x = y = 1/2$
  - remaps onto itself
- pySecDec splits at $x \neq y$

3 loop Vertex Diagram

J. Ablinger, J. Blümlein, P. Marquard, N. Rana, C. Schneider [1804.07313]

R. N. Lee, A. V. Smirnov, V. A. Smirnov, M. Steinhauser [1804.07310]





- symmetry finder reduces 310 to 155 sectors
- plot above with QMC on 4 GPUs in 15m
  - 5 digits accuracy
  - 3 minutes per point and GPU

Summary

- sector decomposition and pySecDec
- phenomenological applications
  - mass corrections in diboson production
- ongoing development of pySecDec
  - fewer integrands by exploiting symmetries
  - faster numerical integration with QMC on GPUs
  - improved handling of integrals without Euclidean region

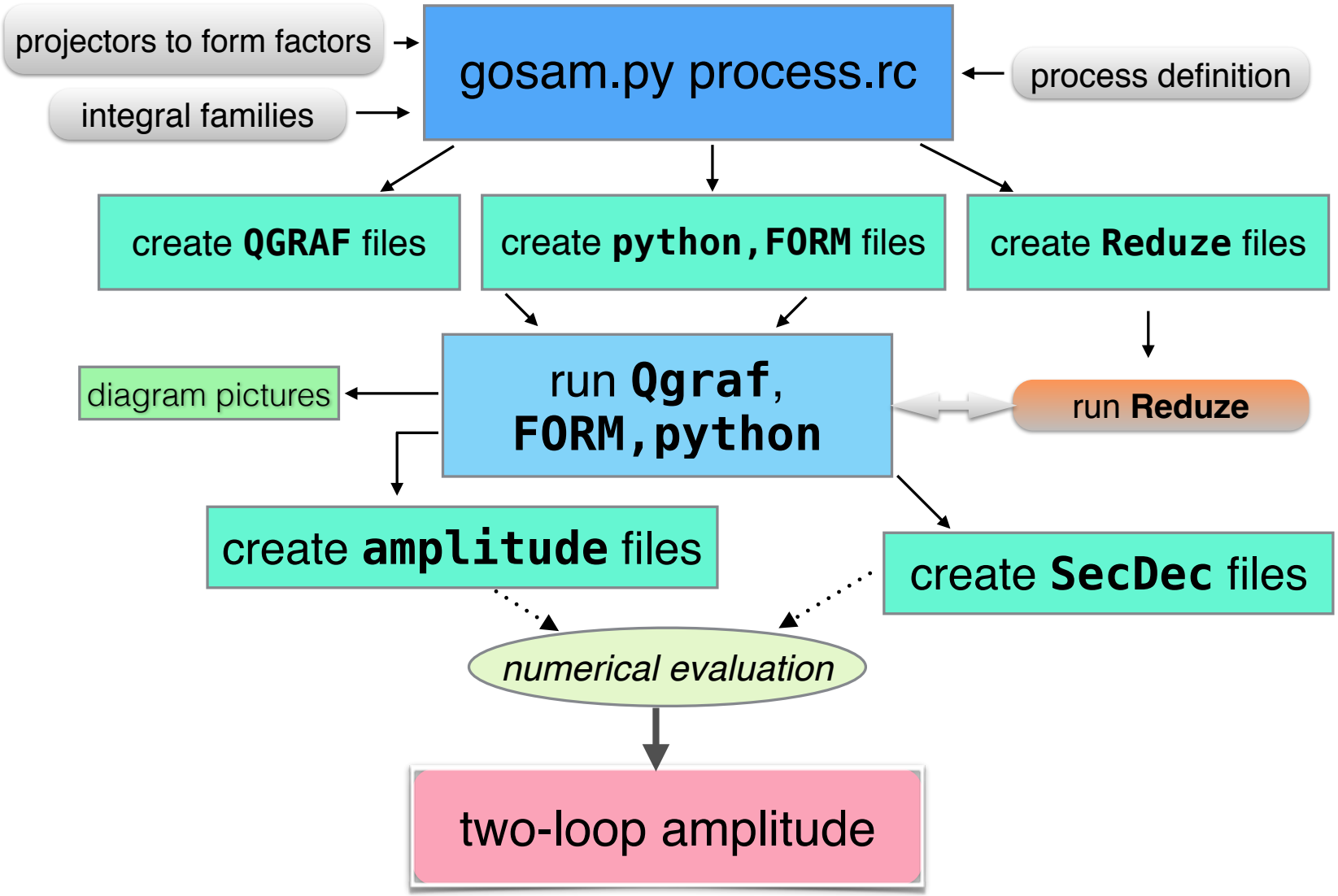
BACKUP

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)

Numerical evaluation of multi-loop integrals



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

```
>>> 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)))))]
```

Basic Usage - Analytical Calculation

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

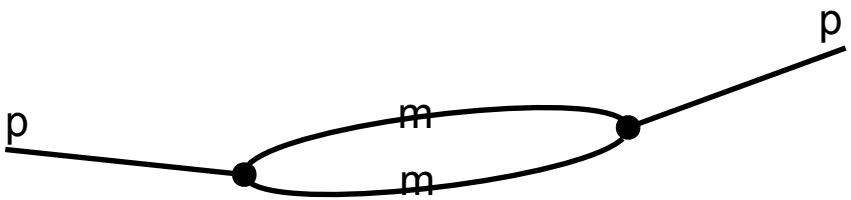
$$= 2 \int_0^1 dx\, x^{-1+\epsilon} \int_0^1 dt\, (1+t)^{-2+\epsilon}$$

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

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

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

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

- 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 $(< momentum >^2 \left[- < mass >^2\right] + \boxed{i\delta})$

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

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

such that

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

Contour Deformation

or: Satisfying the Feynman Prescription

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

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

correct sign if λ_k "small enough"

Symmetry Finder

represent polynomials as integer arrays

$$x + 2\,y = 1 \cdot x^1 \cdot y^0 + 2 \cdot x^0 \cdot y^1 \quad \equiv \quad \begin{array}{c} \text{coefficient} \quad x \quad y \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 2 & 0 & 1 \end{array}\right] \end{array}$$

$$2\,x + y = 2 \cdot x^1 \cdot y^0 + 1 \cdot x^0 \cdot y^1 \quad \equiv \quad \begin{array}{c} \text{coefficient} \quad x \quad y \\ \left[\begin{array}{ccc} 2 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right] \end{array}$$

Symmetry Finder

compare arrays allowing for row- and columnwise permutations

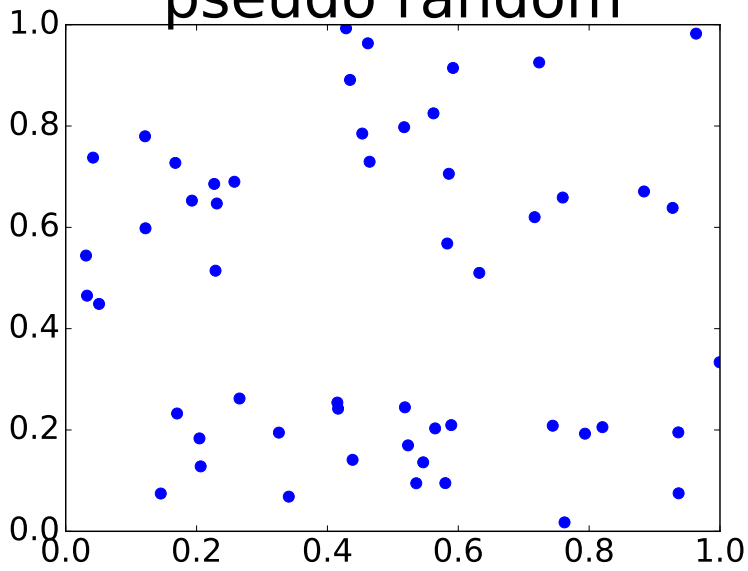
$$\begin{array}{c} \\ \text{term1} \\ \text{term2} \end{array} \begin{array}{c} \begin{array}{ccc} \text{coefficient} & x & y \end{array} \\ \left[\begin{array}{ccc} 2 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right] \end{array} \qquad \begin{array}{c} \\ \text{term2} \\ \text{term1} \end{array} \begin{array}{c} \begin{array}{ccc} \text{coefficient} & y & x \end{array} \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 2 & 0 & 1 \end{array}\right] \end{array}$$

$$2\,x + y \qquad \overset{x \leftrightarrow y}{\longleftrightarrow} \qquad x + 2\,y$$

Quasi Monte Carlo (QMC) on GPUs

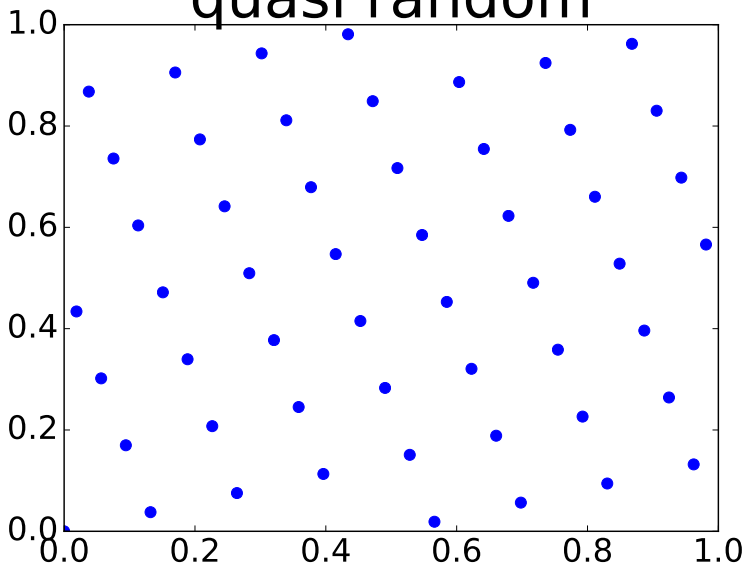
D. Nuyens et al. (2006), *J. Dick et al.* (2013), *Z. Li et al.* [1508.02512]

pseudo random



integral error $\sim \frac{1}{\sqrt{n}}$

quasi random



integral error between $\sim \frac{1}{n}$ and $\sim \frac{1}{\sqrt{n}}$

with n: number of integrand evaluations