

Loop integral evaluation and asymptotic expansion with **pySECDEC**

Vitaly Mageria

*Institute for Theoretical Physics, Karlsruhe Institute of Technology,
Wolfgang-Gaede-Str. 1, Geb. 30.23, 76131 Karlsruhe, Germany*

E-mail: vitalii.maheria@kit.edu

pySECDEC is a computer program that evaluates arbitrary multi-loop Feynman integrals numerically as expansions in the dimensional regulator based on the sector decomposition approach. In the recent release version 1.5 pySECDEC introduces asymptotic expansion in the kinematic ratios using the method of expansion by regions, and an automatic adaptive evaluation of weighted sums of integrals (e.g. amplitudes). In this article we discuss how these features work, and what performance benefits they bring.

*Loops and Legs in Quantum Field Theory - LL2022,
25-30 April, 2022
Ettal, Germany*

1. Introduction

A key part in increasing the accuracy of theoretical predictions to match the experimental accuracy of LHC and beyond is the calculation of higher-loop Feynman integrals. Already at LHC 2-loop QCD corrections are required, with future colliders demanding 3-loop QCD and mixed QCD-electroweak corrections [1], which means that 2- and 3-loop integrals with masses are of practical interest. This poses a big challenge to the analytical methods of calculating loop integrals, and there are classes of integrals that are needed for phenomenological calculations but are not fully known analytically (such as the massive 2-loop 5-point functions).

The proposed solution is to move away from the analytical, and instead use numerical methods such as sector decomposition [2, 3] as implemented in pySecDec [4–6] and FIESTA [7], and Mellin-Barnes integration [8–10], or semi-analytical methods such as solving the differential equations for master integrals numerically [11–13], as implemented in e.g. DIFFEXP or AMFLOW. The limiting factor of the latter family of methods is the need to compute symbolic solutions to integration-by-parts (IBP) relations for the master integrals at least in one variable, whereas the former two approaches do not rely on IBP and instead are limited by the numerical properties of the integrands. In other words, none of these methods can fully replace the others; in fact they can be used complementary, as it is in the case of using sector decomposition to derive boundary conditions for differential equations.

Here we shall concentrate on pySecDec and sector decomposition. pySecDec has a long history; it has started as a Python-based successor to the Mathematica code SECDEC [14–16]. The current pySecDec is written in Python and C++, and is developed on GitHub¹, where one can find the source code, the installation instructions, and the issue tracker. Recently version 1.5 of pySecDec was released bringing a number of usability improvements such as

- simplified installation via the Python package installer² as `pip3 install --user pySecDec`, with all the required dependency software built and installed automatically;
- automatic adjustment of the contour deformation parameters, which means that users no longer need to manually fix “*sign check errors*”;
- automatic adjustment of the `WorkSpace` parameter of FORM [17], which means that users no longer need to manually fix “*workspace overflow*” errors.

More importantly, version 1.5 comes with two major new features targeted at improving pySecDec performance:

- an implementation of the expansion-by-regions method of the asymptotic expansion of integrals in kinematic invariants to help with e.g. high-energy regions;
- adaptive sampling of weighted sums of integrals (i.e. amplitudes).

A more detailed description and usage examples can be found in [6]; here we shall only briefly describe how these features work, and what performance benefits should be expected from them.

¹<https://github.com/gudrunhe/secdec>

²<https://pypi.org/project/pySecDec>

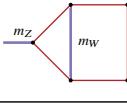
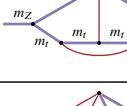
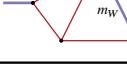
Diagram \ Relative precision		10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
	GPU	15s	20s	40s	200s	13m	50m
	CPU	10s	50s	400s	4000s	180m	1200m
	GPU	18s	19s	30s	20s	1.2m	2m
	CPU	5s	14s	60s	50s	12m	16m
	GPU	6s	11s	12s	30s	3m	24m
	CPU	5s	10s	50s	800s	60m	800m

Table 1: pySecDec 1.5.3 integration time with the QMC integrator for a few massive 3-loop electroweak self-energy integrals taken from [18]. The CPU is AMD Epyc 7302 with 32 threads; the GPU is NVidia A100.

2. The expected performance

When used in the recommended configuration pySecDec can deliver precision sufficient for practical purposes in seconds to minutes; for example see the integration times for a selection of 3-loop massive integrals in [Table 1](#). This recommended configuration consists of:

1. Using the QMC integrator [5], as opposed to the more well known integrators like Vegas (which pySecDec also supports). This is because QMC implements the Randomized Quasi Monte-Carlo integration using rank-1 lattice rules [19] specifically constructed so that if the integrand is smooth enough then the precision of the result is guaranteed to scale with the number of integrand evaluations as $1/N^2$, while the classical Monte-Carlo techniques, even the advanced ones such as Vegas, scale as $1/\sqrt{N}$. See [Figure 1](#) for a comparison of the scaling for an example integral. Note that the scaling QMC achieves is not exactly $1/N^2$: in practice depending on the integrand we see everything from $1/N$ to $1/N^3$ (as it is for e.g. the second integral in [Table 1](#)).
2. Using a GPU, preferably a server-grade one. Our testing shows that a single server-grade CPU (e.g. a 32-thread AMD Epyc 7302) is typically as powerful as a top consumer-grade GPU (e.g. an NVidia RTX 2080 Ti), and a server-grade GPU (e.g. NVidia A100) is 10x faster than that, so GPUs provide a more cost-effective way of running numerical integration.

Even in the recommended configuration, some integrals will converge slowly. See for example [Table 2](#): four very similar integrals have wildly different integration times. For this reason we advise spending additional time on evaluating the choice of the integrals passed to pySecDec.

While we lack a general recipe, we have found that integrals with a negative power of the U polynomial in the Feynman parametrization converge especially slowly; same for those with the F polynomial power lower than -2 . If possible these should be avoided, and quasi-finite integrals should be preferred (see e.g. [20]).

A different source of performance problems are extreme kinematics: if an integral depends on multiple scale ratios, the more extreme (i.e. big or small) the ratios get, the longer the integration time will become. See [Figure 2](#) for an illustration. The reason is that in this case the majority of the

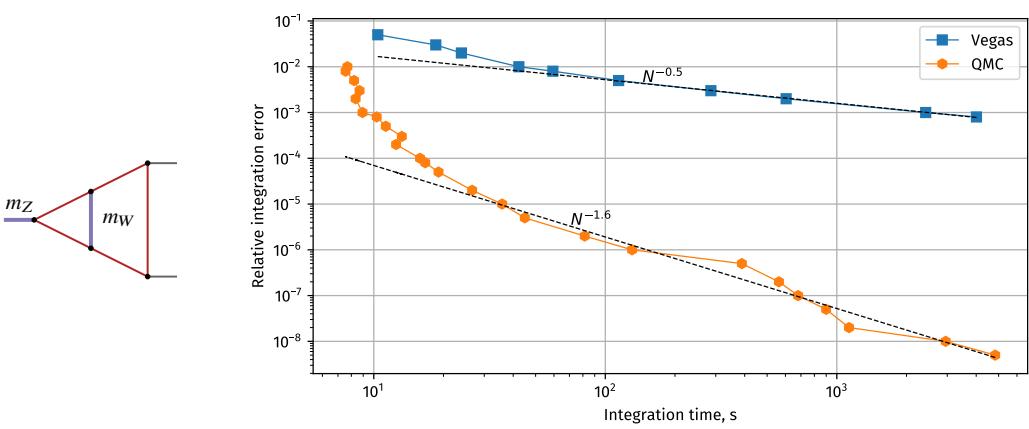


Figure 1: Integration time scaling for the depicted integral using Monte Carlo (Vegas) and Randomized Quasi Monte Carlo (QMC) with *pySecDEC* version 1.5.3 on an NVidia A100 GPU.

Integral	Expansion orders	Integration time
	$\epsilon^{-3}, \dots, \epsilon^0$	27s
	$\epsilon^{-2}, \dots, \epsilon^0$	57s
	$\epsilon^{-2}, \dots, \epsilon^0$	1230s
	$\epsilon^{-2}, \dots, \epsilon^0$	>9000s

Table 2: Integration time (to 10^{-3} precision on an NVidia A100 GPU with *pySecDEC* 1.5.3) of a set of similar integrals, different only in the position of the squared (dotted) propagator.

integral's value becomes progressively concentrated in a smaller region of the integration space, and more integrand evaluations are needed to sample this small space precisely. To solve this, one can try to series-expand the integrand in the extreme ratio, taking it out from the integrand. This brings us to the first major new feature of *pySecDEC* 1.5: asymptotic expansion.

3. Asymptotic expansion

Expanding an integral in a small parameter may not be as simple as Taylor-expanding the integrand. For example, consider the following integral dependent on a small parameter t and some constants a, b, c, d, α :

$$I = \int_0^1 P^\alpha(x) dx, \quad P \equiv ax^3 + btx + ct^3 + dt^3x^2. \quad (1)$$

Representing P simply as $ax^3 \left(1 + \frac{bx+ct^2+dt^2x^2}{ax^3}t\right)$ and series-expanding the bracket is not a valid procedure in the whole integration space, because in the region where $x \ll t$ the value of

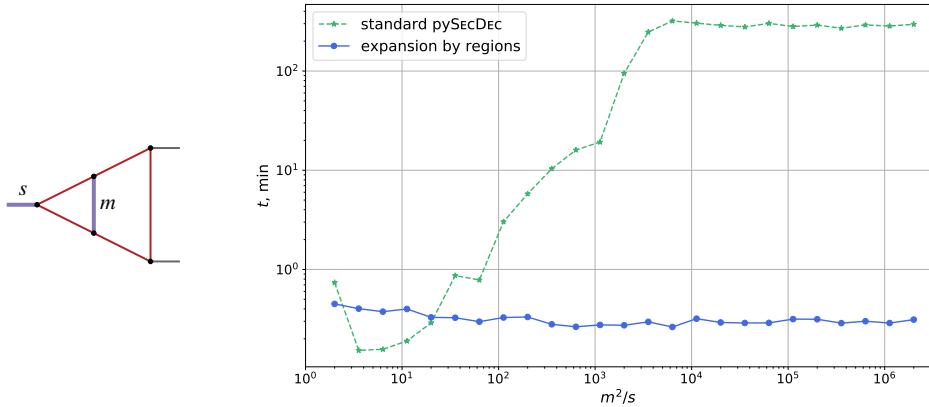


Figure 2: Time to evaluate the depicted integral to 3 digits of precision depending on the m^2/s ratio. The integration time is capped at 5 hours, and when $m^2/s > 3000$ the precision target can not be reached in time.

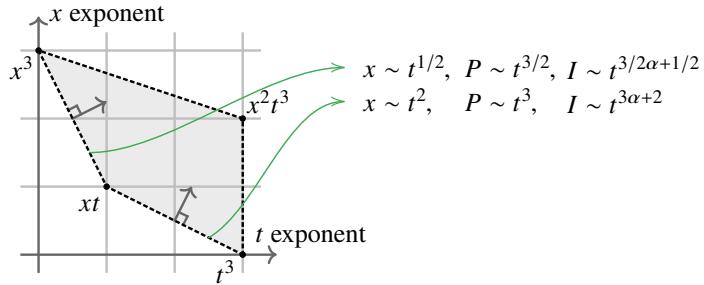


Figure 3: The Newton polytope of the polynomial P from eq. (1).

$\frac{bx+ct^2+dt^2x^2}{ax^3}t$ is not small. Instead we can split the integration space into a set of regions such that the integration variables are of the order of a particular power of the small parameter t in each region. Once this is done, figuring out which terms of P are small in each regions is easy, and the corresponding Taylor expansions are valid.

This method is *expansion by regions* [21]. A geometric formulation of it is given in [22]: if we plot the monomials of P and form its Newton polytope as in Figure 3, then each facet of the polytope pointing in a positive direction defines a region where expansion must be performed. For example, for I we find two required regions:

1. A region where $x \sim t^{1/2}$, and P can be expanded as $(ax^3 + btx) \left(1 + \frac{ct^3 + dt^3x^2}{ax^3 + btx}\right)$.
2. A region where $x \sim t^2$, and P can be expanded as $(btx + ct^3) \left(1 + \frac{ax^3 + dt^3x^2}{btx + ct^3}\right)$.

As explained in [22], all other possible regions can be ignored, as their contribution turns out to be zero. In fact, after we have determined the regions and Taylor-expanded in each, we can forget that the integration space was split into regions, and integrate each expansion in the whole integration space: the contribution of the additional integration space parts also turn out to be zero.

Note that in the above example Taylor expansion in the first region is only valid if $a, b > 0$, and in the second only if $b, c > 0$: otherwise there are values of x for which the terms after 1+ are not small. In general expansion by regions only works if the coefficients in front of the monomials of P

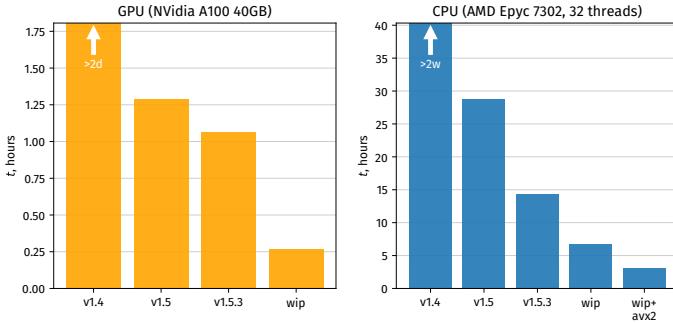


Figure 4: Integration time of the first diagram from Table 1 to 7 digits of precision by pySECDEC version. Here `wip` stands for the work-in-progress code to be released in the future, and `avx2` stands for compilation with AVX2 and FMA processor instruction sets allowed (the `wip` code has special provisions to use them).

are positive. This limits the applicability of the method a bit; in practice this usually means that an expansion is possible in a high-energy region, but may not be possible around a threshold.

Expansion by regions was previously implemented in `ASY.M` [23] and `ASY2.M` [24], which is currently a part of FIESTA. As of version 1.5 pySECDEC provides the function `loop_regions()` that expands a given loop integral by regions; the result of it can be directly turned into a standard pySECDEC integration library, or inspected for further processing.

4. Adaptive sampling of amplitudes

The other major new improvement in pySECDEC 1.5 is the adaptive sampling of weighted sums of integrals (i.e. amplitudes). The basic idea is simple: if one wants to evaluate a sum like

$$1000 \text{---} \textcolor{red}{\circlearrowleft} + 10 \text{---} \textcolor{red}{\diamond} + 1 \text{---} \textcolor{red}{\circlearrowright}, \quad (2)$$

then it makes sense to spend more time on sampling the first integral as it has the largest contribution to the sum, and spend little time on the last one: this way more precision for the sum can be achieved within the same integration time.

This is why pySECDEC 1.5 comes with a function `sum_package()` that takes a list of integrals and a matrix of coefficients, and produces an integration library to integrate the specified weighted sums. During the integration the optimal sampling distribution will then be automatically determined based on how big the coefficients of the integrals are, how well they converge, and how fast their integrands can be evaluated. Moreover, by default pySECDEC uses the same technique for single integrals too: because during sector decomposition an integral is split into a sum of sectors, the evaluation of this sum can be similarly optimized.

To see how much improvement this makes, take a look at the performance progression of pySECDEC versions depicted in Figure 4. The big jump from version 1.4 to 1.5 is mainly due to adaptive sum sampling—even though only a single integral is evaluated here.

Note that this technology is not new, and pySECDEC equipped with it has previously been successfully used in multiple 2-loop calculation such as [25–27].

5. Conclusions

The recently released pySecDec version 1.5 comes with two major new features: an implementation of asymptotic expansion of integrals and automatic adaptive sampling of the weighted sums of integrals (i.e. amplitudes). The first provides an essential tool in handling extreme kinematics (e.g. high-energy regions). The second one brings a major speedup in the evaluation of single integrals and allows using pySecDec to evaluate whole amplitudes optimally.

Following the release, the pySecDec team continues incrementally improving its performance and usability, hoping to make it applicable to even more challenging integrals, and to establish it as a tool for evaluations of whole amplitudes.

Acknowledgements

This research was supported in part by the COST Action CA16201 (“Particleface”) of the European Union and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 396021762 (TRR 257).

References

- [1] A. Freitas, *Theory Needs for Future e^+e^- Colliders*, *Acta Phys. Polon. B* **52** (2021) 929.
- [2] T. Binoth and G. Heinrich, *An automatized algorithm to compute infrared divergent multiloop integrals*, *Nucl. Phys. B* **585** (2000) 741 [[hep-ph/0004013](#)].
- [3] G. Heinrich, *Sector Decomposition*, *Int. J. Mod. Phys. A* **23** (2008) 1457 [[0803.4177](#)].
- [4] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones et al., *pySecDec: a toolbox for the numerical evaluation of multi-scale integrals*, *Comput. Phys. Commun.* **222** (2018) 313 [[1703.09692](#)].
- [5] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones et al., *A GPU compatible quasi-Monte Carlo integrator interfaced to pySecDec*, *Comput. Phys. Commun.* **240** (2019) 120 [[1811.11720](#)].
- [6] G. Heinrich, S. Jahn, S.P. Jones, M. Kerner et al., *Expansion by regions with pySecDec*, *Comput. Phys. Commun.* **273** (2022) 108267 [[2108.10807](#)].
- [7] A.V. Smirnov, N.D. Shapurov and L.I. Vysotsky, *FIESA5: numerical high-performance Feynman integral evaluation*, [2110.11660](#).
- [8] M. Czakon, *Automatized analytic continuation of Mellin-Barnes integrals*, *Comput. Phys. Commun.* **175** (2006) 559 [[hep-ph/0511200](#)].
- [9] J. Gluza, K. Kajda and T. Riemann, *AMBRE: A Mathematica package for the construction of Mellin-Barnes representations for Feynman integrals*, *Comput. Phys. Commun.* **177** (2007) 879 [[0704.2423](#)].
- [10] J. Usovitsch, I. Dubovyk and T. Riemann, *MBnumerics: Numerical integration of Mellin-Barnes integrals in physical regions*, *PoS LL2018* (2018) 046 [[1810.04580](#)].

- [11] M. Hidding, *DiffExp*, a *Mathematica* package for computing Feynman integrals in terms of one-dimensional series expansions, *Comput. Phys. Commun.* **269** (2021) 108125 [[2006.05510](#)].
- [12] X. Liu and Y.-Q. Ma, *AMFlow*: a *Mathematica* package for Feynman integrals computation via Auxiliary Mass Flow, [2201.11669](#).
- [13] M. Hidding and J. Usovitsch, Feynman parameter integration through differential equations, [2206.14790](#).
- [14] J. Carter and G. Heinrich, *SecDec*: A general program for sector decomposition, *Comput. Phys. Commun.* **182** (2011) 1566 [[1011.5493](#)].
- [15] S. Borowka, J. Carter and G. Heinrich, Numerical Evaluation of Multi-Loop Integrals for Arbitrary Kinematics with *SecDec* 2.0, *Comput. Phys. Commun.* **184** (2013) 396.
- [16] S. Borowka, G. Heinrich, S.P. Jones, M. Kerner et al., *SecDec-3.0: numerical evaluation of multi-scale integrals beyond one loop*, *Comput. Phys. Commun.* **196** (2015) 470.
- [17] B. Ruijl, T. Ueda and J. Vermaseren, *FORM* version 4.2, [1707.06453](#).
- [18] I. Dubovyk, J. Usovitsch and K. Grzanka, Toward Three-Loop Feynman Massive Diagram Calculations, *Symmetry* **13** (2021) 975.
- [19] J. Dick, F.Y. Kuo and I.H. Sloan, High-dimensional integration: The quasi-monte carlo way, *Acta Numerica* **22** (2013) 133–288.
- [20] A. von Manteuffel, E. Panzer and R.M. Schabinger, A quasi-finite basis for multi-loop Feynman integrals, *JHEP* **02** (2015) 120 [[1411.7392](#)].
- [21] M. Beneke and V.A. Smirnov, Asymptotic expansion of Feynman integrals near threshold, *Nucl. Phys. B* **522** (1998) 321 [[hep-ph/9711391](#)].
- [22] B. Jantzen, Foundation and generalization of the expansion by regions, *JHEP* **12** (2011) 076 [[1111.2589](#)].
- [23] A. Pak and A. Smirnov, Geometric approach to asymptotic expansion of Feynman integrals, *Eur. Phys. J. C* **71** (2011) 1626 [[1011.4863](#)].
- [24] B. Jantzen, A.V. Smirnov and V.A. Smirnov, Expansion by regions: revealing potential and Glauber regions automatically, *Eur. Phys. J. C* **72** (2012) 2139 [[1206.0546](#)].
- [25] L. Chen, G. Heinrich, S.P. Jones, M. Kerner et al., ZH production in gluon fusion: two-loop amplitudes with full top quark mass dependence, *JHEP* **03** (2021) 125 [[2011.12325](#)].
- [26] L. Chen, G. Heinrich, S. Jahn, S.P. Jones et al., Photon pair production in gluon fusion: Top quark effects at NLO with threshold matching, *JHEP* **04** (2020) 115 [[1911.09314](#)].
- [27] S.P. Jones, M. Kerner and G. Luisoni, Next-to-Leading-Order QCD Corrections to Higgs Boson Plus Jet Production with Full Top-Quark Mass Dependence, *Phys. Rev. Lett.* **120** (2018) 162001 [[1802.00349](#)].