SecDec: A tool for numerical multi-loop/leg calculations

Jonathon Carter[1]
Institute for Particle Physics Phenomenology, University of Durham, UK

Sophia Borowka, Gudrun Heinrich
Max-Planck Institute for Physics, Munich, Germany

Abstract. The new version of the program SecDec is described, which can be used for the extraction of poles within dimensional regularisation from multi-loop integrals as well as phase space integrals. The numerical evaluation of the resulting finite functions is also done by the program in an automated way, with no restriction on the kinematics in the case of loop integrals.

1. Introduction

Nowadays we are in the long awaited situation to be confronted with a wealth of high energy collider physics data, enabling us to explore physics at the TeV scale. However, for the analysis and interpretation of these data, precise theory predictions are mandatory. In most cases, this means that calculations beyond the leading order in perturbation theory are necessary. Such calculations either involve integrations over loop momenta for the virtual corrections, or integrations over phase spaces for the real radiation corrections. In both cases, multi-dimensional parameter integrals need to be evaluated, which can contain ultraviolet singularities and, in the presence of massless particles, contain soft and/or collinear singularities. These singularities, if regulated by dimensional regularisation, appear as poles in $1/\epsilon$, but factorising the poles from complicated multi-loop or multi-parameter integrals is a highly non-trivial task. The program SecDec, presented in [1], performs this task in an automated way, based on the algorithm of sector decomposition [2, 3, 4]. Other implementations of sector decomposition into public programs can be found in [5, 6, 7]. The method already has been applied in various calculations, for a review see e.g. [8], and combined with other techniques like non-linear transformations [9] or the use of the Bernstein-Tkachov theorem [10, 11, 12]. In this talk the emphasis is on the presentation of the new features of the program SecDec.

[1] Speaker; presented at the conference ACAT 2011, Uxbridge, London, UK, September 2011.

2. The Algorithm

2.1. General framework

The decomposition algorithm is described in detail in [8], here we only describe the basic concepts. Consider an N-dimensional parameter integral, for example

$$I_N = \int_0^1 dx_1 \ldots \int_0^1 dx_N \, x_1^{-1-\epsilon} \, \frac{g_3(\vec{x})}{[x_1 \, g_1(\vec{x}) + x_2 \, g_2(\vec{x})]} \, , \tag{1}$$

where g_1, g_2 are polynomial functions which do not vanish for $x_1, x_2 \to 0$. However, we would like to factorize the integrand such that the term in square brackets is non-vanishing in the limit $x_1 \to 0, x_2 \to 0$. This can be achieved by a decomposition into sectors where x_1 and x_2 are ordered: we multiply eq. (1) with $1 = \Theta(x_1 - x_2) + \Theta(x_2 - x_1)$ and substitute $x_2 \to x_1 \, x_2$ in the first sector and $x_1 \to x_2 \, x_1$ in the second sector, to arrive at

$$I_N = \int_0^1 dx_1 \ldots \int_0^1 dx_N \, x_1^{-1-\epsilon} \left\{ \frac{\tilde{g}_3}{[\tilde{g}_1(\vec{x}) + x_2 \, \tilde{g}_2(\vec{x})]} + x_2^{-1-\epsilon} \, \frac{\bar{g}_3}{[x_1 \, \bar{g}_1(\vec{x}) + \bar{g}_2(\vec{x})]} \right\} \, , \tag{2}$$

where $\tilde{g}_i, \bar{g}_i$ are functions of the transformed variables. As g_1, g_2 are nonvanishing at the origin, the terms in square brackets cannot lead to singularities anymore; the singularities are factored out into the monomials of type $x_i^{-1-\epsilon}$ instead, and subtraction of the poles and expansion in ϵ are straightforward in this form. In more complicated cases, the decomposition procedure may have to be iterated to achieve a full factorisation. A detailed description can be found e.g. in [2, 8].

2.2. Loop integrals

A scalar Feynman integral G in D dimensions at L loops with N propagators, where the propagators can have arbitrary, not necessarily integer powers ν_j, has the following representation in momentum space:

$$\begin{aligned} G &= \int \prod_{l=1}^{L} \mathrm{d}^D \kappa_l \, \frac{1}{\prod\limits_{j=1}^{N} P_j^{\nu_j}(\{k\},\{p\},m_j^2)} \\ \mathrm{d}^D \kappa_l &= \frac{\mu^{4-D}}{i\pi^{\frac{D}{2}}} \, \mathrm{d}^D k_l \, , \; P_j(\{k\},\{p\},m_j^2) = q_j^2 - m_j^2 + i\delta \, , \end{aligned} \tag{3}$$

where the q_j are linear combinations of external momenta p_i and loop momenta k_l. Introducing Feynman parameters leads to

$$G = \frac{(-1)^{N_\nu}}{\prod_{j=1}^{N} \Gamma(\nu_j)} \Gamma(N_\nu - LD/2) \int_0^\infty \prod_{j=1}^{N} dx_j \; x_j^{\nu_j - 1} \, \delta(1 - \sum_{l=1}^{N} x_l) \frac{\mathcal{U}^{N_\nu - (L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}} \, , \; N_\nu = \sum_{j=1}^{N} \nu_j \, , \tag{4}$$

where $\mathcal{U}$ is a polynomial of degree L in the Feynman parameters, while $\mathcal{F}$ is of degree $L+1$ in the Feynman parameters, and also contains the Lorentz invariants which can be formed from the external momenta of the diagram, as well as propagator masses. As a simple example, consider the function $\mathcal{F}$ for a massles one-loop box diagram:

$$\mathcal{F} = -s_{12} \, x_2 x_4 - s_{23} \, x_1 x_3 - i \, \delta \, . \tag{5}$$

The functions $\mathcal{U}$ and $\mathcal{F}$ can also be constructed directly from the topology of the corresponding Feynman graph [13, 14], and the implementation of this construction in SecDec version 2.0 is one of the new features of the program.

For a diagram with massless propagators, none of the Feynman parameters occurs quadratically in the function $\mathcal{F} = \mathcal{F}_0$. If massive internal lines are present, $\mathcal{F}$ gets an additional term $\mathcal{F}(\vec{x}) = \mathcal{F}_0(\vec{x}) + \mathcal{U}(\vec{x}) \sum_{j=1}^{N} x_j m_j^2$. $\mathcal{U}$ is a positive semi-definite function. A vanishing $\mathcal{U}$ function is related to the UV subdivergences of the graph. In the region where all invariants formed from external momenta are negative, which we will call the *Euclidean region* in the following, $\mathcal{F}$ is also a positive semi-definite function. Its vanishing does not necessarily lead to an IR singularity. Only if some of the invariants are zero, for example if some of the external momenta are light-like, the vanishing of $\mathcal{F}$ may induce an IR divergence. Thus it depends on the *kinematics* and not only on the topology (like in the UV case) whether a zero of $\mathcal{F}$ leads to a divergence or not. The necessary (but not sufficient) conditions for a divergence are given by the Landau equations [15, 16, 13]:

$$x_j \left(q_j^2 - m_j^2\right) = 0 \quad \forall\, j \tag{6}$$

$$\frac{\partial}{\partial k_l^\mu} \sum_j x_j \left(q_j^2(k,p) - m_j^2\right) = 0 \,. \tag{7}$$

If all kinematic invariants formed by external momenta are of the same sign, the necessary condition $\mathcal{F} = 0$ for an IR divergence can only be fulfilled if some of the parameters x_i go to zero. These singularities can be regulated by dimensional regularisation and factored out of the function $\mathcal{F}$ using sector decomposition. The same holds for dimensionally regulated UV singularities contained in $\mathcal{U}$. However, after these singularities, appearing as poles in $1/\epsilon$, have been extracted using sector decomposition, for non-Euclidean kinematics we are still faced with integrable singularities related to kinematic thresholds. How we deal with these singularities will be described in Section 4.4.

2.3. General parameter integrals

The program can also deal with parameter integrals which are more general than the ones related to multi-loop integrals, for example phase space integrals involving massless particles, where the regions in phase space corresponding to soft and/or collinear configurations lead to singularites which can be extracted as poles in $1/\epsilon$. The only restrictions are that the integration domain should be the unit hypercube, and the singularities should be only endpoint singularities, i.e. should be located at zero or one. We assume that the singularities are regulated by non-integer powers of the integration parameters, where the non-integer part is the ϵ of dimensional regularisation or some other regulator. The general form of the integrals is

$$I = \int_0^1 dx_1 \ldots \int_0^1 dx_N \prod_{i=1}^{m} P_i(\vec{x}, \{\alpha\})^{\nu_i} \,, \tag{8}$$

where $P_i(\vec{x}, \{\alpha\})$ are polynomial functions of the parameters x_j, which can also contain some symbolic constants $\{\alpha\}$. The user can leave the parameters $\{\alpha\}$ symbolic during the decomposition, specifying numerical values only for the numerical integration step. This way the decomposition and subtraction steps do not have to be redone if the values for the constants are changed. The ν_i are powers of the form $\nu_i = a_i + b_i\epsilon$ (with a_i such that the integral is convergent). Note that half integer powers are also possible.

3. The SecDec program

The program consists of two parts, an algebraic part and a numerical part. The algebraic part uses code written in Mathematica [17] and does the decomposition into sectors, the subtraction

of the singularities, the expansion in ϵ and the generation of the files necessary for the numerical integration. In the numerical part, Fortran or C++ functions forming the coefficient of each term in the Laurent series in ϵ are integrated using the Monte Carlo integration programs contained in the CUBA library [18, 19], or BASES [20]. The different subtasks are handled by perl scripts. The flowchart of the program is shown in Fig. 1 for the basic flow of input/output streams.

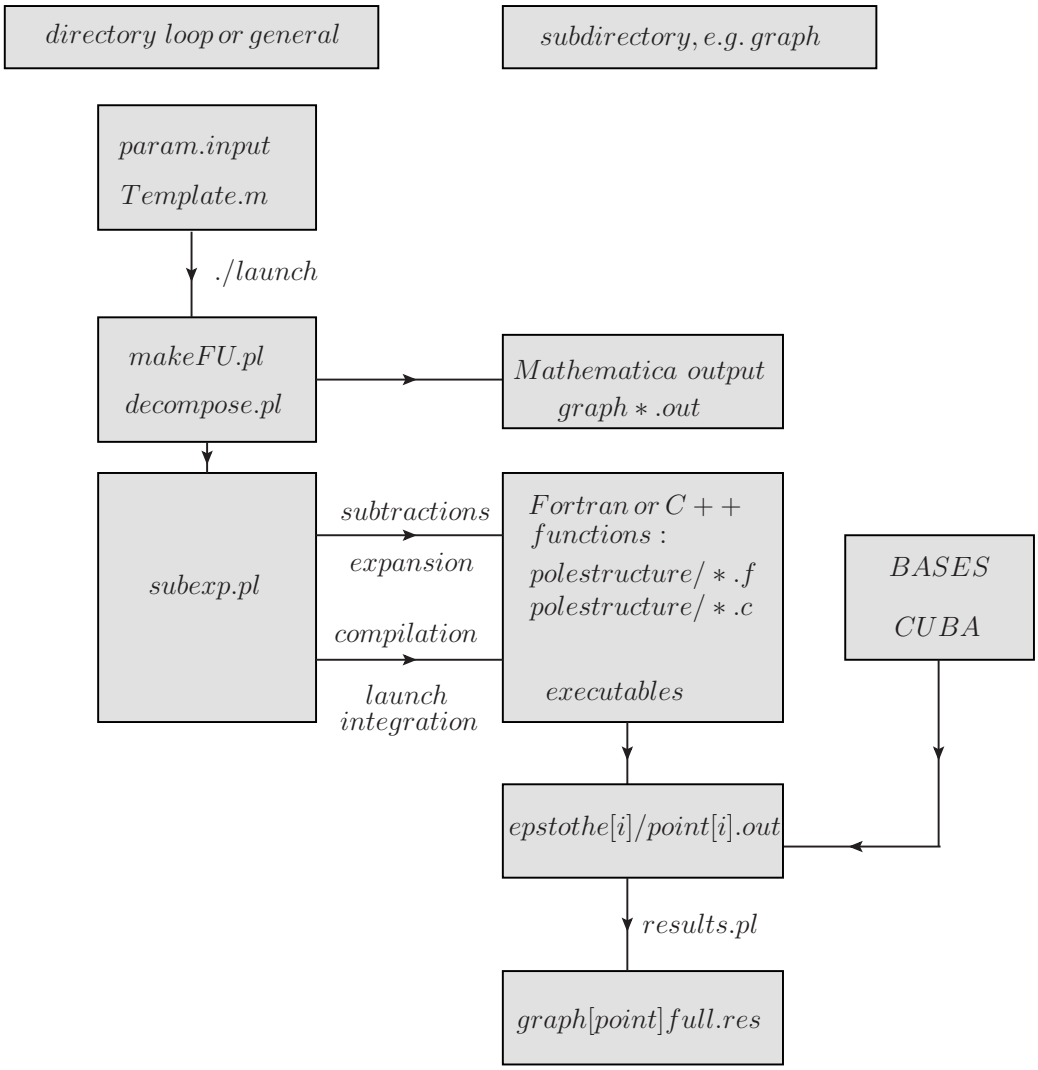

Figure 1. Flowchart showing the main steps the program performs to produce the result files. In each of the subdirectories `loop` or `general`, the file `Template.m` can be used to define the integrand. The produced files are written to a subdirectory created according to the settings given in `param.input`. By default, a subdirectory with the name of the graph or integrand is created to store the produced functions. This directory will contain subdirectories according to the pole structure of the integrand.

The directories `loop` and `general` have the same global structure, only some of the individual files are specific to loops or to more general parametric functions. The directories contain a number of perl scripts steering the decomposition and the numerical integration. The scripts use perl modules contained in the subdirectory `perlsrc`.

The Mathematica source files are located in the subdirectories `src/deco` (files used for the decomposition), `src/subexp` (files used for the pole subtraction and expansion in ϵ) and `src/util` (miscellaneous useful functions). The documentation, created by *robodoc* [21] is contained in the subdirectory `doc`. It contains an index to look up documentation of the source code in html format by loading `masterindex.html` into a browser.

In order to use the program, the user only has to edit the following two files:

- `param.input`: (text file)
  specification of paths, type of integrand, order in ϵ, output format, parameters for numerical integration, further options
- `Template.m`: (Mathematica syntax)
  - for loop integrals: specification of loop momenta and propagators, resp. of the topology; optionally numerator, non-standard propagator powers, space-time dimensions
  - for general functions: specification of integration variables, integrand, variables to be split

The program comes with example input and template files in the subdiretories `loop/demos` respectively `general/demos`, described in detail in [1].

4. Installation and usage

4.1. Installation

The program can be downloaded from
`http://projects.hepforge.org/secdec`.

Unpacking the tar archive via *tar xzvf SecDec.tar.gz* will create a directory called `SecDec` with the subdirectories as described above. Then change to the `SecDec` directory and run *./install*.

Prerequisites are Mathematica, version 6 or above, perl (installed by default on most Unix/Linux systems), a Fortran compiler (e.g. gfortran, ifort), and a C++ compiler if the C++ option is used.

4.2. Usage

(i) Change to the subdirectory `loop` or `general`, depending on whether you would like to calculate a loop integral or a more general parameter integral.

(ii) Copy the files `param.input` and `Template.m` to create your own parameter and template files `myparamfile`, `mytemplatefile`.

(iii) Set the desired parameters in `myparamfile` and specify the integrand in `mytemplatefile`.

(iv) Execute the command *./launch -p myparamfile -t mytemplatefile* in the shell.
If you omit the option *-p myparamfile*, the file `param.input` will be taken as default. Likewise, if you omit the option *-t mytemplatefile*, the file `Template.m` will be taken as default. If your files *myparamfile, mytemplatefile* are in a different directory, say, *myworkingdir*, use the option **-d myworkingdir**, i.e. the full command then looks like *./launch -d myworkingdir -p myparamfile -t mytemplatefile*, executed from the directory `SecDec/loop` or `SecDec/general`.

(v) Collect the results. Depending on whether you have used a single machine or submitted the jobs to a cluster, the following actions will be performed:

- If the calculations are done sequentially on a single machine, the results will be collected automatically (via `results.pl` called by `launch`). The output file will be displayed with your specified text editor.
- If the jobs have been submitted to a cluster, when all jobs have finished, execute the command *./results.pl [-d myworkingdir -p myparamfile]*. This will create the files containing the final results in the `graph` subdirectory specified in the input file.

(vi) After the calculation and the collection of the results is completed, you can use the shell command *./launchclean[graph]* to remove obsolete files.

It should be mentioned that the code starts working first on the most complicated pole structure, which takes longest. This is because in case the jobs are sent to a cluster, it is advantageous to first submit the jobs which are expected to take longest.

4.3. New Features

Version 2.0 of SecDec contains the following new features, some of which will be illustrated by examples in Section 5. More details will be given in a forthcoming publication [22].

- Multi-scale loop integrals can be evaluated without restricting the kinematics to the Euclidean region.
- The possibility to loop over ranges of parameter values is automated, with the option of outputting results in a format suitable for plotting.
- The user can define functions at a symbolic level and specify them only later after the integrand has been transformed into a set of finite functions for each order in ϵ.
- The regulator of the parameter integrals can be different from the dimensional regulator ϵ. This is particularly useful to define e.g. measurement functions at a later stage of the calculation.
- For scalar multi-loop integrals, the integrand can be constructed directly from the topology of the diagram. The user only has to provide the labels of the vertices connected by the propagators and the propagator masses, but does not have to provide the momentum flow.
- The files for the numerical integration of multi-scale loop integrals are written in C++ rather than Fortran. For integrations in Euclidean space, both the Fortran and the C++ versions are supported.

4.4. Implementation of the contour deformation

Unless the function $\mathcal{F}$ in eq. (4) is of definite sign for all possible values of invariants and Feynman parameters, the integrand of a multi-loop integral will vanish within the integration region on a hypersurface given by the solutions of the Landau equations (6),(7). However, we can avoid the poles on the real axis by a deformation of the integration contour into the complex plane. As long as the deformation is in accordance with the causal $i\delta$ prescription of the Feynman propagators, and no poles are crossed while changing the integration path, we can make use of Cauchy's theorem to choose an integration contour such that the integration is convergent. The $i\delta$ prescription for the Feynman propagators tells us that the contour deformation into the complex plane should be such that the imaginary part of $\mathcal{F}$ should always be negative. For real masses and Mandelstam invariants s_{ij}, the following Ansatz [23, 24, 25] is convenient:

$$\begin{aligned} \vec{z}(\vec{x}) &= \vec{x} - i\,\vec{\tau}(\vec{x}) \\ \tau_k &= \lambda\, x_k(1-x_k) \sum_{j=1}^{N-1} \frac{\partial \mathcal{F}}{\partial x_j}\,. \end{aligned} \tag{9}$$

In terms of the new variables, we thus obtain

$$\mathcal{F}(\vec{z}(\vec{x})) = \mathcal{F}(\vec{x}) - i\,\lambda \sum_{j=1}^{N-1} \left(\frac{\partial \mathcal{F}}{\partial x_j}\right)^2 + \mathcal{O}(\lambda^2)\,, \tag{10}$$

such that $\mathcal{F}$ aquires a negative imaginary part of order λ.

The convergence of the numerical integration can be improved significantly by choosing an "optimal" value for λ. Values of λ which are too small lead to contours which are too close to the poles on the real axis and therefore lead to bad convergence. Too large values of λ can modify the real part of the function to an unacceptable extent and could even change the sign of the imaginary part if the terms of order λ^3 get larger than the terms linear in λ. This would lead to a wrong result. Therefore we implemented a four-step procedure to optimize the value of λ, consisting of

- ratio check: To make sure that the terms of order λ^3 in eq. (10) do not spoil the sign of the imaginary part, we evaluate the ratio of the terms linear and cubic in λ for a quasi-randomly chosen set of sample points to determine a maximal allowed $\lambda = \lambda_{max}$.
- modulus check: The imaginary part is vital at the points where the real part of $\mathcal{F}$ is vanishing. In these regions, the deformation should be large enough to avoid large numerical fluctuations due to a highly peaked integrand. Therefore we check the modulus of each subsector function $\mathcal{F}_i$ at a number of sample points, and pick the fraction of the value of λ_{max} maximising the function with the minimal modulus, i.e. the value of lambda which keeps $\mathcal{F}$ furthest from zero.
- individual $\lambda(i,j)$ adjustments: If the values of $\frac{\partial \mathcal{F}_i}{\partial x_j}$ are very different in magnitude, it can be convenient to have an individual parameter $\lambda(i,j)$ for each subsector function $\mathcal{F}_i$ and each Feynman parameter x_j.
- sign check: After the above adjustments to λ have been made, the sign of $\mathrm{Im}(\mathcal{F})$ is again checked for a number of sample points. If the sign is ever positive, this value of λ is disallowed.

5. Examples and Results

In this example, we will demonstrate three of the new features of the SecDec 2.0 program: the construction of $\mathcal{F}$,$\mathcal{U}$ directly from the topology of the graph, the evaluation of the graph in the physical region, and how results for a whole set of different numerical values for the invariants can be produced and plotted in an automated way. We will use the two-loop diagram shown in Fig. 2 as an example. Numerical results for this diagram have been produced in [26, 27], and an analytical result can be found in [28], where it is called P_{126}.

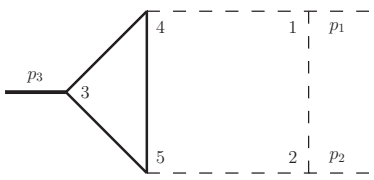

Figure 2. Two-loop vertex graph P_{126}, containing a massive triangle loop. Bold lines are massive, dashed lines are massless. The vertices are labelled to match the construction of the integrand from the topology as explained in the text.

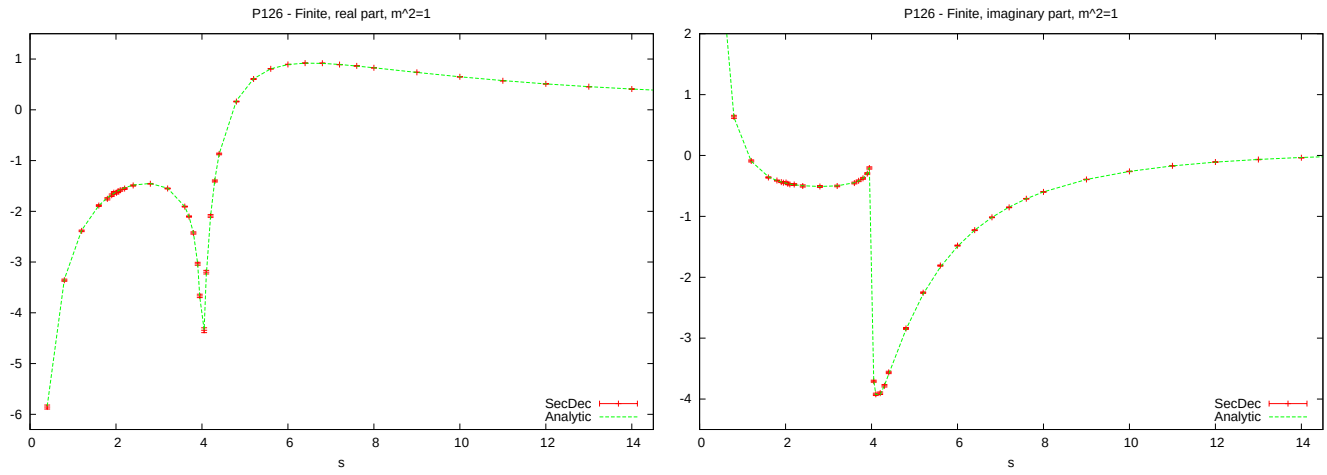

Figure 3. Comparison of analytic and numerical results for the diagram P_{126}.

5.1. Topology-based construction of the integrand
The template file `templateP126.m` in the `demos` subdirectory contains the following lines:
$proplist = \{\{ms[1],\{3,4\}\},\{ms[1],\{4,5\}\},\{ms[1],\{5,3\}\},\{0,\{1,2\}\},\{0,\{1,4\}\},\{0,\{2,5\}\}\};$
$onshell = \{ssp[1] \to 0, ssp[2] \to 0, ssp[3] \to sp[1,2]\};$
where each entry in *proplist* corresponds to a propagator of the diagram; the first entry is the mass of the propagator, and the second entry contains the labels of the two vertices which the propagator connects. Note that if an external momentum p_k is flowing into the vertex, the vertex also must have the label k. For vertices containing only internal propagators the labelling is arbitrary. The on-shell conditions in the above example state that $p_1^2 = p_2^2 = 0$, $p_3^2 = s$.

5.2. Results in the physical region
To run this example, from the `loop` directory, issue the command `./launch -d demos -p paramP126.input -t templateP126.m`. The timings for the finite part and a relative accuracy of 1%, using CUBA-3.0 [19], are about 12 secs for a typical point far from the $s = 4m^2$ threshold, and 13.6 secs for a point close to threshold ($s/m^2 = 3.9$) on an Intel(R) Core i7 CPU at 2.67GHz.

5.3. Producing data files for sets of numerical values
To loop over a set of numerical values for the invariants s and m^2 once the C++ files are created, issue the command
`./multinumerics.pl -d demos -p multiparamP126.input`. This will run the numerical integrations for the values of s and m^2 specified in the file `demos/multiparamP126.input`. The files containing the results will be found in `demos/2loop/P126`, and the files `p-2.gpdat`, `p-1.gpdat` and `p0.gpdat` will contain the data files for each point, corresponding to the coefficients of $\epsilon^{-2}, \epsilon^{-1}$ and ϵ^0 respectively. These files can be used to plot the results against the analytic results using gnuplot. This will produce the files `P126R0.ps, P126I0.ps` which will look like Fig. 3.

6. Conclusions
We have presented the program SECDEC 2.0, which can be used to factorise dimensionally regulated singularities and numerically calculate multi-loop integrals in an automated way. As a new feature of the program, it now can deal with fully physical kinematics, i.e. is not restricted to the Euclidean region anymore. A new construction of the integrand, based entirely

on topological rules, is also included, and the new features are demonstrated for the examples of a massive two-loop three-point function. In addition, the program can produce numerical results for more general parameter integrals, as they occur for example in phase space integrals for multi-particle production with several unresolved massless particles, and offers the possibility to include symbolic functions which can be used to define measurement functions like e.g. jet algorithms at a later stage. We are looking forward to applications of the program for the calculation of higher order corrections to various observables.

Acknowledgments
J.C. was supported by the British Science and Technology Facilities Council (STFC). We also acknowledge the support of the Research Executive Agency (REA) of the European Union under the Grant Agreement number PITN-GA-2010-264564 (LHCPhenoNet).

References

[1] Carter J and Heinrich G 2011 *Comput.Phys.Commun.* **182** 1566–1581 (*Preprint* 1011.5493)
[2] Binoth T and Heinrich G 2000 *Nucl. Phys.* **B585** 741–759 (*Preprint* hep-ph/0004013)
[3] Roth M and Denner A 1996 *Nucl. Phys.* **B479** 495–514 (*Preprint* hep-ph/9605420)
[4] Hepp K 1966 *Commun. Math. Phys.* **2** 301–326
[5] Smirnov A V, Smirnov V A and Tentyukov M 2009 (*Preprint* 0912.0158)
[6] Bogner C and Weinzierl S 2008 *Comput. Phys. Commun.* **178** 596–610 (*Preprint* 0709.4092)
[7] Gluza J, Kajda K, Riemann T and Yundin V 2011 *Eur.Phys.J.* **C71** 1516 (*Preprint* 1010.1667)
[8] Heinrich G 2008 *Int. J. Mod. Phys.* **A23** 1457–1486 (*Preprint* 0803.4177)
[9] Anastasiou C, Herzog F and Lazopoulos A 2010 (*Preprint* 1011.4867)
[10] Ferroglia A, Passera M, Passarino G and Uccirati S 2003 *Nucl.Phys.* **B650** 162–228 (*Preprint* hep-ph/0209219)
[11] Bernstein L 1972 66
[12] Tkachov F V 1997 *Nucl.Instrum.Meth.* **A389** 309–313 (*Preprint* hep-ph/9609429)
[13] Nakanishi N 1971 *Graph Theory and Feynman Integrals* (Gordon and Breach, New York)
[14] Tarasov O V 1996 *Phys. Rev.* **D54** 6479–6490 (*Preprint* hep-th/9606018)
[15] Landau L D 1959 *Nucl. Phys.* **13** 181–192
[16] Eden R J, Landshoff P V, Olive D I and Polkinghorne J C 1966 *The Analytic S-Matrix* (Cambridge University Press)
[17] Wolfram S , Mathematica, Copyright by Wolfram Research
[18] Hahn T 2005 *Comput. Phys. Commun.* **168** 78–95 (*Preprint* hep-ph/0404043)
[19] Agrawal S, Hahn T and Mirabella E 2011 (*Preprint* 1112.0124)
[20] Kawabata S 1995 *Comp. Phys. Commun.* **88** 309–326
[21] Slothouber F and et al *http://www.xs4all.nl/ rfsber/Robo/robodoc.html*
[22] Borowka S, Carter J and Heinrich G , in preparation.
[23] Soper D E 2000 *Phys. Rev.* **D62** 014009 (*Preprint* hep-ph/9910292)
[24] Nagy Z and Soper D E 2006 *Phys. Rev.* **D74** 093006 (*Preprint* hep-ph/0610028)
[25] Binoth T, Guillet J P, Heinrich G, Pilon E and Schubert C 2005 *JHEP* **10** 015 (*Preprint* hep-ph/0504267)
[26] Bonciani R, Mastrolia P and Remiddi E 2004 *Nucl.Phys.* **B690** 138–176 (*Preprint* hep-ph/0311145)
[27] Ferroglia A, Passera M, Passarino G and Uccirati S 2004 *Nucl.Phys.* **B680** 199–270 (*Preprint* hep-ph/0311186)
[28] Davydychev A I and Kalmykov M 2004 *Nucl.Phys.* **B699** 3–64 (*Preprint* hep-th/0303162)