1 SecDec User Manual

Version 3.0, last manual update February 22, 2016

1.1 Installation

The program can be downloaded from `http://secdec.hepforge.org`. Unpacking the tar archive via *tar xzvf SecDec-3.0.tar.gz* will create a directory called `SecDec-3.0`. Running `make` in the `SecDec-3.0` directory will call the *install* script, which will check whether Mathematica and perl are present and compile the numerical integration libraries Cuba [1, 2, 3], Bases [4] and Cquad [5], along with the quasi-random sequence generator Sobol, which come with the package. Prerequisites are Mathematica [6] version 7 or above, perl (installed by default on most Unix/Linux systems), a C++03 compliant compiler, and a Fortran compiler if the Fortran part is used. Contour deformation is not available in Fortran.

Please note that the current stable release of Mathematica (v10.0.2) has a bug which causes non-interactive sessions, such as those used by SecDec , to hang if parallel kernels are launched. Therefore we have decided (starting from version 3.0.6) to disable the parallelization for the *algebraic* part of the program by setting `nbmathsubkrnls=0` as the default. If the user would like to speed up the algebraic part by specifying a nonzero value for `nbmathsubkrnls` , we recommend using a different version of Mathematica.

The program `Normaliz` 2.10.1 [7, 8] is needed for the geometric decomposition strategies `G1` and `G2`. Precompiled executables for different systems can be downloaded from `http://www.math.uos.de/normaliz/Normaliz2.10.1/` and have to be moved into the `src/` subdirectory of SecDec-3.0.

The user can check whether the installation was successful with the command `make check`, which will run a few test examples and compare the results to the pre-calculated result coming with the program package.

1.2 Conventions for the integral definition

The loop integrals calculated by SecDec are defined as

$$G = \int \prod_{l=1}^{L} \mathrm{d}^D \kappa_l \, \frac{1}{\prod_{j=1}^{N} P_j^{\nu_j}(\{k\},\{p\},m_j^2)} \tag{1}$$

$$\mathrm{d}^D \kappa_l = \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \, , \; P_j(\{k\},\{p\},m_j^2) = q_j^2 - m_j^2 + i\delta \, ,$$

i.e. we do *not* include a factor $i\pi^{\frac{D}{2}}$ for each loop in the numerical result. Introducing Feynman parameters in Eq. (1) leads to

$$\begin{aligned} G &= \frac{(-1)^{N_\nu}}{\prod_{j=1}^{N}\Gamma(\nu_j)}\Gamma(N_\nu - LD/2) \\ &\quad \times \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_j - 1}\, \delta(1 - \sum_{l=1}^{N} x_l) \frac{\mathcal{U}^{N_\nu-(L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}} \end{aligned} \tag{2}$$

where

$$\mathcal{F}(\vec{x}) = \det(M) \left[\sum_{j,l=1}^{L} Q_j \, M_{jl}^{-1} \, Q_l - J - i\,\delta \right] \tag{3}$$

$$\mathcal{U}(\vec{x}) = \det(M),\; N_\nu = \sum_{j=1}^{N} \nu_j \; . \tag{4}$$

In the expressions above, M is an $L \times L$ matrix containing Feynman parameters, Q is an L-dimensional vector, where each entry is a linear combination of external momenta and Feynman parameters, and J is a scalar expression containing kinematic invariants and Feynman parameters, for more details we refer to Ref. [9].

The numerical result given by SecDec will be the one for the integral as defined in Eq. (1). This implies that the prefactor $\frac{(-1)^{N_\nu}}{\prod_{j=1}^{N}\Gamma(\nu_j)}\Gamma(N_\nu - LD/2)$ coming from the Feynman parametrisation by default will be *included* in our numerical result. However, the user can define a different prefactor to be factored out from the numerical result in the input file `math.m`.

1.3 Usage

In the following we refer to three directory structures:

- **input directory**: the directory in which the user's input files (parameter file, math file and kinematics file) reside,
- **output directory**: the directory into which SecDec-3.0 will write output files
- **SecDec directory**: the location of SecDec-3.0.

SecDec has various setups: one for standard loop integrals, termed 'loop setup' in the following, and one for general parametric functions, termed 'general setup', as reflected by the two directories `loop` and `general`. While the parametric functions treated in the `general` folder are not accessible to contour deformation, the latter is still available for integrals which go beyond standard loop integrals. Such non-standard integrals can be defined by the user, and

therefore this setup is referred to as 'user-defined setup', which is an option within the `loop` directory.

The program is called by invoking the script `secdec`, located in the main SecDec directory. We recommended to add the path to the `secdec` script to the user's default search paths, so that it can be called from anywhere on the system. In the following, we assume that `secdec` was added to the search path, so that it can be called without always specifying the path to the script explicitly.

1.3.1 Usage in the loop setup

1.3.2 Basic usage

- Create templates for the input files with the command
  `secdec -prep`
  This command generates the files `param.input`, `kinem.input` and `math.m`.

- Input the graph name and desired order in ϵ (`epsord`) into `param.input`, define the loop integral in `math.m` and specify one or more kinematic points in `kinem.input`. Please note that the order of the values for the kinematic invariants given in `kinem.input` must match the order of the invariants given in `math.m`, i.e. the entries of the lists `KinematicInvariants`, `Masses` (masses always listed last).

- To run SecDec : simply issue the command
  `secdec`
  If the input files have been renamed, the command is
  `secdec -p <myparam.input> -m <mymath.m> -k <mykinem.input>`.

- Switch to the output directory which has been created. It carries the *graph name* specified in `param.input`. The results are in the `results` folder, containing a file `graph_pointname.res` for each point specified in the kinematics file, and `plotfile<i>.gpdat` files for each ϵ-order `i`, where the results for all kinematic points have been appended.

1.3.3 Intermediate usage

The different stages of SecDec can also be run separately. Creating and editing the input files as before, one can run the algebraic part only by
`secdec -algebraic`
This allows for example to get a first idea of the pole structure generated during the decomposition. One can also examine the $\mathcal{F}$ and $\mathcal{U}$ functions in the folder named `FU` in the output directory. It is also the way to proceed if the subsequent numerical evaluation should not be done locally, but on a cluster.

The numerical part is run by the command
`secdec -numerics`
The results are collected by
`secdec -collect`.

Options:

- to add a few more kinematic points, for example in the threshold region, one can edit the `kinem.input` file. The user should delete the old (already calculated) points from the `kinem.input` file and add the new points with new point names. It is also possible to edit the `kinematics.input` file in the output directory and run `secdec -numeric` from there.

- One can re-run the numerics in cluster mode by uploading the output directory to the head node and running the submission scripts in the `cluster` folder. (Note: please examine the submission scripts before submitting them to your cluster, they may need to be adapted to the particular cluster setup).

Expert usage

There are various possibilities for the user to control the different stages of the calculation, shown in Table 1. The optional detailed or basic commands must be run in the order presented. If the user specifies an `exe flag` and calls `secdec` without a basic or detailed command then all tasks with a lower exe flag will also be executed. If the user calls `secdec` with a basic or detailed command, for example `secdec -subexpand` then only that task will be performed. Specifically, in this example `makeparams`, `makeFU`, `decompose` and `preparesubexpand` would not be executed.

1.3.4 Evaluation of selected pole coefficients

In version 3, the user has the possibility to calculate certain pole structures/epsilon orders individually. The pole structures are labelled by a string of the type `i` l `j` h `k`, where l stands for linear poles, h stands for higher than linear poles, while the number `i` of "usual" logarithmic poles is put at the beginning of the string. For example, the pole structure `2l1h0` means "2 logarithmic poles, 1 linear pole, 0 higher than linear poles. By "linear pole" we mean that a factorized Feynman parameter has an exponent of the form $x^{-2-b\epsilon}$. For more details we refer to Ref. [10].

Each pole structure contains several orders in the ϵ-expansion, ranging from the maximal pole of the given pole structure to the expansion order in ϵ specified by the user. The coefficients of a certain order in ϵ are stored in subfolders of the pole structure folders, which are labelled `epstothe-2`, `epstothe-1`, etc. The user can select different pole structures as well as individual ϵ orders by commands of the form

```
secdec [-<subexpand/preparenumerics/numerics>] [-polestructs=
<polestructs>] [-epsords=<epsords>]
```

The behaviour of the program is the following:

- if `polestructs` is not specified, loop over all (contributing) pole structures

exe-flag	command (detailed)	command (basic)	functionality
≥ 0	makeparams makeFU decompose preparesubexpand	algebraic	• extracts parameters from the graph definition given in math.m • constructs the graph polynomials $\mathcal{F}$ and $\mathcal{U}$ • performs the iterated sector decomposition • writes the Mathematica files which are needed to run the subtractions and ϵ-expansions for each pole structure
≥ 1	subexpand preparenumerics		• performs the subtractions and ϵ-expansions for each pole structure and writes the functions `f*.cc` (resp. `f*.m`) to be evaluated numerically for each pole structure • writes the files needed to perform the numerical integration of each pole structure (including the scripts for job submission to a cluster)
≥ 2	numerics	numerics	• performs the compilation and runs the executables
3	collectresults	collect	• performs the collection of the results

Table 1: The different execution stages of SecDec and the possibilities for the user to steer them. The optional detailed or basic commands can be issued with, e.g. `secdec -algebraic`, and must be run in the order presented.

- if `epsords` is not specified, loop over all ϵ orders
- if `polestructs` is specified either as a list
  `secdec [-<subexpand/preparenumerics/numerics>] -polestructs=2l0h0,1l0h0`, or separately
  `secdec [-<subexpand/preparenumerics/numerics>] -polestructs=2l0h0 -polestructs=1l0h0`
  loop over just these pole structures (for all ϵ orders)
- if `epsords` is specified either as a list or separately, loop over just these ϵ orders (for all contributing pole structures)
- if both `polestructs` and `epsords` is specified, loop over just these ϵ orders and pole structures.

If the *togetherflag* is set to one, all pole structures have been combined into one folder called `together`. In this case, the individual ϵ orders can still be calculated separately with the same logic as above, except that there is only one pole structure called "together". For example, in order to perform the numerical evaluation only for the ϵ^{-4} part of all orders in ϵ contained in the `together` folder, the command would be
`secdec -numerics -epsords=-4`.

1.3.5 Description of the input files

The description below is for loop diagrams; the input files in the subdirectory `general` to compute more general parametric functions did not change between versions 2 and 3, and therefore we refer to descriptions in Refs. [11, 12] in the `general` branch.

- `param.input`: (text file)
  The mandatory parameters the user needs to specify are
  - `graphname`: a name for the graph to be calculated
  - `epsord`: the desired expansion order in ϵ.

  All other values take defaults if not specified. A detailed description of all options is given is Section 1.3.6.
- `math.m`: (Mathematica syntax)
  This file contains the definition of the graph to be calculated.
  `momlist`: list of loop momenta
  `proplist`: list of propagators, either in momentum flow representation or as a list containing the propagator mass and the labels of the vertices the propagator is connecting. Please note that the label of the vertex which contains the external momentum p_i should be i. For vertices involving only internal lines, the labelling is arbitrary. For more details we refer to Ref. [11].

`numerator`: there are two possiblities to define a numerator:
(a) give a list of loop momenta contracted with either external momenta or loop momenta. Each Lorentz contraction should be denoted by "*", while each contracted factor should form an element of a list. E.g., for $2\,k_1 \cdot p_1\, k_2 \cdot p_2$, the syntax is `numerator=2,k1*p1,k2*p2`.
(b) define an additional propagator and specify a negative index in `powerlist` (see below).
The default is numerator=1 (scalar integral).
`powerlist`: list of porpagator powers, also called "indices" in the literature. Can also take zero or negative integer values.
`Dim`: the dimension of the loop momenta. The default is Dim$=4-2\epsilon$.
`prefactor`: the prefactor specified here will be factored out of the numerical result. This means that the numerical result will be divided by the prefactor given here. Please note that a factor $\frac{(-1)^{N_\nu}}{\prod_{j=1}^{N}\Gamma(\nu_j)}\Gamma(N_\nu - LD/2)$, which comes from the Feynman parametrisation, will be included by default in the numerical result, according to the integral definition in Eq. (1).
`ExternalMomenta`: list of external momenta occurring in the graph definition (only necessary if the graph definition contains the momenta explicitly, i.e. in momentum flow representation). If the length of this list is different from the number of external legs, give also the number of external legs as `externallegs=...`.
`KinematicInvariants`: list of symbols for kinematic invariants (formed from Lorentz vectors) occurring in the diagram. The symbols can be chosen by the user.
`Masses`: list of symbols for the masses. In the case of massive on-shell lines, i.e. $p^2 = m^2$, where m is a propagator mass as well as the mass of an external leg, the way to proceed is to define `SP[p,p]`$\rightarrow$ `m2` in `ScalarProductRules`, such that no extra symbol needs to be specified for p^2.
Note that the list of masses should always contain the propagator masses used in `proplist`.
`ScalarProductRules`: list of replacement rules for the kinematic invariants formed by external momenta.
`splitlist`: allows to specify those Feynman parameters which may have an endpoint singularity at $z_i = 1$, such that the integral will be split at 1/2 and the singularity at one will be remapped to a singularity at zero.

- `kinem.input`: (text file)
  Should contain numerical values for the kinematic invariants, **in the same order** as the symbols for the kinematic invariants given in the fields `KinematicInvariants` and `Masses` in the Mathematica input file (called `math.m` here). The numerical values for the masses should always be listed *after* the invariants formed from Lorentz vectors.

  If kinemloop.input contains several lines, each line will be evaluated as a new kinematic point. In order to be able to distinguish the runs/results for

the different kinematic points, each array of numerical values should have a label prepended which defines the "pointname". Example: if `mathloop.m` contains `KinematicInvariants = s,t,p1sq` and `Masses=m1sq`, the corresponding kinematics input file where `m1sq` takes the value 1 for point `p1` and 5 for point `p2` should look like (the values for s,t,p1sq are e.g. 500, -88, 100)

```
p1 500 -88 100 1
p2 500 -88 100 5
```

1.3.6 Detailed description of all options in `param.input`

It should be emphasized again that the only mandatory fields are `graph` and `epsord`, all other parameters take default values if not specified, which are given in brackets after the keyword in the description below.

graph The name of the diagram or parametric function to be computed is specified here. The graph name can contain underscores and numbers, but should not contain commas.

epsord The order to which the Laurent series in ϵ should be expanded, starting from $\epsilon^{-maxpole}$. The value of *epsord=0* will calculate the pole coefficients and the finite part. Note that epsord can be negative if only the pole coefficients up to a certain order should be computed.

outputdir () specifies the name of the directory where the produced files will be written to, the **absolute path** should be given. If left empty, a subdirectory of the input directory with the name of the graph will be created.

contourdef (False) The contour deformation can be switched on or off by choosing *contourdef=True/False* (lower case letters are also possible). For multi-scale problems, respectively diagrams with non-Euclidean kinematics, set *contourdef=True*.

lambda (1.0) λ is a parameter controlling the contour deformation. The program takes the λ value given by the user in the input file as a starting point. The program then performs checks and optimizations to find an 'optimal' value for λ. The user should pick an initial value which is rather large, as the program will decrease λ appropriately, while it cannot increase λ. Values of λ between 1 and 3 are usually a good choice.

strategy (X) Choice of the decomposition strategy. The default is `X`, which is the same strategy as in previous versions, which is usually the most efficient one, but is not guaranteed to stop. If the decomposition does not seem to stop, strategies `G1` or `G2` should be chosen, which are based on a geometrical algorithm as described in the SecDec-3.0 paper, and which cannot run into an infinite recursion. Within strategy `G2`, no primary sector decomposition is done. Therefore it is obviously not possible to specify only selected primary sectors to be calculated when using this strategy.

exeflag (3) The *exeflag* can be used to execute the program only up to a certain stage. There are three basic stages: i) the algebraic part, ii) the numerical part and iii) collecting the results. The algebraic part can be further split into a part doing only the iterated sector decomposition (e.g. to get an idea about the pole structure), and a part performing the subtractions and the expansion in epsilon. The values of the *exeflag* correspond to the following stages:

- 0: The parameters like the number of loops, propagators etc are extracted from the user's Mathematica input file; the graph polynomials $\mathcal{F}$ and $\mathcal{U}$ (and the numerator in the case of tensor integrals) are constructed; the iterated sector decomposition is done; the scripts `subandexpand*.m` in the *graph* subdirectory for the subtractions and epsilon expansions are created, but not run.
- 1: The subtractions and expansions in ϵ are performed and the resulting functions for the pole coefficients are written to C^{++} or Mathematica files; all the other files needed for the numerical integration are created as well.
- 2: Compilation and numerical integrations are performed.
- 3: The results are collected.

All exeflags imply that the steps corresponding to lower exeflags will automatically be performed as well. However, there is also the possibility to skip previous steps by calling SecDec with a basic or detailed command. For example, to run only the the numerical part the call is `secdec -numerics`. This command will *not* restart the whole algebraic part, but just compile the functions and run the executables. An error will be thrown if the functions have not been produced beforehand. Table 1 gives a schematic overview of the various options to control the program flow.

Please note that if the `clusterflag` is switched on, it is assumed that the user will produce the functions locally and then compile and run them on a cluster. Therefore, in cluster mode, the program will perform the algebraic part and produce the submission scripts, and then stop, independent of the value of the `exeflag`, as the user should control the job submission (`secdec -numerics`) and the collection of the results (`secdec -collect`) in cluster mode.

Advanced usage

togetherflag (0) This flag defines whether to integrate subsets[1] of functions contributing to a certain ϵ-order separately (*togetherflag=0*), or to sum all functions for a certain order in ϵ prior to integration (*togetherflag=1*).

[1]The subsets are naturally formed by the fact that functions contributing to a certain ϵ-order can descend from different pole structures.

The latter will avoid large cancellations between results for functions descending from different pole structures and thus give a more realistic error. However, *togetherflag=1* is not recommended for cases where the individual functions are already very complicated.

grouping (2000) It can be beneficial to first sum a few functions before integrating them. Choosing a value for the grouping which is nonzero defines an upper limit (in kilobytes) for a file containing the sum of a number of functions. The number of kbytes is set by *grouping=#kbytes*. Setting *grouping=0*, all functions `f*.cc` resp. `f*.m` are integrated separately. In practice, grouping=0 has proven to lead to faster convergence and more accurate results in most cases. However, for integrals where large cancellations among the different functions occur, the grouping value should be chosen $\neq 0$. The log files `*results*.log` in the results directory contain the results from the individual sub-sector integrations. These files are useful to spot cancellations between the individual functions and to adjust the settings accordingly.

IBPflag (0) Set *IBPflag=0* if the integration by parts option should not be used and *IBPflag=1* if it should be used. *IBPflag=2* is designed to use IBP relations when it is deemed efficient to do so. Using the integrations by parts method takes more time in the subtraction and expansion step and generally results in more functions for numerical integration. Its usefulness is mainly for cases where (spurious) linear poles of the type $x^{-2-b\epsilon}$ are found in the decomposition, as it reduces the power of x in the denominator.

infinitesectors () This field should be empty if the default strategy `X` or the strategies `G1` or `G2` are applied. It offers the possibility to use a different 'heuristic' strategy [13] for certain primary sectors if they seem to suffer from infinite recursion. The primary sectors given in the list (comma separated) will then be decomposed with this heuristic strategy. It can avoid infinite recursion in cases where strategy `X` fails, but is not guaranteed to stop. For example, *infinitesectors=2,3* results in the application of this heuristic strategy to primary sectors 2 and 3. In examples with massive propagators, one should put the labels belonging to the massive propagators into the list.

primarysectors () This field allows to calculate selected primary sectors only. If left blank, *primarysectors* defaults to all, i.e. *primarysectors=1,...,N* will be assumed, where N is the number of propagators. This option is useful if a diagram has symmetries such that some primary sectors yield the same result. It cannot be used in combination with strategy `G2`. See Section 1.3.7 for the special usage in combination with the user-defined setup.

multiplicities () Specify the *multiplicities* of the primary sectors listed above. List the *multiplicities* in same order as the corresponding sectors above.

If left blank, a default multiplicity of 1 is set for each primary sector. See Section 1.3.7 for the special usage in combination with the user-defined setup.

rescale (0) If there are large differences in magnitude in the kinematic invariants occurring in a diagram, it can be beneficial to rescale all invariants by the largest one in order to reach faster convergence during the numerical integration. The rescaling can be switched on with *rescale=1* (default is *rescale=0*). *Please note:* If switched on, it is not possible to set explicit values (numbers) for any non-zero invariant in the *ScalarProductRules=* conditions in the Mathematica file `math.m`.

nbmathsubkrnls (0) Maximal number of Mathematica subkernels to be used by Mathematica. The iterated decomposition is not parallelized by default; setting it to a nonzero number switches on the parallelization using the specified number of subkernels.

smalldefs (0) *smalldefs=1* minimizes the deformation of the contour. It can be useful for example in the presence of oscillatory integrands.

largedefs (0) If the integrand is expected to have (integrable) endpoint singularities at $x_j = 0$ or 1, *largedefs=1*, can help to have a sufficiently large deformation close to the endpoints. The default is *largedefs=0*.
Note that setting both flags *largedefs* and *smalldefs* to zero is perfectly possible, as the flags operate on different parts of the deformation internally. For more details we refer to Ref. [11].

optlamevals (4000) The number of pre-samples to determine the optimal contour deformation parameter λ can be chosen by assigning a number to *optlamevals*.

Parameters related to the numerical integration and external libraries

We restrict ourselves to the settings common to all integrators in the description below. For more details about the CUBA parameters, the user is referred to the CUBA documentation, Refs. [1, 3].

compiler (gcc) Choice of the C-compiler.

CCargs (-O) Compiler options for the C-compiler to compile the numerical integration files.

sobolpath () The path to `sobol` can be specified here, if different from the default `[path_to_secdec]/src/sobol`. The `sobol` quasi-random number generator is only used if *contourdef=True*.

cquadpath () The path to `cquad` can be specified here, if different from the default `[path_to_secdec]/src/cquad`. Note that the program will use this integrator automatically if an integral or a pole contribution is found to depend on only 1 Feynman parameter, irrespective of what has been chosen as integrator below.

integrator (3) The program for the numerical integration can be chosen here. Vegas (*integrator=1*), Suave (*integrator=2*), Divonne (*integrator=3*) and Cuhre (*integrator=4*) are part of the CUBA library. To choose a numerical integrator included in Mathematica, *integrator=5* can be chosen.

cubapath () The path to the CUBA library can be specified here, if different from the default `[path_to_secdec]/src/Cuba-4.1`.

cubacores (1/0) The maximal number of cores Cuba is allowed to use. In cluster mode, the default is 1, in single machine mode the default is zero, which means that Cuba will use all available idle cores.

NIntegrateOptions (`AccuracyGoal->3`) Options for the Mathematica NIntegrate command, if *integrator=5* is chosen. Example:
NIntegrateOptions=AccuracyGoal → 2,WorkingPrecision→12,
Method→"AdaptiveMonteCarlo"
Please note that when using NIntegrate, the derired accuracy must be specified using this command (`epsrel` and `epsabs` (see below) just as the other CUBA-specific options have no effect) and it is not possible to obtain an error estimate.

maxeval (10000000) The maximal number of evaluations to be used by the numerical integrator. For this fields and the fields below, a value can be specified for each order in ϵ, separated by commas and starting with the leading pole. If only one value is given, it will be used for all pole coefficients. If the list is shorter than the number of orders in ϵ, the last value of the list will be repeated as often as necessary.

mineval (0) The number of evaluations which should at least be done before the numerical integrator returns a result.

epsrel (`1.e-2`) The desired relative accuracy for the numerical evaluation. Please note that each order in ϵ can be evaluated with a different value for `epsrel` (and `epsabs`), by specifying an individual value for each order in ϵ, separated by commas and starting with the leading pole, as explained above.

epsabs (`1.e-6`) The desired absolute accuracy for the numerical evaluation. These values are particularly important when either the real or the imaginary part of an integral is close to zero.

cubaflags (2) Sets the CUBA verbosity flags. The default is 2, which means that the CUBA input parameters and other useful information, e.g. about

numerical convergence, are written to the log file of the numerical integration.

seed (0) The seed used to generate random numbers for the numerical integration with Cuba. The default is *seed=0*: Cuba will use the Sobol (quasi-) random number generator

Parameters related to the cluster mode

clusterflag (0) Determines how jobs are submitted. Setting *clusterflag=0* (default) the jobs will run on a single machine, with *clusterflag=1* the jobs will run on a cluster (the corresponding files to submit jobs to a cluster will be created, see below).

batchsystem (0) Chooses a format for the scripts steering the submission to a cluster. If *batchsystem* is set to 0, the setup is for the PBS (portable batch system). If the flag is set to 1, a user-defined setup is activated. Currently this is the submission via `condor`, but it can be easily adapted to other batch systems by editing the templates in `loop/src/numerics/` and `loop/perlsrc/makejob.pm`.

clusteroptscompile () In cluster mode: command line arguments passed verbatim to the job submission script for compilation jobs on a cluster

clusteroptsrun () In cluster mode: command line arguments passed verbatim to the job submission script for numerical integration jobs on a cluster.

Parameters related to plotting

xplot (1) This option can be used to control the format of the data files where the results for a range of kinematic points are listed. The variable `xplot` denotes a position in the list of invariants. The corresponding invariant then will be the one which will be plotted on the x-axis. Example: the invariants are s,t,u,m1sq,m2sq. If a scan over m1sq has been performed, such that m1sq should be plotted on the x-axis, then xplot=4 would tell SecDec to write the values for m1sq into the first column of the `*.gpdat` file. The `*.gpdat` files produced by the program have the form `[invariant chosen by xplot] real_result real_error imag_result imag_error timing`.
For 3D plots: if xplot is a list of length L, the first L columns of `*.gpdat` will correspond to the values of the invariants singled out by the xplot labels (e.g. xplot=1,2 would produce data files for a 3D plot in s,t).

1.3.7 User-defined setup

This setup allows to define functions in the Mathematica input file (`math_userdefined.m`) which are not standard loop integrals. It is invoked by the option **`-u`** when calling `secdec`. While the `param.input` file has the

same form as for standard loop integrals, the file `math[userdefined].m` should contain the specification of the user-defined functions. Most fields are the same as in `math[loop].m`. However, instead of the definition of a graph, the user can define a list of functions to be decomposed. The detailed format is specified in the example `10_userdefined_triangle_1L` contained in the `demos` folder of the program.

primarysectors () and multiplicities () As soon as the number of functions `#f` defined does not correspond to the number of Feynman parameters `#(z[i])+1` appearing in the functions, *primarysectors* have to be specified in the input. The first entry in each function carries a number which labels the function. If there are 5 functions but only 4 Feynman parameters, *primarysectors* has to be defined as follows:
primarysectors = 1,2,3,4,5
With that, their *multiplicities* have to be defined as well. Assuming the second function appeared twice in your calculation, but needs to be calculated only once, the definition of the input reads:
multiplicities = 1,2,1,1,1.
Analogously to the description of the *primarysectors* option in the Advanced Usage section, *primarysectors* can also be specified if only certain functions should be computed. If the functions labeled with 1, 4 and 5 should be computed, the input would read:
primarysectors = 1,4,5
multiplicities = 1,1,1.
Note: the function labeled 4 can also be second in the list of functions defined. Hence, for the specification of the *primarysectors*, the label is decisive, not the position in the functionlist.

1.3.8 Looping over ranges of parameters

In order to do the numerical integration for a whole set of numerical points, the `multinumerics` script which was present in version 2 has become obsolete in the loop setup. For this purpose, the user only needs to specify numerical values for the kinematic invariants in `kinem.input`, where each line defines a new kinematic point.

1.4 General setup

The structure of the directory `general` has changed only slightly in version 3 of the program. The command to launch SecDec is `secdec -g` similar to the `loop` case. Templates for the input files can be generated by `secdec -prep -g`.

The command `secdec -g -p <param.input> -m <math.m>` will calculate the integral using the parameters specified in `param.input`. To evaluate several points one can add the additional option `-k <kinem.input>` to the above command.

The commands

```
secdec -g -p <param.input> -m <math.m> -k <kinem.input> -algebraic
```

and
`secdec -g -p <param.input> -m <math.m> -k <kinem.input> -numerics`
will also work in the `general` setup.

- `param.input`: (text file)
  In this file the user needs to specify a name for the functions to be evaluated, the desired order in ϵ, the parameters for the numerical integration, and he can specify further options. The format is similar to `param.input` in the loop integral case, except that by default all parameters occurring in the function are specified in the file `param.input` and no `kinem.input` file is needed. However, latter can be used to facilitate parameter scans, following the syntax of the file `multiparamfile` of SecDec version 2.

- `math.m`: (Mathematica syntax)
  Contains the definition of the integrand and further options.

References

[1] T. Hahn, *CUBA: A library for multidimensional numerical integration*, *Comput. Phys. Commun.* **168** (2005) 78–95, [hep-ph/0404043].

[2] S. Agrawal, T. Hahn, and E. Mirabella, *FormCalc 7*, *J.Phys.Conf.Ser.* **368** (2012) 012054, [arXiv:1112.0124].

[3] T. Hahn, *Concurrent Cuba*, arXiv:1408.6373.

[4] S. Kawabata, *A New version of the multidimensional integration and event generation package BASES/SPRING*, *Comp. Phys. Commun.* **88** (1995) 309–326.

[5] P. Gonnet, *Increasing the reliability of adaptive quadrature using explicit interpolants*, *CoRR* **abs/1006.3962** (2010).

[6] Mathematica, Copyright by Wolfram Research.

[7] W. Bruns, B. Ichim, and C. Söger, *The power of pyramid decomposition in Normaliz*, *ArXiv e-prints* (June, 2012) [arXiv:1206.1916].

[8] W. Bruns, B. Ichim, T. Römer, and C. Söger, "Normaliz. Algorithms for rational cones and affine monoids. Available from http://www.math.uos.de/normaliz."

[9] T. Binoth and G. Heinrich, *An automatized algorithm to compute infrared divergent multi-loop integrals*, *Nucl. Phys.* **B585** (2000) 741–759, [hep-ph/0004013].

[10] J. Carter and G. Heinrich, *SecDec: A general program for sector decomposition*, *Comput.Phys.Commun.* **182** (2011) 1566–1581, [arXiv:1011.5493].

[11] 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) 396–408, [arXiv:1204.4152].

[12] S. Borowka, *Evaluation of multi-loop multi-scale integrals and phenomenological two-loop applications*, *Ph.D. thesis, Max Planck Institute for Physics/Technical University Munich* (2014) [arXiv:1410.7939].

[13] T. Binoth and G. Heinrich, *Numerical evaluation of multi-loop integrals by sector decomposition*, *Nucl. Phys.* **B680** (2004) 375–388, [hep-ph/0305234].