CutTools: a program implementing the OPP reduction method to compute one-loop amplitudes

Giovanni Ossola(1) , **Costas G. Papadopoulos**(1) , **Roberto Pittau**(2,3)

(1)Institute of Nuclear Physics, NCSR Demokritos, 15310 Athens, Greece

(2)Dipartimento di Fisica Teorica
Univ. di Torino and INFN sez. di Torino
V. P. Giuria 1, I-10125 Torino, Italy

(3)Departamento de Física Teórica y del Cosmos
Centro Andaluz de Física de Partículas Elementales (CAFPE)
Universidad de Granada, E-18071 Granada, Spain

`http://www.ugr.es/~pittau/CutTools`

Abstract

We present a program that implements the OPP reduction method to extract the coefficients of the one-loop scalar integrals from a user defined (sub)-amplitude or Feynman Diagram. Also the rational terms coming from the 4-dimensional part of the numerator are computed. Possible numerical instabilities are treated with the help of arbitrary precision routines, that activate only when needed.

Contents

1 Introduction

Developing efficient tools to compute one-loop corrections for multi-particle processes is an important task needed to cope with the complexity of LHC and ILC Physics. In the last few years a big effort has been devoted by several authors to this problem [1]. The used techniques range from analytic methods to purely numeric ones, also including semi-numerical approaches.

In the analytical approaches, computer algebra is used to reduce generic one-loop integrals into a minimal set of scalar integrals and remaining pieces (called rational terms), mainly by tensor reduction [2–5]. For multi-particle processes this method becomes quite cumbersome because of the large number of generated terms.

In the numerical or semi-numerical methods a direct computation of the tensor integrals is performed [6], capable, in principle, to deal with any configuration of masses. However, their applicability remains limited due to the high demand of computational resources and the non-existence of an efficient automation.

In a different approach, called the unitarity cut method [7], the one-loop amplitude rather than the individual integrals are evaluated, avoiding the computation of Feynman diagrams. In another development, the 4-dimensional unitarity cut method has been used for the calculation of QCD amplitudes [8], using twistor-based approaches [9]. Moreover, a generalization of the the unitarity cut method in d dimensions, has been pursued recently [10]. Nevertheless, in practice, only the part of the amplitude proportional to the loop scalar functions can be obtained straightforwardly. The remaining rational part, should then be reconstructed either by using a direct computation based on Feynman diagrams [11–13] or by using a bootstrap approach [14]. Furthermore the complexity of the calculation increases away from massless theories.

In two recent papers [15, 16], we proposed a reduction technique (`OPP`) for arbitrary one-loop sub-amplitudes at *the integrand level* [17] by exploiting numerically the set of kinematical equations for the integration momentum, that extend the quadruple, triple and double cuts used in the unitarity-cut method. The method requires a minimal information about the form of the one-loop (sub-)amplitude and therefore it is well suited for a numerical implementation. The method works for any set of internal and/or external masses, so that one is able to study the full electroweak model, without being limited to massless theories. In [18] the `OPP` method has been used, in the framework of the unitarity cut technique, to explicitly compute the subtraction terms needed not to double count the contribution of the various scalar integrals.

In this paper, we describe a `FORTRAN90` implementation of the `OPP` algorithm. In section 2, we recall the basics of the method and present our solution to compute Rational Terms and to deal with numerical inaccuracies. In section 3 we outline the conventions used in the program. In section 4 we describe the `FORTRAN90` code that implements the method and, in the last section, we discuss our conclusions. Finally, two appendices integrate the content of the paper.

2 Theory and general features

2.1 The OPP method

The starting point of the OPP reduction method is the general expression for the *integrand* of a generic m-point one-loop (sub-)amplitude [15]

$$A(\bar{q}) = \frac{N(q)}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}}\,, \quad \bar{D}_i = (\bar{q} + p_i)^2 - m_i^2\,, \quad p_0 \neq 0\,. \tag{1}$$

In the previous equation, we use a bar to denote objects living in $n = 4 + \epsilon$ dimensions, and $\bar{q}^2 = q^2 + \tilde{q}^2$, where $\tilde{q}^2$ is ϵ-dimensional and $(\tilde{q} \cdot q) = 0$. $N(q)$ is the 4-dimensional part of the numerator function of the amplitude. If needed, the ϵ-dimensional part of the numerator should be treated separately, as explained in [19]. $N(q)$ depends on the 4-dimensional denominators $D_i = (q + p_i)^2 - m_i^2$ as follows

$$\begin{aligned}
N(q) &= \sum_{i_0<i_1<i_2<i_3}^{m-1} \left[d(i_0 i_1 i_2 i_3) + \tilde{d}(q; i_0 i_1 i_2 i_3)\right] \prod_{i \neq i_0, i_1, i_2, i_3}^{m-1} D_i \\
&+ \sum_{i_0<i_1<i_2}^{m-1} \left[c(i_0 i_1 i_2) + \tilde{c}(q; i_0 i_1 i_2)\right] \prod_{i \neq i_0, i_1, i_2}^{m-1} D_i \\
&+ \sum_{i_0<i_1}^{m-1} \left[b(i_0 i_1) + \tilde{b}(q; i_0 i_1)\right] \prod_{i \neq i_0, i_1}^{m-1} D_i \\
&+ \sum_{i_0}^{m-1} \left[a(i_0) + \tilde{a}(q; i_0)\right] \prod_{i \neq i_0}^{m-1} D_i \\
&+ \tilde{P}(q) \prod_{i}^{m-1} D_i\,.
\end{aligned} \tag{2}$$

Inserted back in Eq. (1), this expression simply states the multi-pole nature of any m-point one-loop amplitude, that, clearly, contains a pole for any propagator in the loop, thus one has terms ranging from 1 to m poles. Notice that the term with no poles, namely that one proportional to $\tilde{P}(q)$ is polynomial and vanishes upon integration in dimensional regularization; therefore does not contribute to the amplitude, as it should be. The coefficients of the poles can be further split in two pieces. A piece that still depend on q (the terms $\tilde{d}, \tilde{c}, \tilde{b}, \tilde{a}$), that vanishes upon integration, and a piece that do not depend on q (the terms d, c, b, a). Such a separation is always possible, as shown in [15], and, with this choice, the latter set of coefficients is therefore immediately interpretable as the ensemble of the coefficients of all possible 4, 3, 2, 1-point one-loop functions contributing to the amplitude.

Once Eq. (2) is established, the task of computing the one-loop amplitude is then reduced to the algebraical problem of fitting the coefficients d, c, b, a by evaluating the function $N(q)$ a sufficient number of times, at different values of q, and then inverting the system. That can be achieved quite efficiently by singling out particular choices of q such that, systematically, 4, 3, 2 or 1 among all possible denominators D_i vanishes. Then the system of equations is solved iteratively. First one determines all possible 4-point functions, then the 3-point functions and so on. For example, calling $q_0^\pm$ the 2 (in general complex) solutions for which

$$D_0 = D_1 = D_2 = D_3 = 0\,, \tag{3}$$

(there are 2 solutions because of the quadratic nature of the propagators) and since the functional form of $\tilde{d}(q;0123)$ is known, one directly finds the coefficient of the box diagram containing the above 4 denominators through the two simple equations

$$N(q_0^\pm) \;=\; [d(0123) + \tilde{d}(q_0^\pm;0123)] \prod_{i\neq 0,1,2,3} D_i(q_0^\pm)\,. \tag{4}$$

This algorithm also works in the case of complex denominators, namely with complex masses. Notice that the described procedure can be performed *at the amplitude level.* One does not need to repeat the work for all Feynman diagrams, provided their sum is known: we just suppose to be able to compute $N(q)$ numerically.

The modifications one has to apply to the method when working in $d = 4+\epsilon$ dimensions are described in the next subsection.

As a further remark notice that, since the terms $\tilde{d}, \tilde{c}, \tilde{b}, \tilde{a}$ still depend on q, also the separation among terms in Eq. (2) is somehow arbitrary. Terms containing a different numbers of denominators can be shifted from one piece to the other in Eq. (2), by relaxing the requirement that the integral over the terms containing q vanishes. This fact provides an handle to cure numerical instabilities occurring at exceptional phase-space points. In `CutTools` such a mechanism is implemented for the 2-point part of the amplitude, as described in subsection 2.3 .

2.2 The rational terms

The described procedure works in 4 dimensions. However, even when starting from a perfectly finite tensor integral, the tensor reduction may eventually lead to integrals that need to be regularized [1]. Such tensors are finite, but tensor reduction iteratively leads to rank m m-point tensors with $1 \leq m \leq 5$, that are ultraviolet divergent when $m \leq 4$. For this reason, we introduced, in Eq. (1), the d-dimensional denominators $\bar{D}_i$, that differs by an amount $\tilde{q}^2$ from their 4-dimensional counterparts

$$\bar{D}_i = D_i + \tilde{q}^2\,. \tag{5}$$

The result of this is a mismatch in the cancellation of the d-dimensional denominators of Eq. (1) with the 4-dimensional ones of Eq. (2). The rational part of the amplitude, called R_1 [20], comes from such a lack of cancellation and is computed automatically in `CutTools`.

[1] We use dimensional regularization as a regulator.

A different source of Rational Terms, called R_2, can also be generated from the ϵ-dimensional part of $N(q)$ (that is missing in Eq. (1)). For the time being, it should be added by hand on the top of `CutTools`'s results by looking at the analytical structure of the Feynman Diagrams of via a dedicated set of Feynman Rules. Examples on how to compute R_2 are reported in [20].

The Rational Terms R_1 are generated by the following extra integrals, introduced in [15]

$$\begin{aligned}
\int d^n\bar{q}\frac{\tilde{q}^2}{\bar{D}_i\bar{D}_j} &= -\frac{i\pi^2}{2}\left[m_i^2+m_j^2-\frac{(p_i-p_j)^2}{3}\right]+\mathcal{O}(\epsilon)\,,\\
\int d^n\bar{q}\frac{\tilde{q}^2}{\bar{D}_i\bar{D}_j\bar{D}_k} &= -\frac{i\pi^2}{2}+\mathcal{O}(\epsilon)\,,\\
\int d^n\bar{q}\frac{\tilde{q}^4}{\bar{D}_i\bar{D}_j\bar{D}_k\bar{D}_l} &= -\frac{i\pi^2}{6}+\mathcal{O}(\epsilon)\,.
\end{aligned} \tag{6}$$

The coefficients of the above integrals are computed in `CutTools` by looking at the implicit mass dependence (namely reconstructing the $\tilde{q}^2$ dependence) in the coefficients d, c, b of the one-loop functions, once $\tilde{q}^2$ is reintroduced through the mass shift $m_i^2 \to m_i^2 - \tilde{q}^2$. One gets

$$\begin{aligned}
b(ij;\tilde{q}^2) &= b(ij)+\tilde{q}^2 b^{(2)}(ij)\,,\\
c(ijk;\tilde{q}^2) &= c(ijk)+\tilde{q}^2 c^{(2)}(ijk)\,.
\end{aligned} \tag{7}$$

Furthermore, by rewriting the first line of Eq. (2) as

$$\mathcal{D}^{(m)}(q,\tilde{q}^2) \equiv \sum_{i_0<i_1<i_2<i_3}^{m-1}\left[d(i_0i_1i_2i_3)+\tilde{d}(q;i_0i_1i_2i_3)\right]\prod_{i\neq i_0,i_1,i_2,i_3}^{m-1} D_i\,, \tag{8}$$

the following expansion holds

$$\mathcal{D}^{(m)}(q,\tilde{q}^2) = \sum_{j=2}^{m}\tilde{q}^{(2j-4)}d^{(2j-4)}(q)\,, \tag{9}$$

where the last coefficient is independent on q

$$d^{(2m-4)}(q) = d^{(2m-4)}\,. \tag{10}$$

In practice, once the 4-dimensional coefficients have been determined, `CutTools` redoes the fits for different values of $\tilde{q}^2$, in order to determine $b^{(2)}(ij)$, $c^{(2)}(ijk)$ and $d^{(2m-4)}$. Such three quantities are the coefficients of the three extra scalar integrals listed in Eq. (6), respectively.

A different way of computing $d^{(2m-4)}$ is implemented in `CutTools` when the `Logical` variable `inf` is set to `.true.` in `subroutine dp_get_coefficients` and `subroutine mp_get_coefficients`. In this case the code computes

$$d^{(2m-4)} = \lim_{\tilde{q}^2\to\infty}\frac{\mathcal{D}^{(m)}(q,\tilde{q}^2)}{\tilde{q}^{(2m-4)}}\,. \tag{11}$$

This limit is numerically quite stable and the computation faster. However, the default for `inf` is `.false.`.

2.3 Dealing with numerical inaccuracies

During the fitting procedure to determine the coefficients, numerical inaccuracies may occur due to

1) appearance of Gram determinants in the solutions for which 4, 3, 2 or 1 denominators vanish;

2) vanishing of some of the remaining denominators, when computed at a given solution;

3) instabilities occurring when solving systems of linear equations;

In principle, each of these three sources of instabilities can be cured by performing a proper expansion around the problematic Phase-Space point [2]. An attempt in this direction is described in [16]. However, this often results in a huge amount of work that, in addition, spoils the generality of the algorithm. Furthermore, one is anyway left with the problem of choosing a separation criterion to identify the region where applying the proper expansion rather than the general algorithm.

The solution implemented in `CutTools` is, instead, of a purely numerical nature and relies on a unique feature of the `OPP` method: the fact that the reduction is performed at the integral level. In detail, the `OPP` reduction is obtained when, as in Eq. (2), the numerator function $N(q)$ is rewritten in terms of denominators. Therefore $N(q)$ computed for some arbitrary value of q by using the l. h. s. of Eq. (2) should always be *numerically* equal to the result obtained by using the expansion in the r. h. s. This is a very stringent test that is applied in `CutTools` for any Phase-Space point [3]. When, in an exceptional Phase-Space point, these two numbers differ more than a user defined quantity (`limit`), the coefficients of the loop functions *for that particular point* are recomputed by using multi-precision routines (with up to 2000 digits) contained in `CutTools` [21]. The only price to be payed by the user is writing, beside the normal ones (namely written in double-precision), a multi-precision version of the routines computing $N(q)$, that is anyway easily obtained by just changing the definition of the variables used in the routines, as explained in appendix A. The described procedure ensures that the coefficients of the scalar loop functions are computed with the precision given by `limit`. This is usually sufficient; however, when strong cancellations are expected among different loop functions, a multi-precision version of the one-loop scalar functions should also be used. Then, a complete control over any kind of numerical inaccuracy is guaranteed. Finally, one should mention

[2] From now on we will denote such a point as exceptional.

[3] The arbitrary, complex, 4-vector q used for this test is randomly chosen by the code in a point by point basis.

that, usually, only very few points are potentially dangerous, namely exceptional, so that a limited fraction of additional CPU time is used to cure the numerical instabilities, therefore compensating the fact that the multi-precision routines are by far much slower than the normal ones. This procedure has been shown to work rather well in practice.

A final remark is in order. For strictly massless momenta, all Phase-Space points are exceptional in the 2-point sector. Differently stated, expressing tensors such as

$$\int d^n\bar{q}\frac{q^\mu}{\bar{D}_0\bar{D}_1} \quad \text{or} \quad \int d^n\bar{q}\frac{q^\mu q^\nu}{\bar{D}_0\bar{D}_1} \tag{12}$$

in terms of scalar 2 and 1-point functions necessarily involves the appearance of powers of $\frac{1}{(p_1-p_0)^2}$, that is always a problem when $(p_1-p_0)^2=0$.

For this reason, a different basis [16] is implemented, for the 2-point sector, in `CutTools`. This basis makes use of an arbitrary massless vector v and the code computes the coefficients of the following three scalar integrals

$$\int d^n\bar{q}\frac{[(q+p_0)\cdot v]^\ell}{\bar{D}_0\bar{D}_1} \quad \text{with} \quad \ell=0,1,2 \;\text{ and }\; v^2=0\,. \tag{13}$$

Notice that, when $k_1^2 \equiv (p_1-p_0)^2 = 0$ and $m_0=m_1$,

$$\begin{aligned}
\int d^n\bar{q}\frac{[(q+p_0)\cdot v]}{\bar{D}_0\bar{D}_1} &= -\frac{(k_1\cdot v)}{2}\int d^n\bar{q}\frac{1}{\bar{D}_0\bar{D}_1}\,, \\
\int d^n\bar{q}\frac{[(q+p_0)\cdot v]^2}{\bar{D}_0\bar{D}_1} &= \frac{(k_1\cdot v)^2}{6}\int d^n\bar{q}\frac{1}{\bar{D}_0\bar{D}_1}\,,
\end{aligned} \tag{14}$$

exactly.

3 Conventions used in the program

The information to be provided by the user is

- `number_propagators` (integer)
- `rank` (integer)
- `num(q,qt2)` (complex function)
- `den0,den1,den2,den3,den4,den5` (derived types: see below)

The first variable refers to the number of propagators in the (sub)-amplitude to be computed. The second variable is the maximum rank of $N(q)$ (not greater than `number_propagators`, condition that is anyway always fulfilled in renormalizable gauges). `num(q,qt2)` is the numerator function $N(q)$, that, when pieces of amplitude containing a different number of loop propagators are put together, also can depend on $\tilde{q}^2$, that is the second entry of the function `num(q,qt2)`.

The last line of the above list refers to a derived type defined as follows

```
module def_propagator
  implicit none
  type propagator
   integer :: i
   real(kind(1.d0)) :: m2
   real(kind(1.d0)), dimension(0:3) :: p
  end type propagator
end module def_propagator
```

Therefore, denj contains the information sufficient to denote the j^{th} loop propagator, namely squared mass and 4-momentum. These loop propagators are internally classified according to a binary notation $\texttt{denj} \rightarrow 2^j$ (following the user defined input ordering). The integer variable i of the previous derived type, is internally set to $\mathrm{i} = 2^{\mathrm{j}}$ for each propagator. In the present version of CutTools, the maximum allowed number of loop propagators is six. When less propagators are needed, they should be loaded starting from the lowest value of j.

At the end of the fitting procedure, the final results, namely the coefficients of the scalar loop functions and the rational part R_1, are loaded in the variables

$$\begin{aligned} &\texttt{dcoeff}(0,\texttt{j})\,,\ \texttt{ccoeff}(0,\texttt{j})\,, \\ &\texttt{bcoeff}(0,\texttt{j})\,,\ \texttt{bcoeff}(3,\texttt{j})\,,\ \texttt{bcoeff}(6,\texttt{j}) \quad \text{and} \quad \texttt{rat1}. \end{aligned} \tag{15}$$

The second index labels the relevant scalar loop functions, according to the above binary notation. For example the coefficient of the 3-point function

$$\int d^n\bar{q} \frac{1}{\bar{D}_0\bar{D}_2\bar{D}_4} \tag{16}$$

is $\mathrm{ccoeff}(0, 2^0 + 2^2 + 2^4) = \mathrm{ccoeff}(0, 21)$ and that one of

$$\int d^n\bar{q} \frac{1}{\bar{D}_1\bar{D}_2\bar{D}_3\bar{D}_4} \tag{17}$$

is $\texttt{dcoeff}(0, 30)$. Furthermore, $\texttt{bcoeff}(0,\texttt{j})$, $\texttt{bcoeff}(3,\texttt{j})$, and $\texttt{bcoeff}(6,\texttt{j})$ are the coefficients of the scalar integrals in Eq. (13) with $\ell = 0, 1, 2$, respectively. When $\ell \neq 0$, also the knowledge of the vector v is needed [4]. This information is stored in the array

$$\texttt{vvec}(0:3,\texttt{j})\,, \tag{18}$$

where the second index j follows the same binary notation used for the loop propagators.

Finally, when the multi-precision version of the code is activated, the relevant output information is stored in the variables:

$$\begin{aligned} &\texttt{mp\_dcoeff}(0,\texttt{j})\,,\ \texttt{mp\_ccoeff}(0,\texttt{j})\,, \\ &\texttt{mp\_bcoeff}(0,\texttt{j})\,,\ \texttt{mp\_bcoeff}(3,\texttt{j})\,,\ \texttt{mp\_bcoeff}(6,\texttt{j})\,, \\ &\texttt{mp\_vvec}(0:3,\texttt{j}) \quad \text{and} \quad \texttt{mp\_rat1}. \end{aligned} \tag{19}$$

[4]The massless vector v is determined by CutTools in an event by event basis, to maximize the numerical stability.

4 Program structure

The directory structure looks as follows:

```
avh_olo_s4.f   dynamics.f90     MPREC        README
cuttools.f90   kinematics.f90   process.f90  tensors.f90
DOC            Makefile         rambo.f      type.f90
```

In the following, we briefly discuss the content of each file or directory in the previous list.

4.1 avh_olo_s4.f

This set of routines, provided by Andre van Hameren, evaluates the scalar one-loop functions. In the current version the fully massless scalar one-loop functions are included [22].

4.2 cuttools.f90

It is the `main` program. The distributed version implements, as a simple example, the reduction of a five-point function with a "toy" numerator [5]. A test run output is given in appendix B.

The user should first initialize a few variables, such as the numbers of digits used by the multi-precision routines (`idig`), filling the internal tables of combinatorial factors by the calling the subroutine `load_combinatorics`, setting the the number of propagators for the case at hand (`number_propagators`), the maximum rank of $N(q)$ (`rank`) and the limit of precision below which the multi-precision routines activate (`limit`).

Then, for each generated Phase-Space point (the maximum number of points `nitermax` should be provided at running time), the user should define the derived types `denj` (`j = 0`, $\cdots$, `number_propagators-1`) referring to the loop propagators, and load them by calling the subroutine `load_denominators(den0,`$\cdots$`)`, with a number of arguments equal to the number of propagators.

Finally, the needed coefficients of the one-loop scalar functions and the rational part R_1 (`rat1`) [6] are obtained by calling the subroutine `get_coefficients`. At this point, if the precision test described in subsection 2.3 gives a result less then `limit`, the program multiplies all coefficients by the proper loop functions (this is achieved by calling `dp_result(dbl_prec,cutpart)`), adds the rational parts and stores the event. Otherwise the entire procedure is repeated by using multi-precision. If the test fails even using multi-precision (that may happen if `idig` is too small), the event is discarded.

At the end, the code, prints out the result of the Monte Carlo Phase-Space integration in the form of real and imaginary parts of the finite term (`sigma(0)`) and of the coefficients

[5]The routines for the evaluation of the complete one-loop QCD virtual corrections to the process $q\bar{q} \to ZZZ$ will also be available from our webpage.

[6]We recall that the rational term R_2 (`rat2`) should be computed separately.

of the $1/\epsilon$ (`sigma(1)`) and $1/\epsilon^2$ (`sigma(2)`) poles. A statistics is also provided of the percentage of points computed with multi-precision or discarded.

4.3 DOC

It is a directory containing this paper and any other updated documentation.

4.4 dynamics.f90

It is the part of the code where the user has to insert the numerator function $N(q)$, namely the complex function `num(q,qt2)`.

4.5 kinematics.f90

It is the core of `CutTools`. It contains all routines needed to perform the fits. All the output variables listed in Eq. (15), Eq. (18) and Eq. (19) are located in `module coefficients`.

4.6 Makefile

It is the `Makefile` of `CutTools`. The user should specify, among other things, the `FORTRAN90` compiler and the compilation flags he/she is using. Notice that the multi-precision library in `MPREC` should be compiled first (see next subsection).

4.7 MPREC

It is a directory containing the multi-precision package of [21]. More precisely, before compiling `CutTools`, the user should go to `/MPREC/mpfun90/f90` and give the command `make` to compile the multi-precision library.

4.8 process.f90

All routines needed to compute $N(q)$ should be put in this file.

4.9 rambo.f

It contains the random number generator and the routines for Phase-Space generation, histogramming and bookkeeping of the events.

4.10 README

It is a `.txt` file with information on the current version of `CutTools`.

4.11 tensors.f90

It contains the routines needed to perform scalar products of 4-vectors.

4.12 type.f90

It contains the `FORTRAN90` derived types used by `CutTools`.

5 Conclusion

We have presented `CutTools`, a program implementing the `OPP` reduction method [15] to extract the coefficients of the one-loop scalar integrals from a user defined numerator function (namely (sub)-amplitude or Feynman Diagram), as well as the rational terms of type R_1 [20]. The possible occurring numerical instabilities are treated with the help of arbitrary precision routines [21]. The `OPP` algorithm allowed us to implement a trivial check in order to activate the time consuming arbitrary precision routines only when necessary.

Acknowledgments

We thank Andre van Hameren for providing us with his private code `avh_olo_s4.f` to compute massless one-loop scalar integrals.

G.O. and R.P. acknowledge the financial support of the ToK Program "ALGOTOOLS" (MTKD-CD-2004-014319).

C.G.P.'s and R.P.'s research was partially supported by the RTN European Programme MRTN-CT-2006-035505 (HEPTOOLS, Tools and Precision Calculations for Physics Discoveries at Colliders).

The research of R.P. was also supported in part by MIUR under contract. 2006020509_004.

C.G.P. and R.P. thank the Galileo Galilei Institute for Theoretical Physics for the hospitality and the INFN for partial support during the completion of this work.

Appendices

A Going from double to multi-precision

All routines in `CutTools` have been written both in a normal form (namely in double-precision) and in a multi-precision form. Once a routine is written in normal form, the multi-precision version of it can be easily obtained through the following changes in the declarations statements [21]:

$$\texttt{real(kind(1.d0))} \rightarrow \texttt{type(mp\_real)}$$

$$\texttt{complex(kind(1.d0))} \rightarrow \texttt{type(mp\_complex)}\,. \tag{20}$$

The same strategy should be applied by the user to provide the multi-precision version of the routines to compute $N(q)$. Finally, an `interface` statement can be used to call both versions with the same name.

B Test run output

With `nitermax= 1`, the final output of the program reads as follows:

```
Result of the integration:

real_sigma(0)=  -67151075.5213172       +-    0.00000000000000
imag_sigma(0)=  -28426491.5346667       +-    0.00000000000000

real_sigma(1)=   3822974.11389803       +-    0.00000000000000
imag_sigma(1)=   3694493.63813566       +-    0.00000000000000

real_sigma(2)=  1.925254707235981E-029  +-    0.00000000000000
imag_sigma(2)= -1.427701757691647E-028  +-    0.00000000000000

  Statistics on the mp routines:

percentage of mp        points=   0.00000000000000
percentage of discarded points=   0.00000000000000

digits used in mp routines (if called) =          57
```

References

[1] R. K. Ellis, W. T. Giele and G. Zanderighi, JHEP **0605** (2006) 027 [arXiv:hep-ph/0602185];
R. Britto, B. Feng and P. Mastrolia, Phys. Rev. D **73** (2006) 105004 [arXiv:hep-ph/0602178];
C. F. Berger, Z. Bern, L. J. Dixon, D. Forde and D. A. Kosower, Phys. Rev. D **74** (2006) 036009 [arXiv:hep-ph/0604195];
Z. Bern, N. E. J. Bjerrum-Bohr, D. C. Dunbar and H. Ita, JHEP **0511** (2005) 027 [arXiv:hep-ph/0507019];
J. Bedford, A. Brandhuber, B. J. Spence and G. Travaglini, Nucl. Phys. B **712** (2005)

59 [arXiv:hep-th/0412108];
G. Belanger *et al.*, Phys. Lett. B **576** (2003) 152 [arXiv:hep-ph/0309010];
A. Denner, S. Dittmaier, M. Roth and M. M. Weber, Nucl. Phys. B **680**, 85 (2004) [arXiv:hep-ph/0309274];
A. Denner, S. Dittmaier, M. Roth and L. H. Wieders, Nucl. Phys. B **724** (2005) 247 [arXiv:hep-ph/0505042] and Phys. Lett. B **612** (2005) 223 [arXiv:hep-ph/0502063];
K. Kato *et al.*, PoS **HEP2005** (2006) 312;
T. Binoth, T. Gehrmann, G. Heinrich and P. Mastrolia, arXiv:hep-ph/0703311;
S. Weinzierl, arXiv:0707.3342 [hep-ph];
D. Maitre and P. Mastrolia, arXiv:0710.5559 [hep-ph];
Z. Nagy and D. E. Soper, Phys. Rev. D **74** (2006) 093006 [arXiv:hep-ph/0610028].

[2] G. 't Hooft and M. J. G. Veltman, Nucl. Phys. B **153** (1979) 365.

[3] G. Passarino and M. J. G. Veltman, Nucl. Phys. B **160** (1979) 151.

[4] A. Denner and S. Dittmaier, Nucl. Phys. B **734** (2006) 62 [arXiv:hep-ph/0509141] and Nucl. Phys. Proc. Suppl. **157** (2006) 53 [arXiv:hep-ph/0601085].

[5] T. Binoth, J. P. Guillet and G. Heinrich, Nucl. Phys. B **572** (2000) 361 [arXiv:hep-ph/9911342];
G. Devaraj and R. G. Stuart, Nucl. Phys. B **519** (1998) 483 [arXiv:hep-ph/9704308].

[6] A. Ferroglia, M. Passera, G. Passarino and S. Uccirati, Nucl. Phys. B **650** (2003) 162 [arXiv:hep-ph/0209219];
W. T. Giele and E. W. N. Glover, arXiv:hep-ph/0402152;
D. E. Soper, Phys. Rev. D **62** (2000) 014009 [arXiv:hep-ph/9910292] and Phys. Rev. D **64** (2001) 034018 [arXiv:hep-ph/0103262];
Z. Nagy and D. E. Soper, JHEP **0309** (2003) 055 [arXiv:hep-ph/0308127].

[7] Z. Bern, L. J. Dixon, D. C. Dunbar and D. A. Kosower, Nucl. Phys. B **435** (1995) 59 [arXiv:hep-ph/9409265];
Z. Bern, L. J. Dixon, D. C. Dunbar and D. A. Kosower, Nucl. Phys. B **425**, 217 (1994);
Z. Bern, L. J. Dixon and D. A. Kosower, Annals Phys. **322** (2007) 1587 [arXiv:0704.2798 [hep-ph]].

[8] R. Britto, F. Cachazo and B. Feng, Nucl. Phys. B **725**, 275 (2005).

[9] E. Witten, Commun. Math. Phys. **252**, 189 (2004);
F. Cachazo, P. Svrcek and E. Witten, JHEP **0409** (2004) 006;
A. Brandhuber, B. J. Spence and G. Travaglini, Nucl. Phys. B **706**, 150 (2005);
F. Cachazo, P. Svrcek and E. Witten, JHEP **0410**, 074 (2004);
I. Bena, Z. Bern, D. A. Kosower and R. Roiban, Phys. Rev. D **71**, 106010 (2005).

[10] C. Anastasiou, R. Britto, B. Feng, Z. Kunszt and P. Mastrolia, JHEP **0703** (2007) 111 [arXiv:hep-ph/0612277];
C. Anastasiou, R. Britto, B. Feng, Z. Kunszt and P. Mastrolia, Phys. Lett. B **645** (2007) 213 [arXiv:hep-ph/0609191];
D. Forde, Phys. Rev. D **75** (2007) 125019 [arXiv:0704.1835 [hep-ph]];
N. E. J. Bjerrum-Bohr, D. C. Dunbar and W. B. Perkins, arXiv:0709.2086 [hep-ph].

[11] T. Binoth, J. P. Guillet, G. Heinrich, E. Pilon and C. Schubert, JHEP **0510** (2005) 015 [arXiv:hep-ph/0504267].

[12] T. Binoth, J. P. Guillet and G. Heinrich, JHEP **0702** (2007) 013 [arXiv:hep-ph/0609054].

[13] Z. G. Xiao, G. Yang and C. J. Zhu, Nucl. Phys. B **758** (2006) 1 [arXiv:hep-ph/0607015];
X. Su, Z. G. Xiao, G. Yang and C. J. Zhu, Nucl. Phys. B **758** (2006) 35 [arXiv:hep-ph/0607016].

[14] Z. Bern, L. J. Dixon and D. A. Kosower, Phys. Rev. D **73** (2006) 065013 [arXiv:hep-ph/0507005];
S. D. Badger, E. W. N. Glover and K. Risager, JHEP **0707** (2007) 066 [arXiv:0704.3914 [hep-ph]].

[15] G. Ossola, C. G. Papadopoulos and R. Pittau, Nucl. Phys. B **763** (2007) 147 [arXiv:hep-ph/0609007].

[16] G. Ossola, C. G. Papadopoulos and R. Pittau, JHEP **0707** (2007) 085 [arXiv:0704.1271 [hep-ph]].

[17] F. del Aguila and R. Pittau, JHEP **0407** (2004) 017 [arXiv:hep-ph/0404120] and R. Pittau, arXiv:hep-ph/0406105.

[18] R. K. Ellis, W. T. Giele and Z. Kunszt, arXiv:0708.2398 [hep-ph].

[19] R. Pittau, Comput. Phys. Commun. **104**, 23 (1997) [arXiv:hep-ph/9607309] and **111** (1998) 48 [arXiv:hep-ph/9712418].

[20] G. Ossola, C. G. Papadopoulos and R. Pittau, in preparation.

[21] D. H. Bailey; ARPREC (C++/Fortran-90 arbitrary precision package) http://crd.lbl.gov/~dhbailey/mpdist/.
See also D. H. Bailey, "A Fortran-90 Based Multiprecision System," ACM Transactions on Mathematical Software, vol. 21, no. 4 (Dec 1995), pg. 379-387.

[22] A. van Hameren, J. Vollinga and S. Weinzierl, Eur. Phys. J. C **41** (2005) 361 [arXiv:hep-ph/0502165].