Published by Institute of Physics Publishing for SISSA/ISAS

Received: *May 27, 2004*
Revised: *July 19, 2004*
Accepted: *July 19, 2004*

Automatic integral reduction for higher order perturbative calculations

Charalampos Anastasiou*

Theory Group, MS81, SLAC
2575 Sand Hill Rd, Menlo Park, CA 94025, U.S.A.
E-mail: babis@slac.stanford.edu

Achilleas Lazopoulos†

Katholieke Universiteit Nijmegen, Theoretische Fysica
Postbus 9010, NL-6500 GL Nijmegen, The Netherlands
E-mail: lazopoul@sci.kun.nl

Abstract: We present a program for the reduction of large systems of integrals to master integrals. The algorithm was first proposed by Laporta; in this paper, we implement it in MAPLE. We also develop two new features which keep the size of intermediate expressions relatively small throughout the calculation. The program requires modest input information from the user and can be used for generic calculations in perturbation theory.

Keywords: QCD, NLO Computations.

*Research supported by the US Department of Energy under contract DE-AC03-76SF00515.
†Research supported by the E.U. contract no. HPMD-CT-2001-00105.

Contents

1. Introduction

Perturbation theory is an indispensable calculational tool in particle physics. Methods for perturbative calculations have been developed concurrently with the introduction of field theories for describing particle interactions. It is not surprising that we already have very efficient tools which confront experimental data at a quantitive level.

It is evident, however, that the current methods are not suitable for computing with sufficient accuracy all the required cross-sections at modern experiments. At the LHC or a future Linear Collider, for example, we must study a number of new complicated processes in the Standard Model or other theories. In addition, small effects arising at higher orders in perturbation theory will become significant in these experiments. It is important for such studies to improve or replace methods which require substantial human intervention. Ideally, we should develop automated methods applicable to every process, theory, and order in the perturbative expansion.

Two types of computations are generally required for the evaluation of cross-sections and decay rates: loop integrations over the momenta of virtual particles, and phase-space integrations over the momenta of particles in the final state. At higher orders in perturbation theory both tasks are hard; this is primarily due to the large number of integrals which typically appear. Unfortunately, methods for the analytic computation of loop and phase-space integrals are complicated; it is usually unrealistic to attempt a brute-force computation for all terms in the matrix-elements. A solution to this problem is to construct algorithms which reduce the number of integrals to a few master integrals, and calculate directly the master integrals only.

JHEP07(2004)046

The method of integration by parts (IBP) for the reduction of loop integrals was introduced in [1, 2]. Integrals which have common propagators (or, equivalently, belong to the same topology) satisfy linear algebraic identities. These identities can be derived with the IBP method and can be cleverly combined to produce reduction identities to master integrals. Gehrmann and Remiddi introduced a new class of identities for scalar loop integrals due to their invariance under Lorentz transformations [3]. Lorentz invariance (LI) identities are particularly useful for multiloop integrals with many external legs and massive propagators. Recently, the method of IBP and LI identities was extended to phase-space integrals that appear in the evaluation of total cross-sections [4, 5] and various differential distributions [6, 7, 8].

Because of its conceptual simplicity, the IBP method was used to construct reduction algorithms for many classes of multi-loop integrals (see for example [9, 10, 11, 12, 13]). Nevertheless, the construction of such programs was laborious and a systematic approach to produce reduction identities for arbitrary topologies was not available; this was the main reason for the rather slow pace of multiloop calculations.

This situation is now improved due to Laporta, who has proposed a fully automated method for the reduction of generic loop amplitudes [14]. In contrast to earlier approaches, his method does not attempt to derive reduction identities applicable to all the integrals of a topology. Instead, the aim is to reduce one-by-one the integrals by solving a large system of IBP/LI equations. This is achieved using Gauss elimination, after the IBP/LI system is ordered according to the complexity of the equations. Starting from the simplest one, each IBP/LI equation of the system is rearranged following a few algorithmic rules: the terms of the equation are assigned a relative weight for their complexity, and the most complicated term is then isolated on the left hand side. A recursive application of this procedure leads to expressions for complicated loop integrals in terms of master integrals.

The algorithm proposed by Laporta has already been used in a variety of calculations (for example in [15, 16, 17, 4, 5, 18, 19, 20, 7, 8, 21, 22, 23]). However, we have found that its efficient implementation in a computer program is not trivial. The main difficulties arise from the fact that typical multiloop calculations, such as the ones mentioned earlier, require an enormous number of IBP/LI equations ($10^5 - 10^6$). In the process of Gauss elimination the algorithm can produce very large expressions; one must optimize for their efficient manipulation.

In this paper, we provide a MAPLE 9 [24] computer program (AIR) based on the method of [14], for the *Automatic Integral Reduction* at higher orders in perturbation theory. The user should supply template IBP/LI equations for the integrals of a topology, optional information on the vanishing integrals of the topology and the master integrals (if known), and a small number of parameters controlling the treatment of large expressions. There is no need for advanced knowledge of the MAPLE platform. The input can be supplied with easy to modify text files, and AIR can be controlled with very simple scripts.

We believe that theorists who do not wish to invest in studying and implementing reduction methods, but need to study higher order effects in perturbation theory for various physical processes, will find this to be a valuable tool. We also hope that this publication will initiate some activity and exchange of ideas on practical issues concerning the imple-

JHEP07(2004)046

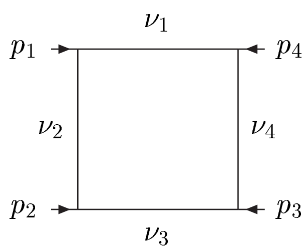
mentation of reduction algorithms. In this program, we have implemented computational tricks which were developed during practical computations; we hope our program will be improved from the experience of other users.

The cost in computer resources grows rapidly with the complexity of the study process. It is inevitable that AIR will fail to solve arbitrarily large systems of equations with large number of symbolic parameters (corresponding to kinematic scales, dimension, etc). However, we do expect AIR to be used for many applications in particle phenomenology beyond the current state-of-the-art. For this purpose, we have included routines which minimize the number of computations during the reductions, mainly by keeping the number and the size of the actively processed expressions for Gauss-elimination to a minimum.

In section 2 we review the main features of the algorithm of Laporta using the massless one-loop box integral as a pedagogical example. In section 3, we explain the main features of AIR. In the rest of the paper we demonstrate the usage of AIR through examples. In section 4 we reduce the massless one-loop box topology with no special algorithms activated for handling large expressions. In section 5 we repeat the reduction by activating a "masking" algorithm which reduces the amount of computations during Gauss elimination by storing away the expressions which get reduced in terms of master integrals. In section 6 we apply a different masking algorithm for very large integral coefficients. In section 7 we show how to reduce topologies with a very large number of kinematic scales, by switching off simplification routines. We use the reduction of the massless one-loop pentagon topology as an explicit example. Finally, we present our conclusions in section 8.

2. The reduction algorithm

In this section we present the reduction algorithm which is used in our program. A detailed description of the algorithm can also be found in ref. [14]. Here we will present its main features using the massless one-loop box topology as an explicit example.



We consider the class of integrals:

$$B(\nu_1, \nu_2, \nu_3, \nu_4) = \int d^d k \frac{1}{[k^2]^{\nu_1} \, [(k+p_1)^2]^{\nu_2} \, [(k+p_{12})^2]^{\nu_3} \, [(k+p_{123})^2]^{\nu_4}} , \qquad (2.1)$$

where we have introduced the shorthand notation $p_{ij\ldots k} = p_i + p_j + \cdots + p_k$. The terms in the denominator are raised into positive or negative integer powers ν_i. Zero powers correspond to scalar triangle and bubble integrals; negative powers correspond to triangle and bubble integrals with irreducible numerators. The external momenta are all taken to be light-like, $p_1^2 = p_2^2 = p_3^2 = p_{123}^2 = 0$. These integrals arise in one-loop QCD amplitudes for $2 \to 2$ scattering processes (e.g. $gg \to gg$ [25]).

It will be useful to know the values of the parameters ν_i for which the corresponding integrals vanish (tadpoles, scale-less bubbles). This information is not formally required; by solving the IBP equations one will eventually find that tadpoles, etc, are indeed vanishing. However, it is more efficient for the reduction to utilize the fact that many terms in the IBP equations are zero. We find the following vanishing integrals:

$$B(\nu_1, \nu_2, \nu_3, \nu_4) = 0\,, \tag{2.2}$$

if

$$\Theta(\nu_1) + \Theta(\nu_2) + \Theta(\nu_3) + \Theta(\nu_4) < 2\,, \tag{2.3}$$

or

$$\Theta(\nu_1) + \Theta(\nu_2) = 0\,, \tag{2.4}$$

or

$$\Theta(\nu_2) + \Theta(\nu_3) = 0\,, \tag{2.5}$$

or

$$\Theta(\nu_3) + \Theta(\nu_4) = 0\,, \tag{2.6}$$

or

$$\Theta(\nu_4) + \Theta(\nu_1) = 0\,, \tag{2.7}$$

where we define $\Theta(x) = 1$ for $x > 0$ and $\Theta(x) = 0$ for $x \le 0$.

Now we proceed to find algebraic equations for the integrals of the box topology. An easy way to derive such identities is the IBP method [2, 1]; we multiply the integrand with a loop or external momentum and differentiate it with the loop momentum. These total derivatives integrate to zero:

$$0 = \int d^d k \frac{\partial}{\partial k_\mu} \frac{\eta^\mu}{[k^2]^{\nu_1}\,[(k+p_1)^2]^{\nu_2}\,[(k+p_{12})^2]^{\nu_3}\,[(k+p_{123})^2]^{\nu_4}}\,, \tag{2.8}$$

where $\eta = k, k+p_1, k+p_{12}, k+p_{123}$. We obtain four IBP identities:

$$\begin{aligned}
T_1 : 0 &= \left[s\nu_1 \mathbf{1}^+ + (d - \nu_{12334}) - \left(\nu_1 \mathbf{1}^+ + \nu_2 \mathbf{2}^+ + \nu_4 \mathbf{4}^+\right) \mathbf{3}^-\right] B && (2.9)\\
T_2 : 0 &= \left[t\nu_2 \mathbf{2}^+ + (d - \nu_{12344}) - \left(\nu_1 \mathbf{1}^+ + \nu_2 \mathbf{2}^+ + \nu_3 \mathbf{3}^+\right) \mathbf{4}^-\right] B && (2.10)\\
T_3 : 0 &= \left[s\nu_3 \mathbf{3}^+ + (d - \nu_{11234}) - \left(\nu_2 \mathbf{2}^+ + \nu_3 \mathbf{3}^+ + \nu_4 \mathbf{4}^+\right) \mathbf{1}^-\right] B && (2.11)\\
T_4 : 0 &= \left[t\nu_4 \mathbf{4}^+ + (d - \nu_{12234}) - \left(\nu_2 \mathbf{1}^+ + \nu_3 \mathbf{3}^+ + \nu_4 \mathbf{4}^+\right) \mathbf{2}^-\right] B && (2.12)
\end{aligned}$$

where the action of $\mathbf{i}^+$ ($\mathbf{i}^-$) increases (decreases) ν_i by one in the integral B, e.g.

$$\mathbf{3}^\pm B = B(\nu_1, \nu_2, \nu_3 \pm 1, \nu_4)\,. \tag{2.13}$$

Products of operators have a straightforward interpretation, e.g.

$$\mathbf{3}^+\mathbf{1}^- B = B(\nu_1 - 1, \nu_2, \nu_3 + 1, \nu_4)\,. \tag{2.14}$$

We have also used the shorthand: $\nu_{ijjk\ldots} = \nu_i + 2\nu_j + \nu_k + \cdots$, and we define the usual Mandelstam variables $s = p_{12}^2$, $t = p_{23}^2$.

The IBP eqs. (2.9)–(2.12) and, optionally, the results of eqs. (2.3)–(2.7) are sufficient to reduce any integral of the box topology to master integrals by using the algorithm of Laporta [14]. In ref. [14], the reader can find a detailed and complete description of the algorithm; here we intend to emphasize its salient features. The user is not required to have knowledge of the algorithm, however, some familiarity will be beneficial. We will describe the algorithm by tracing the first steps of our code when solving the box topology. For concreteness, we will stop when the integral $B(1,-1,1,0)$ is reduced in terms of master integrals.

- *Seed generation*: the program starts with the simplest list $(\nu_1, \nu_2, \nu_3, \nu_4)$ for which $B(\nu_1, \nu_2, \nu_3, \nu_4)$ is not vanishing,

  $$S_1 : (\nu_1, \nu_2, \nu_3, \nu_4) = (1, 0, 1, 0)\,. \tag{2.15}$$

  S_1 will be our first 'seed' for generating identities from the topology IBP equations $T_1, \ldots, T_4$ (eqs. (2.9)–(2.12)), which we use as templates.

- *Identities generated from the template IBP equations*: our first identity is eq. T_1 substituting the values of $\{\nu_i\}$ found in S_1:

  $$E_1^1 : sB(2,0,1,0) + (d-3)B(1,0,1,0) = 0\,. \tag{2.16}$$

  In E_1^1, we have already used our knowledge for the vanishing integrals of the topology (eqs. (2.3)–(2.7)).

  The above equation can be recast to express one of the two integrals in terms of the other. We would like to use such equations to express more complicated integrals in terms of simpler ones and, finally, in terms of the master integrals. It is therefore necessary to introduce criteria for the complexity of the integrals; the most complicated should receive first priority and will be isolated in the left hand side.

- *Integral priority criteria*: we check on three parameters in order to isolate the most complicated integral. First we select the integrals with the largest number of propagators:

  $$\mathcal{N}_{\rm prop} = \sum_i \Theta(\nu_i). \tag{2.17}$$

  If more that one integral has the maximum $\mathcal{N}_{\rm prop}$, we select the one with the largest sum of positive indices ν_i

  $$\mathcal{N}_+ = \sum_i \Theta(\nu_i)(\nu_i - 1)\,. \tag{2.18}$$

  If more than one integral has the maximum values of $N_{\rm prop}$ and N_+, we select the one with the largest sum for the magnitudes of negative indices ν_i

  $$\mathcal{N}_- = -\sum_i \Theta(-\nu_i)\nu_i\,. \tag{2.19}$$

  If still there is an ambiguity, we randomly choose one of the integrals which has survived all three criteria.

- *Rearranging the identities*: following the previous criteria, we find that $B(2,0,1,0)$ is the most complicated integral in E_1^1. We then rearrange the identity to produce:

$$E_1^1 : B(2,0,1,0) = \frac{3-d}{s} B(1,0,1,0)\,. \tag{2.20}$$

We proceed, in the same manner, with the remaining identities T_2, T_3, T_4 for S_1. We obtain:

$$E_1^2 : B(2,0,1,-1) = (d-2)B(1,0,1,0) - B(1,0,2,-1), \tag{2.21}$$

$$E_1^3 : B(1,0,2,0) = \frac{3-d}{s} B(1,0,1,0)\,, \tag{2.22}$$

and

$$E_1^4 : B(2,-1,1,0) = (d-2)B(1,0,1,0) - B(1,-1,2,0)\,. \tag{2.23}$$

- *Seed priority criteria:* we have now processed all IBP equations for the first seed S_1. It is therefore necessary to choose a new seed to obtain more identities. It is important to choose seeds that are most likely to produce equations coupled with the ones which have been processed earlier. For this purpose, we could select the seeds with the opposite priorities than the integral priorities, i.e. the seed with successively minimum values for $[\mathcal{N}_{\mathrm{prop}}, \mathcal{N}_-, \mathcal{N}_+]$. However, the rules for choosing the seeds are mostly empirical and require some experimentation. In fact, the order for applying the criteria for minimum $\mathcal{N}_-$ and $\mathcal{N}_+$ can be judiciously chosen according to the class of integrals that the user needs to compute. For example, we could now pick either $(1,-1,1,0)$ or $(2,0,1,0)$ as the next seed. Since our goal is to compute $B(1,-1,1,0)$ it is better to choose:

$$S_2 : (\nu_1, \nu_2, \nu_3, \nu_4) = (1,-1,1,0)\,. \tag{2.24}$$

which in the IBP equations generates integrals with the same structure as the one we want to solve. Our program generates the seeds automatically; the user must provide the range of $\mathcal{N}_{\mathrm{prop}}, \mathcal{N}_-, \mathcal{N}_+$ as input. It is relatively straightforward to decide the values for these parameters by inspecting the integrals that are required in the study process.

- *Substitutions and Gauss-elimination:* we now find a new feature in eq. T_1 for the seed S_2:

$$E_2^1 : sB(2,-1,1,0) + (d-2)B(1,-1,1,0) = 0\,. \tag{2.25}$$

The integral $B(2,-1,1,0)$ is isolated at the left hand side (lhs) of a previous equation (E_1^4). In such cases, we eliminate the known integral from the equation. Substituting eq. (2.23), and applying the integral priority criteria, we have:

$$E_2^1 : B(1,-1,2,0) = \frac{d-2}{s} B(1,-1,1,0) + (d-2)B(1,0,1,0)\,. \tag{2.26}$$

- *Back-substitution:* E_2^1 is solved in terms of an integral that can be substituted back to E_1^4. We can now see how the Laporta algorithm works in practice; by adding new equations to the already solved equations we form new sub-systems of coupled equations which eliminate previously unknown integrals. In our example, performing the substitution of E_2^1 into E_1^4 we obtain,

$$E_1^4 : B(2,-1,1,0) = \frac{2-d}{s} B(1,-1,1,0)\,. \tag{2.27}$$

  We process two more equations for the seed S_2:

$$\begin{aligned} E_2^2 :\ & B(2,-1,1,-1) = -tB(1,0,1,0) + (d-1)B(1,-1,1,0) + \\ & +B(1,0,1,-1) - B(1,-1,2,-1)\,, \end{aligned} \tag{2.28}$$

  and

$$E_2^3 : B(1,-1,1,0) = -\frac{s}{2} B(1,0,1,0)\,. \tag{2.29}$$

  In the last equation we have computed the integral that we wanted in terms of a simpler one: $B(1,0,1,0)$. It is clear from the previous equations that $B(1,0,1,0)$ is a master integral.

In summary, the algorithm requires the successive generation of identities with terms of increasing complexity. The newly added equations usually contain terms which are also found in equations generated at earlier stages; this produces small subsystems of coupled algebraic identities. A series of substitutions diagonalizes these algebraic subsystems and yields complicated integrals expressed in terms of master integrals. The algorithm is a clever implementation of Gauss elimination. It exploits the fact that Feynman integrals can be ordered according to very simple criteria.

We demonstrated how the algorithm reduces a number of integrals belonging to the massless one-loop box topology. However, there was no step in the previous reduction that depended on the specifics of the topology. Therefore, this algorithm is suitable for the reduction of generic multiloop integrals or, more generally, of parametric functions which satisfy coupled algebraic identities (e.g. hypergeometric functions).

3. Features of AIR

In this section we describe the basic functions of our program. The program is included as a gzipped and tarred file in the source submission of the electronic preprint for this paper, and can also be downloaded from ref. [26]. It is convenient to unzip and untar the distribution file in a directory where AIR can be located permanently.

```
/home> tar -zxvf air.tar.gz
AIR/
AIR/main.map
AIR/BOXA/
AIR/BOXA/input_boxa.map
AIR/BOXA/script_boxa.map
```

```
AIR/BOXB/
AIR/BOXB/input_boxb.map
AIR/BOXB/script_boxb.map
AIR/BOXC/
AIR/BOXC/input_boxc.map
AIR/BOXC/script_boxc.map
AIR/Pentagon/
AIR/Pentagon/input_pentagon.map
AIR/Pentagon/script_pentagon.map
AIR/Pentagon5/
AIR/Pentagon5/input_pentagon.map
AIR/Pentagon5/script_pentagon.map
```

The distribution includes the program file *main.map*, input files *input_⋯.map*, as well as MAPLE scripts *script_⋯.map* for the example reductions in the rest of the paper. The program consists of MAPLE routines for generating seeds for the template IBP/LI identities, finding integral priorities, generating the IBP equations from the seeds, performing Gauss-elimination, masking large integral coefficients and reduced expressions, performing nested substitutions, and collecting the results. The function of the more important routines will be detailed in the following sections.

In typical multiloop computations, a large number of identities should be processed. A database system is therefore required to access, modify, and store the equations. We have implemented a rather simple database system, where each IBP equation is stored in a single file; the name of the file is determined by the integral on the lhs of the equation. We also create separate auxiliary files which serve to point to the equations in the IBP system where a particular integral can be found. Our database system is very robust; however, it creates a rather extended tree of directories which usually contain very short ASCII files.

The program can perform very complicated multiloop reductions. It is often possible to simplify all the terms in the IBP equations as they get substituted and rearranged for Gauss elimination. However, if the topology depends on many kinematic scales, or the IBP equations are loosely coupled (creating large subsystems of equations before they get diagonalized), or the values for $\mathcal{N}_{\rm prop}, \mathcal{N}_{\pm}$ are large, it may not be feasible to perform all simplifications within acceptable times or the available memory. We have implemented two algorithms to perform the reductions efficiently and reduce the amount of computations; the algorithms can be used independently or in conjuction.

The first algorithm masks subexpressions which are reduced in terms of master integrals. The program detects the reduced expressions, stores them in files, and replaces them by an indexed symbol. Thus, the masked expressions are protected from subsequent manipulations during Gauss elimination. This feature is implemented recursively; whenever a new expression is written in terms of masked expressions and/or master integrals, the new expression is also masked. At the end of the reduction, a series of nested substitutions is required in order to rewrite the masked expressions in terms of the master integrals. We will discuss the required nested substitutions later; for now, we should note that the masking algorithm reduces significantly the amount of computations during the process of Gauss elimination. The algorithm requires that the master integrals of the topology are known. To determine them, one can perform a less involved reduction for relatively small values of $\mathcal{N}_\pm$ without using the masking algorithm. When the master integrals are found, the user can repeat the reduction for larger values of $\mathcal{N}_\pm$, activating the masking algorithm.

The second algorithm aims to reduce the size of the equations by masking all integral coefficients which are lengthier than a user-defined maximum value. During Gauss-elimination, however, some integral coefficients vanish; the elimination cannot take place if the coefficients contain masked expressions. To solve this problem we check numerically for cancellations. The user needs to provide as input, numerical values for all the parameters (kinematic scales, dimension) which enter in the symbolic expressions for the integral coefficients of the IBP/LI equations. The masking algorithm substitutes these numerical values and stores both the numerical result and the symbolic expression for the coefficients. The program determines if a coefficient is zero by inspecting the numerical result, thus, avoiding complicated symbolic manipulations. To ensure that cancellations are not accidental, the program can perform the numerical testing of the values of the coefficients for more than one choices of numerical values for the kinematic parameters and the dimension. The analytical value of the lengthy coefficients is computed at the end of the reduction, and only for the integrals that are required for practical purposes.

The purpose of the two algorithms is to remove complications from the symbolic manipulation of very large expressions. However, after Gauss-elimination we must still perform computations which were deferred by using the masking algorithms, i.e. we must perform a series of nested symbolic substitutions for the used alias symbols in order to compute the masked expressions explicitly in terms of the kinematic parameters, the dimension, and the master integrals. It is possible to imagine that this additional computation is as difficult as using the program without the masking algorithms, where all substitutions take place explicitly during Gauss elimination. This is not the case; usually, only a fraction of the total number of masked expressions is required for the integrals that appear in the matrix-elements of a physical process. For example, the integral $B(2,-1,1,0)$ of the previous section does not appear in the computation of e.g. $gg \to gg$, however, it appears in the IBP equations. By using the masking algorithms, we avoid computing the masked expressions for many such integrals.

The remaining nested substitutions can still be challenging for very complicated problems. One can resort to tricks such as expanding in the dimension parameter [9] or in kinematic parameters (e.g. electron mass in Bhabba scattering), if this is justified from the physics of the process. However, this is rarely needed; there are many processes where we can perform the substitutions without giving up on a valid evaluation of the integral coefficients for all values of the kinematic parameters and the dimension. AIR includes general purpose routines for a straightforward computation of recursive substitutions; these routines attempt a brute-force symbolic simplification of all the intermediate expressions. It also provides the option to switch off simplification of expressions that exceed a maximum length or, if necessary, to transfer the most complicated substitutions to another platform, e.g. FORM [27].

In the following sections we perform four example reductions which can serve as a tutorial for using AIR and its main features. A technical description of the AIR routines can be found in [26]. The source code of the program is openly distributed; the users are free to modify it. The authors will be grateful to receive suggestions and constructive feedback.

4. Reduction with no masking

We now perform the reduction of the massless one-loop box topology. In this section we do not activate any of the two masking algorithms. The input file and the corresponding script for the reduction can be found in the directory:

```
/home/AIR/BOXA
```

The input file for the reduction is named `input_boxa.map`. It contains variables which are used globally by AIR. These are:

```
ibp_equations:= [
-nu3*B(nu1-1,nu2,nu3+1,nu4)-nu4*B(nu1-1,nu2,nu3,nu4+1)
-nu2*B(nu1-1,nu2+1,nu3,nu4)+nu3*s*B(nu1,nu2,nu3+1,nu4)
+(-nu3-nu2-2*nu1+d-nu4)*B(nu1,nu2,nu3,nu4),
nu4*t*B(nu1,nu2,nu3,nu4+1)-nu3*B(nu1,nu2-1,nu3+1,nu4)
+(d-2*nu2-nu3-nu1-nu4)*B(nu1,nu2,nu3,nu4)
-nu4*B(nu1,nu2-1,nu3,nu4+1)-nu1*B(nu1+1,nu2-1,nu3,nu4),
-nu1*B(nu1+1,nu2,nu3-1,nu4)+(-nu2+d-nu4-nu1-2*nu3)
*B(nu1,nu2,nu3,nu4)-nu4*B(nu1,nu2,nu3-1,nu4+1)
-nu2*B(nu1,nu2+1,nu3-1,nu4)+nu1*s*B(nu1+1,nu2,nu3,nu4),
-nu1*B(nu1+1,nu2,nu3,nu4-1)-nu2*B(nu1,nu2+1,nu3,nu4-1)
-nu3*B(nu1,nu2,nu3+1,nu4-1)+nu2*t*B(nu1,nu2+1,nu3,nu4)
+(-nu3+d-nu2-nu1-2*nu4)*B(nu1,nu2,nu3,nu4)
]:
```

- This is a list of template IBP identities (eqs. (2.9)-(2.12)) for the box topology. The program reads off some additional implicit definitions from the structure of the IBP equations. For example, it is now defined that the name of the topology is `''B''` and the powers of the propagators are defined through the variables `nu1, nu2, nu3, nu4`.

```
ZERO_TOPOLOGIES:=[
ThetaF(nu1) + ThetaF(nu2) + ThetaF(nu3) +ThetaF(nu4) < 2,
ThetaF(nu1) +ThetaF(nu2) =0,
ThetaF(nu2) +ThetaF(nu3) =0,
ThetaF(nu3) +ThetaF(nu4) =0,
ThetaF(nu4) +ThetaF(nu1) =0,
NULL]:
```

- This is a list of statements (eq. (2.3)–(2.7)) which undergo boolean evaluation when the propagator powers are substituted by integers. If any of the statements is true, then the corresponding integral is set to zero.

```
MASTERS:=[]:
```


JHEP07(2004)046

- This variable activates the algorithm for masking reduced expressions to master integrals. It contains a list of known master integrals. In this example, we do not want to activate the algorithm; therefore we define the above variable to be an empty list.

```
check_values:=[]:
```

- This variable activates the masking algorithm for integral coefficients which exceed a maximum value. It should contain numerical values for all the kinematic parameters and the dimension in the IBP/LI equations. For this example, we do not want to activate the masking algorithm and we set the variable to an empty list.

```
MAXLENGTH:=1000:
```

- This variable is used by the masking algorithm for large integral coefficients. It defines the maximum length for a coefficient in order not to get masked. The length of an expression is measured by the number of characters in the expression and is determined by a MAPLE routine. The value of the variable is irrelevant if the `check_values` list is empty.

```
MAXSIMPLIFY:=10^10:
```

- This variable is used by the routines which perform nested substitutions for the masked expressions and the routines which display the final results. It sets a maximum length for the expressions that MAPLE is allowed to simplify. Larger expressions get substituted but not simplified.

```
MAPLEMAXSUB:=10^10:
```

- This variable is used by the routines which perform nested substitutions of masked expressions, and the routines which display the results. It sets a maximum length for the expressions that MAPLE is allowed to substitute. Larger expressions are not written explicitly and are kept masked. In this reduction we want all coefficients to be explicit, and we set the value of the variable to a practically unreachable value.

```
VERBOSE:=FALSE:
```

- This variable is used to display information about the progress of the program. If set to TRUE, the program outputs on the screen the seed that is processing, or, after Gauss-elimination is completed, the index of the masked expression that is evaluating.

The file `script_boxa.map` contains all the calls to AIR for reducing the box topology. We can run the script from the shell command line:

```
/home/AIR/BOXA> maple script_boxa.map
```

It will be more instructive for this first application to call the routines interactively from within the MAPLE platform. We first launch MAPLE,

```
/home/AIR/BOXA> maple
```

and load the input for the topology and the main program:

```
 > currentdir(``/home/AIR/BOXA''): > read
  ``input_boxa.map'': > read ``/home/AIR/main.map'':
```

The user should now perform the following tasks:

- *Seed generation*: we create a list with sets of integers for deriving IBP equations from the template equations of `ibp_equations`. This is accomplished by calling the routine

  ```
  SEEDGEN(``filename'', maxtop, [minNprop,maxNprop], [minNminus,
                 maxNminus], [minNplus, maxNplus]) ;
  ```

  The first argument is the filename where the seeds will be written. The next argument, `maxtop`, is a list with integers $\{i\}$ denoting the propagators raised to positive powers $\{\nu_i\}$ in the seed with the highest priority. For example, if we are interested in reducing all the integrals of the box topology and all its subtopologies we should set `maxtop` $= [1, 2, 3, 4]$, indicating that all $\nu_1, \nu_2, \nu_3, \nu_4$ can appear with positive values. If we only required integrals of e.g. a t-channel triangle subtopology, we could set `maxtop` $= [1, 3, 4]$, indicating that seeds with positive ν_2 do not need to be included in the reduction. The next three entries determine the range of $\mathcal{N}_{\rm prop}$, $\mathcal{N}_-$, and $\mathcal{N}_+$ repsectively. These values are mostly empirical. A rule of thumb is that one should generate seeds which include the indices of the most complicated integrals to be reduced, as well as the complete tower of integrals with lower priorities. To give a concrete example, we will generate seeds for the integrals that appear in the one-loop $gg \to gg$ amplitude. We find integrals from all subtopologies (bubble and triangles) of the box topology, therefore we set $2 \leq \mathcal{N}_{\rm prop} \leq 4$. We also find that all integrals have $\mathcal{N}_+ = 0$ (there are no squared propagators in the amplitude), and can have up to 4 powers of irreducible numerators: $0 \leq \mathcal{N}_- \leq 4$. To emphasize differences in the running times for various algorithms of the program we will extend the latter interval to $0 \leq \mathcal{N}_- \leq 10$.

  ```
  > SEEDGEN(``seeds.map'', [1, 2, 3, 4], [2, 4], [0, 10], [0, 0]);
  ```

  The routine produces a list of ordered seeds in the file `/home/AIR/BOXA/seeds.map`.

- *Gauss elimination*: we generate the IBP equations from the seeds and perform Gauss elimination by calling the routine

  ```
  Reducer(``seeds_file'', ``monitor_file'', ``RESULTS_DIR'');
  ```

  The first argument is the name of the file with the seeds for the reduction (as it was produced with `SEEDGEN`). The next argument is the name of a file which the program updates with the processed seeds; it serves to monitor the progress of the program. The last argument provides a directory path where the program can deposit the database with the IBP equations. For our example, we type

```
> Reducer(``seeds.map'', ``calc.map'', ``.'');
```

It is useful to inspect the /home/AIR/BOXA directory. The program has created a number of subdirectories (B_1_3, B_1_3_4, etc) which correspond to the non-vanishing subtopologies of the box topology. In these subdirectories, AIR stores the IBP equations after they have been rearranged to isolate the most complicated integral at the lhs of the equation. For example, eq. (2.29) is stored in

```
/home/AIR/BOXA/B_1_3/B_1_-1_1_0.map
```

The first part of the file path /home/AIR/BOXA/ corresponds to the directory in the third argument of the Reducer command. The second part of the file path B_1_3 is created from the integers $\{i\}$ for which the powers $\{\nu_i\}$ are positive. The ending of the file path is created from the indices $\{\nu_i\}$ of the integral.

- *Collecting the results*: Essentially the reduction is now complete; the reduced integrals can be found in the files of the database tree for the IBP equations. Inspecting some of the equations we find that many integrals are reduced in terms of three master integrals: $B(1,0,1,0)$, $B(0,1,0,1)$, and $B(1,1,1,1)$. However, we also find integrals which are not fully reduced, e.g. the integral $B(2,1,1,-11)$. It is usually observed that integrals with the same indices as in the seeds are fully reduced. Motivated from this observation, we have written a routine for collecting in a separate directory the seed integrals:

  ```
  tidy_list(``seeds_file'', ``RESULTS_DIR'' );
  ```

  The first argument is a file with seeds (as generated by SEEDGEN). The last argument is the directory where the program has placed the results (as in the third argument for Reducer). For our example, we type:

  ```
  > tidy_list(``seeds.map'', ``.'');
  ```

  The routine has created a subdirectory, named RESULTS, which contains the expressions for the seed integrals only. For example, one can find the integral $B(1,-4,1,1)$ in the file /home/AIR/BOXA/RESULTS/B_1_3_4/B_1_-4_1_1.map. We should note that when the masking algorithms are activated, the tidy_list routine also performs the required nested substitutions for expressing the coefficients of the master integrals in terms of the kinematic parameters and the dimension. However, it does so only for coefficients with smaller size than the MAPLEMAXSUB value. It is possible to achieve a fast execution of the tidy_list routine if we choose a low value (100-500) of MAPLEMAXSUB. The task of computing large coefficients is postponed further, and is performed by a new routine which we shall describe shortly. This routine is customized to disentangle the newest masking, which has a simpler structure than the masking of Gauss-elimination, very efficiently.

- *Reading the results interactively:* A useful routine for reading the reduced integrals from within the MAPLE environment is

```
show_int(integral, RESULTS_DIR);
```

The first argument is the required integral, and the second is the directory of the reduction (as in `Reducer`). For example,

```
> show_int(B(1, -4, 1, 1), ‘‘.’’);
```

outputs the expression for the required integral in terms of master integrals. This routine can be used for purposes of interfacing the results of the reduction to other programs that users develop for calculating matrix-elements. We should note that if the masking of master integrals is activated and the value of `MAPLEMAXSUB` is small, `show_int` will return the wanted integral as a linear combination of master integrals but with masked coefficients. The routine `show_full_int` displays the unmasked result.

We have described the basic variables and routines of AIR by performing the reduction of the box topology. It is worth noting that the total running time for the three main routines (SEEDGEN, Reducer, tidy_list) is approximately two minutes on a 1.6GHz processor.

5. Masking reduced expressions

In this section we repeat the reduction of the one-loop massless box topology providing the known master integrals as input to the program. The algorithm uses this information to find parts of expressions which are reduced to master integrals, and masks them. The reduction proceeds faster, having replaced the masked expressions by indexed symbols $K(i)$. When the step of Gauss elimination is completed, only the indexed symbols in the expressions of the seed integrals need to be computed explicitly.

The reduction is performed in the directory `/home/AIR/BOXB`. The program allows, in principle, activation of the masking algorithm for master integrals without changing directory. However, we recommend performing the reduction in a new directory if new values for the variables `ZERO_TOPOLOGIES`, `ibp_equations`, `MASTERS`, `check_values`, and `MAXLENGTH` are required. This will prevent possible integral misidentifications caused from creating the database of IBP equations with inconsistent values for these variables.

The input file `input_boxb.map` is modified to activate the masking algorithm for master integrals. In particular, the variable for the master integrals is not empty; we have defined

```
MASTERS:=[
B(1,0,1,0),
B(0,1,0,1),
B(1,1,1,1)]:
```

These are the master integrals that we found in the first reduction.

We now execute the MAPLE script in `script_boxb.map`:

```
currentdir("/home/AIR/BOXB"):
read "input_boxb.map":
read "/home/AIR/main.map":
```

```
SEEDGEN("seeds.map", [1, 2, 3, 4], [2, 4], [0, 10], [0, 0]):
Reducer("seeds.map", "calc.map", "."):
tidy_list("seeds.map", "."):
```

The script is executed in less than a minute; this is approximately 50% faster than without the masking algorithm. The routine `Reducer` has created a tree of subdirectories `ICED/ICED#/KEXPR`, where the masked expressions are stored in files named `kexpr_i.map`. The expressions are replaced in the reduction by the symbol $K(i)$, where i is an integer index.

The masked expressions are defined recursively; it is therefore necessary to perform a series of nested substitutions before we obtain their explicit form in terms of master integrals. This task is performed by the `tidy_list` routine, which stores explicit results for the masked expressions in the directory `/home/AIR/BOXB/KMELT`. The user is not required to know the details about the file structure for the masked expressions; the integrals in the directory `RESULTS` are fully evaluated in terms of master integrals and can be accessed as before.

6. Masking large integral coefficients

In this section we describe the function of the algorithm which masks large integral coefficients in the IBP equations. We perform the reduction in the directory `/home/AIR/BOXC`, which contains the input file `input_boxc.map` and the appropriate MAPLE script `script_boxc.map`. We have modified the input variables:

```
MASTERS:=[]:
check_values:=[
[s=1,t=-0.12,d=3.3],
[s=-1.2, t=12.2, d=42],
[s=1.82, t=-0.345, d=-28.1]
]:
MAXLENGTH:=50:
```

In this run we have deactivated the algorithm for masking reduced expressions. Instead, we have provided a list of numerical values for the kinematic parameters and the dimension which activates the algorithm for masking the integral coefficients with length bigger that the value of the `MAXLENGTH` variable. The MAPLE commands for the reduction are collected in the file `script_boxc.map`,

```
restart;
currentdir("/home/AIR/BOXC"):
read "input_boxc.map":
read "/home/AIR/main.map":
SEEDGEN("seeds.map", [1, 2, 3, 4], [2, 4], [0, 10], [0, 0]):
Reducer("seeds.map", "calc.map", "."):
tidy_list("seeds.map", "."):
```

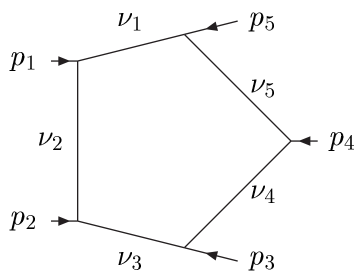
The script is executed in about 90 seconds; this is slower than previously. In general, masking large coefficients is not as fast as masking the reduced expressions to master integrals. However, in reductions of complicated IBP systems, such as in multiloop crossed topologies, we have found that this algorithm is indispensable. In extremely complicated problems it is required that both masking algorithms are activated.

In the directory `ICED` we find two new subdirectories: `EXPR` and `NUM`. The routine `Reducer` saves the expressions for the masked large integral coefficients in the first directory. In the second directory, it stores the numerical values of the same coefficients for the choices of the parameters in `check_values`. Expressions in the directory `EXPR` are defined recursively through other masked expressions. However, in the directory `NUM` they are explicitly evaluated for the special input values of the parameters. The program sets an expression to zero in the IBP equations if its numerical values is zero for all input choices in `check_values`. It is important to provide "sufficiently random" lists of numerical values for the parameters in order to reduce the risk for accidental cancellations. It is important to note that floating point numbers in `check_values` are automatically converted to fractional numbers when used in the program. Therefore, the numerical evaluation of coefficients is exact (using integer arithmetics), avoiding complications due to rounding.

Finally, the results are collected in the directory `RESULTS` using the routines `tidy_list` and `show_int`. The integrals in the `RESULTS` directory are fully reduced to master integrals with all their coefficients explicitly written as combinations of the kinematical variables.

7. Reduction of one-loop pentagons

In this section we perform the reduction of the one-loop pentagon with massless propagators. This is an example of a topology with many kinematic parameters. As is common in such topologies, the expressions for the reduced integrals are large. We will therefore use this example to demonstrate the options in AIR for dealing with large expressions.



We consider the class of integrals:

$$V(\nu_1,\nu_2,\nu_3,\nu_4,\nu_5) = \int d^dk \frac{1}{[k^2]^{\nu_1}\,[(k+p_1)^2]^{\nu_2}\,[(k+p_{12})^2]^{\nu_3}\,[(k+p_{123})^2]^{\nu_4}\,[(k+p_{1234})^2]^{\nu_5}}\,, \tag{7.1}$$

where $p_{12345}=0$ and $p_1^2=p_2^2=p_3^2=p_4^2=p_5^2=0$. The IBP equations for the pentagon topology are:

$$\nu_1\mathbf{1}^+s_{12}+\nu_5\mathbf{5}^+s_{34}+(d-\nu_{123345})-\left(\nu_1\mathbf{1}^++\nu_2\mathbf{2}^++\nu_4\mathbf{4}^++\nu_5\mathbf{5}^+\right)\mathbf{3}^- = 0 \tag{7.2}$$

$$\nu_2\mathbf{2}^+s_{23}+\nu_1\mathbf{1}^+s_{45}+(d-\nu_{123445})-\left(\nu_1\mathbf{1}^++\nu_2\mathbf{2}^++\nu_3\mathbf{3}^++\nu_5\mathbf{5}^+\right)\mathbf{4}^- = 0 \tag{7.3}$$

$$\nu_3\mathbf{3}^+s_{34}+\nu_2\mathbf{2}^+s_{51}+(d-\nu_{123455})-\left(\nu_1\mathbf{1}^++\nu_2\mathbf{2}^++\nu_3\mathbf{3}^++\nu_4\mathbf{4}^+\right)\mathbf{5}^- = 0 \tag{7.4}$$
$$\nu_4\mathbf{4}^+s_{45}+\nu_3\mathbf{3}^+s_{12}+(d-\nu_{112345})-\left(\nu_2\mathbf{2}^++\nu_3\mathbf{3}^++\nu_4\mathbf{4}^++\nu_5\mathbf{5}^+\right)\mathbf{1}^- = 0 \tag{7.5}$$
$$\nu_5\mathbf{5}^+s_{51}+\nu_4\mathbf{4}^+s_{23}+(d-\nu_{122345})-\left(\nu_1\mathbf{1}^++\nu_3\mathbf{3}^++\nu_4\mathbf{4}^++\nu_5\mathbf{5}^+\right)\mathbf{2}^- = 0 \tag{7.6}$$

where we denote the invariant masses $s_{ij}\equiv p_{ij}^2$. We also note that the topology vanishes if any three adjacent propagators are raised to non-positive powers.

After a preliminary run with no masking algorithms we find 11 master integrals:

- the pentagon integral $V(1,1,1,1,1)$,
- the box integrals $V(1,1,1,1,0)$, $V(1,1,1,0,1)$, $V(1,1,0,1,1)$, $V(1,0,1,1,1)$, $V(0,1,1,1,1)$, and
- the bubble integrals $V(1,0,1,0,0)$, $V(1,0,0,1,0)$, $V(0,1,0,1,0)$, $V(0,1,0,0,1)$, and $V(0,0,1,0,1)$.

Remarkably, by using the algorithm for masking reduced expressions, the step of Gauss elimination is very fast and can be performed for far more complicated integrals than the ones that are practically needed; it only takes a few minutes before a sufficiently large number of integrals for QCD five-point amplitudes is solved. However, keeping the variable `MAPLEMAXSUB` in unreachable values (i.e. switching off the special routines for large coefficients, as we did in all previous examples), forces the routine `tidy_int` to carry the load of all substitutions and simplifications. The presence of five independent Mandelstam variables and the dimension parameter makes the step of nested substitutions very hard for the integrals with highest priority. The routine can get stalled in computers with a memory smaller than 1GB.

We set the variable `MAPLEMAXSUB` to a value of 100 in the `input_pentagon.map`. The reduction routine `Reducer` is not affected by the new setting, however, the nested substitution routine `tidy_list` runs differently; it does not perform simplifications on any expression with more than 100 characters. Instead, it replaces these coefficients with indexed aliases, $\{f[i]\}$. The routine does not process large coefficients, and finishes very quickly. The difficult substitutions for the masked expressions $f[i]$ are performed with a new routine `melt_all_f(".")`. This routine is designed to work with the smallest possible memory consumption. Finally, the user can display the seed integrals explicitly written in terms of master integrals, the dimension, and the kinematic parameters, using the `show_full_int` routine.

It is worth demonstrating some important technical details for the reduction. We perform the reduction in the directory `/home/AIR/Pentagon`, where we have placed the input file `input_pentagon.map`, and a script for executing the AIR routines `script_pentagon.map`. As usual, in the input file we have provided the IBP equations, conditions for vanishing integrals, and the list of master integrals. We also set values for the variables,

```
MAXLENGTH:=100:
MAXSIMPLIFY:=1500:
MAPLEMAXSUB:=100:
VERBOSE:=FALSE:
```

We now perform the reduction of integrals of the pentagon topology with $\mathcal{N}_- \leq 4$, and $\mathcal{N}_+ = 0$.

```
\home\AIR\Pentagon> maple script_pentagon.map
```

The script is executed in approximately 10 minutes using approximately 40MB of memory. At this point it is worth making some observations about the results of the reduction. We launch a MAPLE window, load the AIR files for the pentagon topology, and read the result for one of the reduced integrals:

```
> currentdir(''/home/AIR/Pentagon'');
> read ''input_pentagon.map'';
> read ''/home/AIR/main.map'';
> show_int(V(1, 1, 1, 1, -4), ''.'');
```

The result now contains masked expressions, e.g. f[67].Their value is stored in the ICED directory tree. We can see the value of the masked expressions using the routine:

```
show_f(i, RESULTS_DIR);
```

which displays the variable f[i] for the integer i. We must also provide the reduction directory RESULTS_DIR For example, the value of f[12] can be retrieved by typing:

```
> show_f(12, ''.'');
```

Moreover, the command

```
> show_full_int(V(1,1,1,1,-4), ''.'');
```

will return the full expression of the wanted integral. The commands `show_int`, `show_full_int` and `show_f` are very convenient to collect the results and export them to other platforms for further calculations.

We proceed, next, with further reductions of integrals of the pentagon topology with $\mathcal{N}_- \leq 5$, and $\mathcal{N}_+ = 0$. To perform this reduction we have to prepare a run using the same input file and requiring additional seeds generated in the script file,

```
> SEEDGEN("seeds.map",[1,2,3,4,5],[2,5],[0,5],[0,0]);
```

The script runs now in approximately 25 minutes making use of 65MB of memory. The coefficients involved are now larger, but they were computed very efficiently.

We have performed a number of reductions for one loop hexagon and heptagon topologies and various two-loop topologies. For example, all loop and phase-space integral topologies in [4, 6] are fully reduced in less than 6 hours. The reduction of the massless double box topology with $\mathcal{N}_- \leq 4$, and $\mathcal{N}_+ = 0$ can be performed in about three days; the `Reducer` routine was running for approximately 7 hours, the `tidy_list` routine went through in one hour and ten minutes, while the `melt_all_f` routine worked for a couple of days. The cross-box massless box topology is reduced in approximately four days. In massive two-loop topologies, we have reduced integral topologies for the production of heavy quarks. The double-box topology with two massive external legs and a massive propagator (all carrying the same mass) is reduced using the masking algorithms in about 32 hours while the nested substitutions are completed in about 20 days for $\mathcal{N}_- \leq 4$, and $\mathcal{N}_+ = 0$.

8. Conclusions

We have presented a MAPLE program for Automated Integral Reductions in perturbative calculations. Our program is based on an algorithm introduced by Laporta [14], and uses the method of Gauss elimination for solving large systems of equations. The program can reduce generic loop or phase-space integrals or other functions (like hypergeometric functions) which satisfy coupled algebraic identities.

The main obstacle in multiloop reductions is the large size of the symbolic expressions. We have implemented two algorithms in our program which organize the reduction more efficiently and reduce the amount of computations. The routines mask reduced expressions or large integral coefficients. This enables solving the system of IBP equations without performing all substitutions explicitly. The computationally intensive task of nested substitutions is performed only after the procedure of Gauss-elimination is completed, and for only a small fraction of expressions which appear in the final results.

Reduction algorithms cannot be extended to arbitrarily large calculations due to finite computing resources. We believe, however, that many phenomenologically interesting problems are tractable using AIR. There are a few improvements that we expect to make in future releases, implementing a more flexible database for storing the equations, and including more efficient algorithms for performing nested substitutions and simplifying large expressions. The methods described in [28] are appropriate for achieving this goal.

Acknowledgments

We are grateful to Frank Petriello for very valuable suggestions, crucial observations, and for his contributions in developing parts of AIR. We would like to thank Lee Garland, Nigel Glover, Thanos Koukoutsakis, Carlo Oleari, and Maria Elena Tejeda-Yeomans for insightful discussions. We would also like to thank Alex Mitov for his detailed feedback and valuable suggestions and Thomas Becher, Lance Dixon, Kirill Melnikov and Marc Schreiber for their suggestions and encourangament.

References

[1] F.V. Tkachov, *A theorem on analytical calculability of four loop renormalization group functions*, *Phys. Lett.* **B 100** (1981) 65.

[2] K.G. Chetyrkin and F.V. Tkachov, *Integration by parts: the algorithm to calculate beta functions in 4 loops*, *Nucl. Phys.* **B 192** (1981) 159.

[3] T. Gehrmann and E. Remiddi, *Differential equations for two-loop four-point functions*, *Nucl. Phys.* **B 580** (2000) 485 [hep-ph/9912329].

[4] C. Anastasiou and K. Melnikov, *Higgs boson production at hadron colliders in NNLO QCD*, *Nucl. Phys.* **B 646** (2002) 220 [hep-ph/0207004].

[5] C. Anastasiou and K. Melnikov, *Pseudoscalar Higgs boson production at hadron colliders in NNLO QCD*, *Phys. Rev.* **D 67** (2003) 037501 [hep-ph/0208115].

JHEP07(2004)046

[6] C. Anastasiou, L.J. Dixon and K. Melnikov, *NLO Higgs boson rapidity distribution at hadron colliders*, *Nucl. Phys.* **116** *(Proc. Suppl.)* (2003) 193 [hep-ph/0211141].

[7] C. Anastasiou, L.J. Dixon, K. Melnikov and F. Petriello, *Dilepton rapidity distribution in the Drell-Yan process at NNLO in QCD*, *Phys. Rev. Lett.* **91** (2003) 182002 [hep-ph/0306192].

[8] C. Anastasiou, L. Dixon, K. Melnikov and F. Petriello, *High-precision QCD at hadron colliders: electroweak gauge boson rapidity distributions at NNLO*, *Phys. Rev.* **D 69** (2004) 094008 [hep-ph/0312266].

[9] S.G. Gorishnii, S.A. Larin, L.R. Surguladze and F.V. Tkachov, *Mincer: program for multiloop calculations in quantum field theory for the schoonschip system*, *Comput. Phys. Commun.* **55** (1989) 381.

[10] T. van Ritbergen, J.A.M. Vermaseren and S.A. Larin, *The four-loop beta function in quantum chromodynamics*, *Phys. Lett.* **B 400** (1997) 379 [hep-ph/9701390].

[11] M. Steinhauser, *Matad: a program package for the computation of massive tadpoles*, *Comput. Phys. Commun.* **134** (2001) 335 [hep-ph/0009029].

[12] V.A. Smirnov and O.L. Veretin, *Analytical results for dimensionally regularized massless on-shell double boxes with arbitrary indices and numerators*, *Nucl. Phys.* **B 566** (2000) 469 [hep-ph/9907385].

[13] C. Anastasiou, T. Gehrmann, C. Oleari, E. Remiddi and J.B. Tausk, *The tensor reduction and master integrals of the two-loop massless crossed box with light-like legs*, *Nucl. Phys.* **B 580** (2000) 577 [hep-ph/0003261].

[14] S. Laporta, *High-precision calculation of multi-loop Feynman integrals by difference equations*, *Int. J. Mod. Phys.* **A 15** (2000) 5087 [hep-ph/0102033].

[15] L.W. Garland, T. Gehrmann, E.W.N. Glover, A. Koukoutsakis and E. Remiddi, *The two-loop QCD matrix element for $e^+e^- \to 3jets$*, *Nucl. Phys.* **B 627** (2002) 107 [hep-ph/0112081].

[16] C. Anastasiou, E.W.N. Glover and M.E. Tejeda-Yeomans, *Two-loop QED and QCD corrections to massless fermion boson scattering*, *Nucl. Phys.* **B 629** (2002) 255 [hep-ph/0201274].

[17] Y. Schroder, *Automatic reduction of four-loop bubbles*, *Nucl. Phys.* **116** *(Proc. Suppl.)* (2003) 402 [hep-ph/0211288].

[18] T. Becher and K. Melnikov, *The asymptotic expansion of lattice loop integrals around the continuum limit*, *Phys. Rev.* **D 66** (2002) 074508 [hep-ph/0207201].

[19] T. Becher and K. Melnikov, *Continuum methods in lattice perturbation theory*, *Nucl. Phys.* **116** *(Proc. Suppl.)* (2003) 407 [hep-ph/0211215].

[20] R. Bonciani, P. Mastrolia and E. Remiddi, *Vertex diagrams for the QED form factors at the 2-loop level*, *Nucl. Phys.* **B 661** (2003) 289 [hep-ph/0301170].

[21] C. Anastasiou, Z. Bern, L.J. Dixon and D.A. Kosower, *Planar amplitudes in maximally supersymmetric Yang-Mills theory*, *Phys. Rev. Lett.* **91** (2003) 251602 [hep-th/0309040].

[22] C. Anastasiou, L.J. Dixon, Z. Bern and D.A. Kosower, *Cross-order relations in $N = 4$ supersymmetric gauge theories*, hep-th/0402053.

[23] I. Blokland, A. Czarnecki, M. Slusarczyk and F. Tkachov, *Heavy-to-light decays with a two-loop accuracy*, hep-ph/0403221.

[24] MAPLE, `http://www.maplesoft.com`.

[25] R.K. Ellis and J.C. Sexton, *QCD radiative corrections to parton parton scattering*, *Nucl. Phys.* **B 269** (1986) 445.

[26] AIR, `http://www.theorphys.sci.kun.nl/people/achilleas/air/air.php`.

[27] J.A.M. Vermaseren, *New features of form*, `math-ph/0010025`.

[28] S. Weinzierl, *Subtraction terms for one-loop amplitudes with one unresolved parton*, *J. High Energy Phys.* **07** (2003) 052 [`hep-ph/0306248`].