Module Intersection for the Integration-by-Parts Reduction of Multi-Loop Feynman Integrals

Dominik Bendle,[a,b] **Janko Böhm,**[a,*] **Wolfram Decker,**[a] **Alessandro Georgoudis,**[c,d] **Franz-Josef Pfreundt,**[b] **Mirko Rahn**[b] **and Yang Zhang**[e,f,*]

[a]*Department of Mathematics, Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany*
[b]*Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany*
[c]*Laboratoire de physique de l'Ecole normale superieure, ENS, Université PSL, CNRS, Sorbonne Université, Université Paris-Diderot, Sorbonne Paris Cité, 24 rue Lhomond, 75005 Paris, France*
[d]*Institut de Physique Théorique, CEA, CNRS, Université Paris-Saclay, F-91191 Gif-sur-Yvette cedex, France*
[e]*Peng Huanwu Center for Fundamental Theory, Hefei, Anhui 230026, China*
[f]*Interdisciplinary Center for Theoretical Study, University of Science and Technology of China, Hefei, Anhui 230026, China*
E-mail: bendle@rhrk.uni-kl.de, boehm@mathematik.uni-kl.de, decker@mathematik.uni-kl.de, alessandro.georgoudis@phys.ens.fr, franz-josef.pfreundt@itwm.fraunhofer.de, mirko.rahn@itwm.fraunhofer.de, yzhphy@ustc.edu.cn

In this manuscript, which is to appear in the proceedings of the conference "MathemAmplitude 2019" in Padova, Italy, we provide an overview of the module intersection method for the the integration-by-parts (IBP) reduction of multi-loop Feynman integrals. The module intersection method, based on computational algebraic geometry, is a highly efficient way of getting IBP relations without double propagator or with a bound on the highest propagator degree. In this manner, trimmed IBP systems which are much shorter than the traditional ones can be obtained. We apply the modern, Petri net based, workflow management system GPI-Space in combination with the computer algebra system Singular to solve the trimmed IBP system via interpolation and efficient parallelization. We show, in particular, how to use the new plugin feature of GPI-Space to manage a global state of the computation and to efficiently handle mutable data. Moreover, a Mathematica interface to generate IBPs with restricted propagator degree, which is based on module intersection, is presented in this review.

MathemAmplitudes 2019: Intersection Theory & Feynman Integrals
18-20 December 2019
Padova, Italy

*Speaker

1. Introduction

In the past years, several new developments have been achieved for multi-loop scattering amplitudes computation. For example, particular interest has been put into tackling the $2 \to 3$ scattering processes at the next-to-next-to-leading order [1–15]. These theoretical inputs are needed for analyzing data coming from the LHC collider, especially for precise background processes.

In obtaining these results, the study of Feynman integrals and their properties is still a fundamental tool to achieve progress. In this context, generating integration-by-parts (IBP) identities [16] is an important step for computing amplitudes and Feynman integrals themselves, using the method of differential equations [17–27]. As such computations are the current bottleneck for most of the needed processes, there have been different approaches that try to bypass [28–35] or to simplify this step by utilizing numerical methods [9, 10, 13, 36–38]. (See also ref. [27, 39–46] for integral reduction not directly applying IBPs.)

The development of new IBP algorithms, based mostly on ideas and tools coming from algebraic geometry, is the main topic of this review. IBP relations are derived from the integration of a total derivative. By generating enough of these identities it is possible to express each integral appearing in the amplitude as a linear combination of a finite basis of master integrals (MI) using Gaussian elimination and the Laporta algorithm [47]. There exist several available software packages, both private and public, to obtain such relations [48–54]. The method presented here, relies on a different approach, where firstly the a priory knowledge of basis of MI is needed. To obtain such a basis, different methods can be used [55, 56]. Recently it has been discovered how a careful choice of the basis can greatly simplify the reduction process [57–60] by generating simpler IBP coefficients. A second step is to construct, using the Baikov representation and the Laplace expansion [61], a canonical set of IBPs. It is then possible to further restrict the number of these identities by requiring that no integral with double propagators appears [62–64]. This is achieved through a module intersection computation [65]. An extra step is then to construct a spanning set of cuts [64], further reducing the problem as it is possible to divide the integrals contributing to the IBP identities into different families and treat them separately[1]. Finally, besides reducing the number of identities generated, it is important to have an efficient Gaussian elimination to solve the associated linear system. The most efficient method for performing this task is to utilize rational reconstruction and interpolation [54, 67–69], where by sampling different numerical points one can obtain the analytical form of the IBP coefficients. In our implementation of this idea, we rely on the Singular-GPI-Space framework [70] for massively parallel computation in computer algebra, which combines the computer algebra system Singular [71] with the workflow management system GPI-Space [72], and thus allows us to massively parallelize the reduction and interpolation part of the algorithm. This framework relies on Petri nets to coordinate the computation in GPI-Space, while Singular is used as the computational back end. The Petri net formalism used in GPI-Space is an extension of the basic idea, which allows tokens in the net to be complex data structures. Modifying our original implementation in the Singular-GPI-Space framework, we illustrate the use of the new plugin feature of GPI-Space to extend the Petri net formalism to manage a global state of the computation and integrate mutable data into the formalism. The IBP reduction method described here has been

[1]It is also possible to singularly nullify the integrals appearing in the reduction process [66].

successfully used to obtain reduction for the non-planar 5-point topologies appearing at 2-loop order [57, 65].

In this review paper, we also present a Mathematica interface for generating IBPs without double propagators (or IBPs with restricted propagator degree), via our module intersection method, see

https://bitbucket.org/yzhphy/module-intersection/src

This review is organized as follows: In Section 2, we present the intersection algorithm used to generate the IBP system. In Section 3, we describe our Gaussian reduction approach, introducing the GPI-Space system and the Petri net formalism to formulate our workflow. We then present some examples and finish with some outlook on possible future directions.

2. Module Intersection

In this section, we explain the module intersection method of generating a trimmed IBP system. We consider the Baikov representation of Feynman integrals.

$$I[\alpha_1,\ldots,\alpha_m] = C_E^L U^{\frac{E-D+1}{2}} \int_\Omega dz_1 \cdots dz_m P(z)^{\frac{D-L-E-1}{2}} \frac{1}{z_1^{\alpha_1}\cdots z_m^{\alpha_m}}, \tag{1}$$

where $P(z)$ is the Baikov polynomial

$$P = \det G \begin{pmatrix} l_1, & \ldots & l_L, & p_1, & \ldots & p_E \\ l_1, & \ldots & l_L, & p_1, & \ldots & p_E \end{pmatrix}. \tag{2}$$

Here U and C_E^L are the Gram determinant and constant factor, which are irrelevant for the IBP reduction. The polynomial P vanishes on the boundary of integration $\partial\Omega$.

In the Baikov representation, an IBP relation reads

$$0 = \int dz_1 \cdots dz_m \sum_{i=1}^m \frac{\partial}{\partial z_i}\left(a_i(z) P^{\frac{D-L-E-1}{2}} \frac{1}{z_1^{\alpha_1}\cdots z_m^{\alpha_m}} \right), \tag{3}$$

where the $a_i(z)$ are polynomials in the Baikov variables $z = (z_1,\ldots,z_m)$. Expanding the total derivative above, we get an IBP relation of Feynman integrals. Note that the resulting relation may not match our demands: the derivative of P provides dimensionally shifted integrals, and for the traditional IBPs there is no control of the propagator power increase. However, it is easy to meet these demands by adding constraints on the $a_i(z)$. Let $\mathcal{S}$ be a subset of $\{1,\ldots,m\}$, for which the propagator indices are supposed to be constrained:

$$\left(\sum_{i=1}^m a_i(z) \frac{\partial P}{\partial z_i} \right) + b(z) P = 0, \tag{4}$$

$$a_i(z) = b_i(z) z_i, \quad i \in \mathcal{S}, \tag{5}$$

where $b(z)$ and the $b_i(z)$ are all polynomials in the Baikov variables. The second equation is to make sure that the resulting IBP relation does not increase the propagator power α_i for $i \in \mathcal{S}$. These are the so-called syzygy equations.

The first syzygy equation (4) can be easily solved by the Laplacian expansion of a symmetric matrix [36], or the canonical IBPs converted to Baikov form [36]. The solution set is a polynomial module called M_1 whose generators are at most linear in the Baikov variables. This step takes almost no computing time.

The module solving the syzygy equations (5) clearly has the trivial generating set $a_i = z_i$, $b_i = 1, i \in \mathcal{S}$. We call this solution set M_2. So without much computing efforts, we get M_1 and M_2, the solution sets for the syzygy equations (4) and (5), individually.

The main computational goal is then to get,

$$M_1 \cap M_2 \,. \tag{6}$$

This module intersection is obtained via a module Gröbner basis in a position-over-term ordering [65].

Although the generators of M_1 and M_2 contain at most linear polynomials in the Baikov variables, the intersection computation needs the following technique to finish efficiently:

- *Localization.* Let the kinematic parameters (Mandelstam and mass parameters) be $(c_1, \ldots, c_t)$. Instead of the obvious computation in the ring $R = \mathbb{Q}(c_1, \ldots, c_t)[z_1, \ldots, z_m]$, we do the computation in the ring $R' = \mathbb{Q}[z_1, \ldots, z_m, c_1, \ldots c_t,]$ with the block ordering

$$z_1, \ldots, z_m \succ c_1, \ldots c_t \,. \tag{7}$$

  This amounts to treat parameters in the same way as the Baikov variables. This technique is essential for the multivariate computations to finish. After the Gröbner basis computation in R', we map the result back to R and remove redundant generators.

- The use of a *degree bound* for the intersection computation. Usually, we do not need the full generating set of the intersection $M_1 \cap M_2$. So heuristically, we can apply a degree bound on the intersection computation to reduce the computation time. In practice, this is done via the "degBound" option in SINGULAR. A posteriori, we verify that the degree bound was chosen large enough.

To this proceedings article, we attach a MATHEMATICA interface to SINGULAR for the module intersection computation and for generating constricted IBP relations. A pure SINGULAR version of this will be available in the future as a SINGULAR library.

3. Efficient Gaussian Elimination with GPI-Space

In this section we discuss our parallel implementations in the workflow management system GPI-SPACE [72] in combination with the computer algebra system SINGULAR [71] using the framework developed in [70]. In the computation of Feynman integrals, GPI-SPACE is used to parallelize the Gaussian reduction of the linear system which is derived from the IBP identities. This is achieved by splitting the large reduction job into numerous small ones by passing to "semi-numeric" computations, which are then recombined using interpolation.

In the following, we will give a short overview of GPI-SPACE and Petri nets, which are used to formulate parallel workflows. We will then describe the Petri nets used in the computation of Feynman integrals. For more details, also refer to [73].

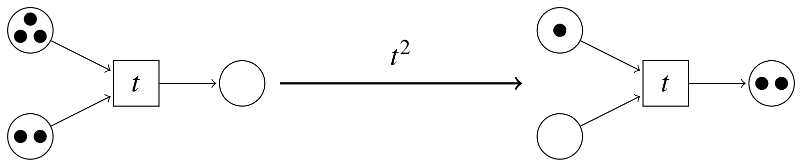


Figure 1: Transition firing until it is disabled.

3.1 GPI-Space and Petri Nets

GPI-Space is a workflow management system developed by the Fraunhofer Institute for Industrial Mathematics (ITWM) and is comprised of three components:

- The *distributed run-time system* (DRTS) initializes and manages workers in accordance with the available computing resources, and allocates computing jobs as they become available.
- The *workflow engine* (WE) tracks the state of the workflow. Jobs available for execution are identified and, together with their input data, sent to the DRTS for scheduling.
- A *virtual memory layer* allows communication and data sharing between different machines managed by GPI-Space.

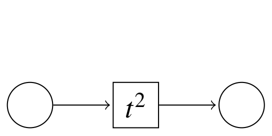
Petri nets serve as the top level description of workflows for the GPI-Space engine. They can be defined as directed bipartite paths where the nodes are divided into *places* and *transitions*, which model the data that is passed between procedures, and data processing algorithms, respectively. Depending on the orientation of the directed vertex between places and transitions, places may be *input* or *output places* of a given transition. Places may hold *tokens*, which represent the data, and can be consumed or produced by transitions. The association of tokens to places at a given time is called a *marking* of the Petri net.

If all input places of a transition hold at least one token, the transition is called *enabled* and may *fire*, that is, one token is consumed from each input place, and each output place receives a new token. If we have a marking M of a Petri net and obtain a new marking M' by firing the transition t, we write $M \xrightarrow{t} M'$. If t can fire multiple times, producing a sequence of markings

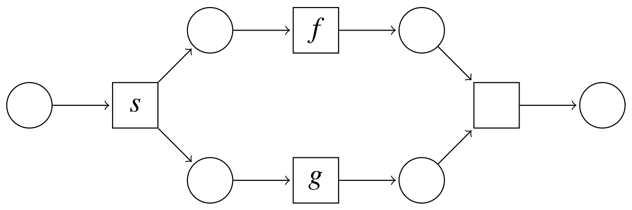
$$M \xrightarrow{t} M^{(1)} \xrightarrow{t} \cdots \xrightarrow{t} M^{(n)}$$

by a sequential execution of the firings, the transition t, in fact, can fire in parallel for each tuple of available input tokens. This form of parallelisim offered by Petri nets is referred to by the notation $M \xrightarrow{t^n} M^{(n)}$ and is called *data parallelism*. Examples of data parallelism are shown in Figures 1 and 2a. In figures, we depict transitions as boxes and places as circles. The flow relations between places and transitions is given by arrows. Of course, this parallelism extends to multiple transitions as well. If several transitions are enabled, they can fire at the same time. This is called *task parallelism* and is illustrated in Figure 2b.

In the basic formulation of a Petri net, tokens are structureless and not distinguishable. In order to make programming in terms of Petri nets a practically feasible task, Petri nets in GPI-Space are augmented with additional features:



(a) Data parallelism



(b) Task parallelism

Figure 2: Minimal Petri nets illustrating parallelisms

1. The most important addition is for tokens to hold actual data (which corresponds to the theoretical concept of so-called *colored* Petri nets). GPI-Space offers primitive data types and allows the use of user-defined types via an embedded programming language. In this way it possible to impose additional restrictions on the tokens which a transition can accept. This can be used to implement `if-then-else` and loop functionality on the Petri net level. In particular, it is then possible to connect a place to more than one transition as an input without ambiguity (a token might otherwise be consumed by any connected transition at random, see for example Figure 3). In fact, even though GPI-Space allows that *conflicts* may arise and then randomly selects a transition to fire, in almost all cases Petri nets are easier to maintain if there never exist more than one transition which can consume a given token.

2. In colored Petri nets, tokens can be equipped with a type and transitions may impose type restrictions on their input. In GPI-Space, places and transitions are strictly typed, which means that they must be defined with a fixed type signature. In particular, places and transitions can only be connected if these signatures match.

3. For small computations, an embedded programming language can be used to carry out so-called *tiny computations*, which are not scheduled for execution in a (potentially remote) worker process, but are instead executed directly in the workflow engine. For tasks involving only small arithmetic computations or manipulation of container data, this avoids unnecessary overhead.

The execution of a Petri net is non-deterministic in both the choice of which transition to fire and the choice which token to consume. The actual choice depends on a random number generator and variations in the execution times of the transitions. In other words, in order to facilitate parallelism, GPI-Space forces the applications to use a functional approach to programming which does not depend on an external state. Especially in the context of computer algebra, the introduction of non-determinism has proved to lead to an increase in consistency and predictability of our algorithms.

3.2 The Petri Net for the Matrix Reduction of a Single Cut

The matrix reduction performed on a given set of IBP identities is the most involved part in terms of computation time and memory, which makes it the most suitable component of our algorithm for a massively parallel implementation. As previously described, this is achieved by substituting a subset of parameters with a number of numeric values, carrying out the necessary computations in

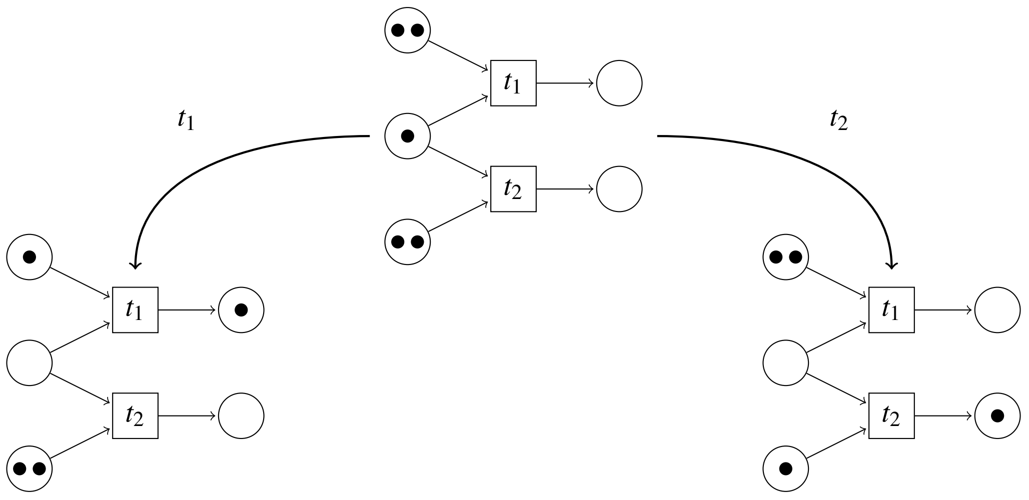


Figure 3: Conflict of two transitions leading to different program flows.

the semi-numeric setting, and then recombining the results into the final, row-reduced linear system via polynomial interpolation.

GPI-Space allows the user to attach so-called *plugins* to a workflow, which can insert tokens (holding data) into the net, independently of the execution of the Petri net by firing transitions. Compared to traditional Petri net formulations, one main advantage of this plugin system is the ability to maintain a *global state* of the computation: Emulating such a state as a single token in a Petri net would require a complete rewrite for each update to the state, since a transition would first have to consume this token and then place back the modified token. For larger tokens, this introduces an unreasonably large overhead and may impact the performance significantly. In the initial version of our IBP reduction Petri net, this was circumvented by simply starting a remote process which kept track of the state. Where required, a transition could then communicate with this process to send and receive tokens.

Using the plugin system, this approach can now be realized without running processes outside of the execution of GPI-Space. The plugin-managed state is created and controlled by the Petri net and its GPI-Space provided identifier is used by the Petri net to steer all modifications and retrieve information stored in the plugin-managed state. The Petri net uses (specifically annotated) transitions to transport tokens into the plugin. The plugin-transition can receive tokens from the plugin and insert them into the Petri net just like every "normal" transition. In addition, the plugin can also work independently from and in parallel to the Petri net and use (specifically annotated) places to *inject* tokens into the running Petri net. In summary: The Petri net plugins are a mechanism to enable Petri nets to explicitly manage *mutable* state without involving the scheduler and the runtime system, while retaining the conceptual advantages of the Petri net formalism.

The Petri net shown in Figure 4 illustrates the current version of the Petri for computing the matrix reduction of the IBP identities obtained from a single cut. Dashed arrows refer to *read-only* connections, which means that tokens are in fact not consumed, allowing parallel access to data where modification is not necessary. Dotted arrows show the special data and token movement between the net and the plugin. Finally, transitions may be annotated with conditions restricting their execution, which enables us to implement conditional execution on the Petri net level. In the following, we describe the execution structure of the net:

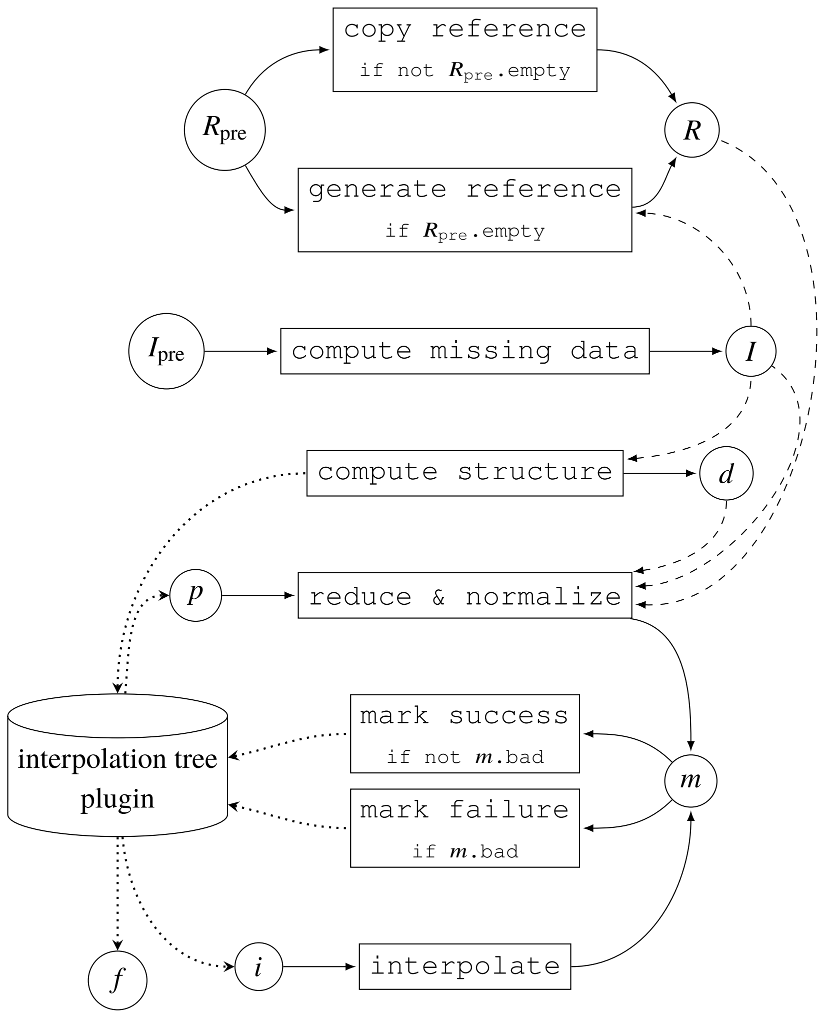


Figure 4: Updated and simplified Petri net incorporating plugin system.

Input tokens: The net is initialized with two tokens. One is the input data token which is put onto I_{pre}. It is a structured type with the following fields:

- The (filename pointing to the) input linear relations stored as a matrix over a rational function field $\mathbb{Q}(\vec{c}, D)$.
- The list of indices of the parameters which will be substituted during the reduction computation and interpolated, say $\{1, \ldots, r\}$.
- The list of indices of parameters which will be substituted during the reference reduction. This is not necessarily the complement of the set of interpolated parameters, for instance, if some parameters in the function field are of no interest during the reduction computations.
- The list of target integral indices.

- Optionally: A precomputed trace for row and column swaps during the reduction step. This field will be referred to as `trace` in the Petri net.
- Optionally: A list of master integral indices. These indices can be obtained by a fixed-column computation and thus can be omitted. In the net, this field is referred to as `master`.

The second token is a reference token which is required by our rational function interpolation algorithm and contains an result for the complementary set of analytic parameters, that is, exchanging in the hybrid approach the interpolated and the analytic parameters. This token is put onto R_{pre}. This token optionally contains a reference substitution point and the relations from I_{pre} already reduced according to this point. The data field `empty` of the token can be used to indicate that this token does *not* contain this information.

This option is useful in the following sense: If the number of substituted parameters is considerably larger than the set of reference parameters, then a direct computation of the reference might not be feasible. In this case, the reference is computed by iteratively using the parallel algorithm.

Transition `copy reference`: If for a token on the place R_{pre} the field $R_{\texttt{pre}}$`.empty` has the value false, this token holds a precomputed reference matrix. In this case, the token is simply moved to R by the transition `copy reference`.

Transition `generate reference`: Otherwise, that is, if $R_{\texttt{pre}}$`.empty` is true, no reference matrix was supplied. This transition then generates a random reference point and computes the reference matrix, which is then put onto the place R.

Transition `compute missing data`: This transition computes the optional fields `trace` and `master`, if they are missing from the input token on I_{pre}. The completed input token is then put onto the place I. In practice, this transition is a slightly more involved subnet. Since it is functionally and structurally not particularly interesting, it is omitted from our presentation.

Transition `compute structure`: Again, this is a simplification of a more involved subnet to simplify the presentation. Here, the polynomial degrees of all coefficients of the fully reduced linear system are computed for each parameter via univariate computations. The maximum degrees per parameter are supplied to the interpolation tree plugin to determine the required number of interpolation points. A list of matrices which hold the degrees of the numerator and denominator of each matrix entry is then put onto the place d. This data on the polynomial degrees is required in the normalization step.

Interpolation Tree Plugin: The tree manages the interpolation progress in a tree structure, as outlined above. As soon as the degree data is determined, the plugin can generate an initial required set of interpolation points, which are then put onto the place p. If reductions fail or result in bad data, points are marked as failures in the tree. Accordingly, the plugin generates new points to ensure that sufficient data for the interpolation is available. If, however, enough reductions (or interpolations) for a given parameter are successful, the plugin generates the corresponding point for this interpolation, which is then put onto the place i. When the root of the interpolation tree, which corresponds to the fully interpolated result, is marked as completed, the interpolation tree plugin outputs the filename of the final result as a token onto the place f. This terminates the execution of the Petri net.

Transition `reduce & normalize`: For a point token from p, this transition performs the matrix reduction according to the traces supplied in the input token on I. As we perform division during the computation with fixed column and row swap traces, we may encounter division by zero after substituting certain integer values. In this case, the output token will be marked as *bad*. With the degree data from d and the reference reduction result from R, the transition in addition normalizes the successful reduction result and potentially marks it as *bad*, if polynomial cancellation occurred in some entry. Output tokens store the associated substitution point and (the filename of) the reduction and normalization result if it exists. Accordingly, the token has the additional field `bad` and is put onto the place m.

Transitions `mark success` and `mark failure`: As stated above, the tokens on m indicate with the field `bad` whether the reduction and normalization computations produced a valid result. These two transitions then mark the associated substitution point accordingly as *completed* or *failed* in the interpolation tree.

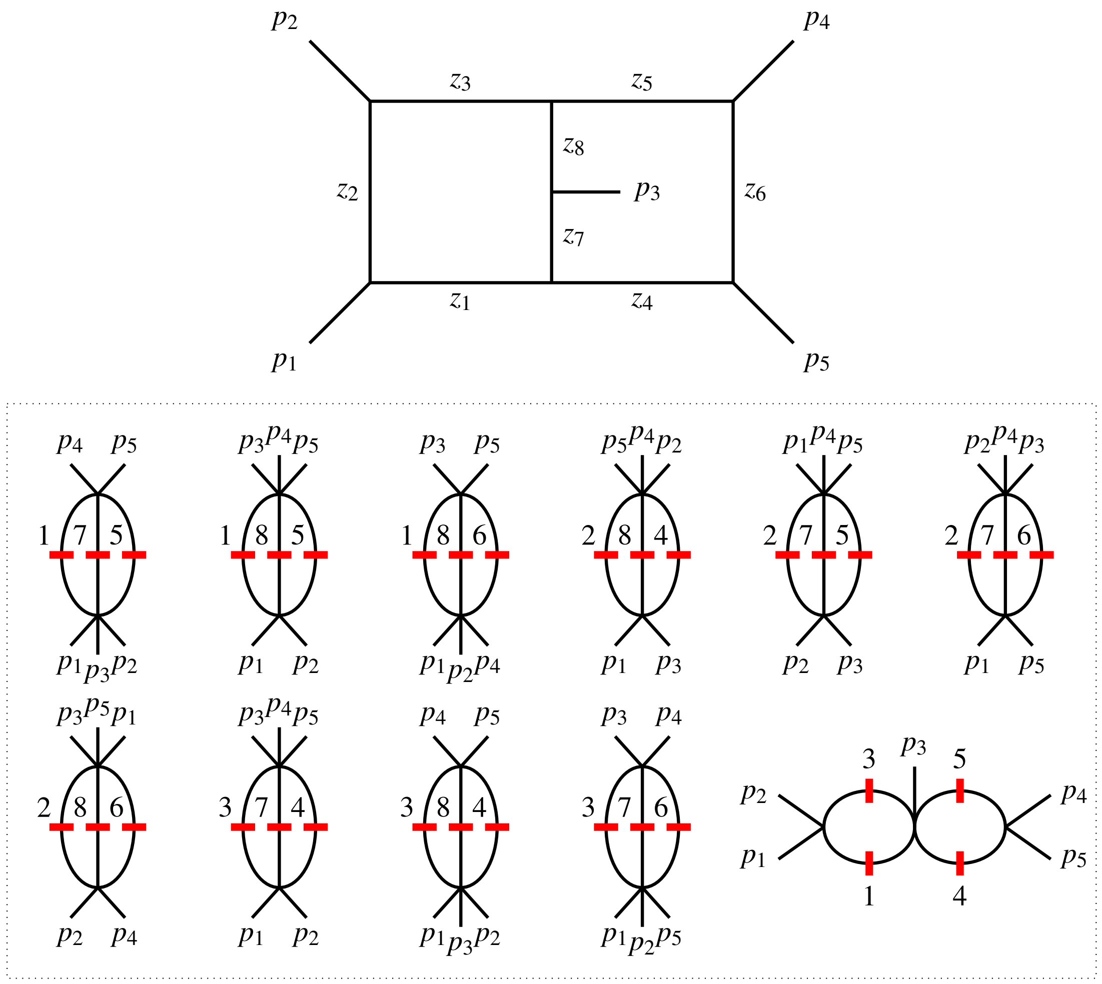
Transition `interpolate`: The tokens in i are generated by the interpolation plugin to contain the filenames for all matrices required for the particular interpolation. This transition then simply computes the polynomial interpolations for both numerators and denominators of the result matrix. The resulting matrix and the substitution point which correspond to this interpolation are then put onto the place m. These tokens will always have a `bad`-value of false.

As stated in the description above, the Petri net execution terminates as soon as the output matrix is fully interpolated and the corresponding token is put onto the place f.

A desirable feature of long-running large-scale computations is restartability from a previously reached state, due to the possibility of compute node failures or job cancellations upon reaching imposed time limits. For our algorithm, this can be achieved by storing – or recreating – the interpolation tree managed by the plugin. Since the tree creation deterministically depends on the list of parameter indices and their degrees, recreating subsequent states can be achieved by logging the *modifications* of this tree, that is, by recording whether a node was marked as failed or successful, and re-applying them. Since the information logged in this way does not comprise the full structure for most of the Petri net execution, this is more efficient than storing the whole tree. Of course, this requires the user to restart the algorithm without changing its parameters, which introduces potentially undesirable inflexibility.

4. Example

In this section, we present a nontrivial example of IBP reductions, namely the two-loop five-point non-planar integrals, as shown in Figure 5. The symbols of its UT integral basis were calculated in [5, 11], and the analytic expressions were obtained in [74].



p_2
p_4
z_3
z_5
z_8
z_2
p_3
z_6
z_7
z_1
z_4
p_1
p_5
p_4 p_5 1 7 5 $p_1 p_3 p_2$
$p_3 p_4 p_5$ 1 8 5 p_1 p_2
p_3 p_5 1 8 6 $p_1 p_2 p_4$
$p_5 p_4 p_2$ 2 8 4 p_1 p_3
$p_1 p_4 p_5$ 2 7 5 p_2 p_3
$p_2 p_4 p_3$ 2 7 6 p_1 p_5
$p_3 p_5 p_1$ 2 8 6 p_2 p_4
$p_3 p_4 p_5$ 3 7 4 p_1 p_2
p_4 p_5 3 8 4 $p_1 p_3 p_2$
p_3 p_4 3 7 6 $p_1 p_2 p_5$
p_2 p_1 3 p_3 5 1 4 p_4 p_5

Figure 5: Two-loop five-point nonplanar double pentagon diagram with inversed propagators z_i.

The propagators of this integral family are

$$\begin{aligned}
&D_1 = l_1^2 && D_2 = (l_1 - k_1)^2 && D_3 = (l_1 - k_{12})^2 && D_4 = l_2^2 \\
&D_5 = (l_2 - k_{123})^2 && D_6 = (l_2 - k_{1234})^2 && D_7 = (l_1 - l_2)^2 && D_8 = (l_1 - l_2 + k_3)^2 \\
&D_9 = (l_1 - k_{1234})^2 && D_{10} = (l_2 - k_1)^2 && D_{11} = (l_2 - k_{12})^2 ,
\end{aligned} \tag{8}$$

where the l_i represent the loop momenta, the k_i represent external momenta, and $k_{i\cdots j} = \sum_i^j k_i$.

The difficulty of the IBP reduction is due to the 6 parameters, which are s_{12}, s_{23}, s_{34}, s_{45}, s_{15} and the spacetime dimension D, and the nonplanar feature. We illustrate our algorithm by IBP reducing the 26 integrals up to the numerator degree of 4, to a Laporta integral basis. (We note that the degree-5 analytic IBP reduction of the same diagram to a Laporta basis was calculated in a recent paper [54] by the reduction of a new type of integral relations in the block triangular form [44].)

Furthermore, we convert the IBP coefficients to the coefficients of a dlog basis and use an improved Leinartars' algorithm to further simplify the coefficients.

First, we use Azurite [56] or mint to find an integral basis. Ignoring symmetries, there are 113 irreducible integrals, while there are 108 master integrals with symmetries. We apply spanning cuts of this integral family, when using the package ModuleIntersection to generate IBPs without double propagators. The 11 spanning cuts are,

$$\begin{gathered}\{1,5,7\},\{1,5,8\},\{1,6,8\},\{2,4,8\},\{2,5,7\},\{2,6,7\},\\ \{2,6,8\},\{3,4,7\},\{3,4,8\},\{3,6,7\},\{1,3,4,5\}\end{gathered} \tag{9}$$

where the numbers refer to the propagator indices.

Relying on the `degbound` option in Singular, the intersections are generated using the command ModuleIntersection. For a polynomial degree bound of 5, utilizing one core, it takes in total less than 5 minutes to *analytically* generate all the module intersections. A posteriori, we see that a degree bound of 5 is sufficient for this IBP reduction problem.

After getting the IBPs without double propagators, we apply a two-step trimming process to select only IBPs necessary for the target integrals. This computation is done over finite fields and powered by the linear algebra package SpaSM.

Different spanning cuts may support the same master integral, that is, there can be a cut overlap, e.g., the two cuts $\{1,5,7\}$ and $\{2,4,8\}$ both support $I[1,1,1,1,1,0,1,1,0,0,0]$. Then, we may apply the non-pivot column mask method in ref. [66] to impose

$$I[1,1,1,1,1,0,1,1,0,0,0]=0 \tag{10}$$

in either cut $\{1,5,7\}$ or $\{2,4,8\}$.

As common when computing IBP reductions, we can remove one scale from the problem by setting

$$s_{12}\mapsto 1,\quad c_2\equiv s_{23}/s_{12},\quad c_3\equiv s_{34}/s_{12},\quad c_4\equiv s_{45}/s_{12},\quad c_5\equiv s_{15}/s_{12}. \tag{11}$$

The resulting IBP relations are very short and sparse, as demonstrated in Table 1.

Cut	# relations	# integrals	size	d_2	d_3
{1,5,7}	1134	1182	0.77 MB	21	22
{1,5,8}	1141	1192	0.85 MB	18	18
{1,6,8}	1203	1205	1.1 MB	19	30
{2,4,8}	1245	1247	1.1 MB	35	24
{2,5,7}	1164	1211	0.84 MB	26	18
{2,6,7}	1147	1206	0.62 MB	16	17
{2,6,8}	1126	1177	0.83 MB	16	18
{3,4,7}	1172	1221	0.78 MB	19	18
{3,4,8}	1180	1226	1.0 MB	19	22
{3,6,7}	1115	1165	0.82 MB	21	28
{1,3,4,5}	721	762	0.43MB	14	14

Table 1: IBP relations generated by the module intersection method. Denoted by d_2 and d_3 we also provide the degree of c_2 and c_3 *after the IBP reduction.*

We use our linear reduction algorithm via Singular and GPI-Space for this problem. The computation is done in a semi-numeric fashion by considering c_4, c_5 and the space-time dimension D as symbolic variables, and taking c_2 and c_3 numerically. It is easy to determine the maximal degree of c_2 resp. c_3 in the final reduced IBPs by a univariate computation. The degrees are listed in Table 1.

Different cuts need different amounts of sample points. For instance, for the cut $\{1, 5, 7\}$, 506 points are required. Different cut IBP reductions also need different running times: For example, the simplest cut $\{1, 3, 4, 5\}$ takes only about 11 minutes when using 384 CPU cores. As a comparison, the cut $\{3, 4, 8\}$ is much more difficult: its running time was 12 hours and 21 minutes, with 384 cores.

We merge the IBP reduction on all the cuts to get the complete IBP reduction to a 113 integral basis. Furthermore, we apply the symmetries from Azurite [56] to reduce the 113 basis further to a 108-integral Laporta basis I. The final result is large, with a size of $\sim$ 2.0 GB with $s_{12} \mapsto 1$, and a size of $\sim$ 2.4GB when the s_{12} dependence is recovered. The result is numerically consistent with that obtained by FIRE6 [50].

As discussed in ref. [57], the resulting IBP coefficients can be simplified by converting to a dlog basis. For this example, we use the dlog basis in [74]. Again we apply the semi-numeric approach and then interpolate the variables c_2 and c_3. The computation is powered by Singular and GPI-Space. A posteriori, we see that the maximal degrees for c_2 and c_3 have decreased to

$$d_2' = 20, \quad d_3' = 20. \tag{12}$$

Note that with the Laporta basis, $d_2 = 35$. After the computation, we see that the IBP reduction coefficients in this dlog basis have a size of 480 MB on disk with $s_{12} \mapsto 1$ or 720 MB with symbolc s_{12}. It is a significant 70% reduction of the IBP coefficient size.

If only the IBP reduction coefficients in the dlog basis are needed, we can skip the computation for the Laporta basis IBP coefficients, and directly convert the semi-numeric intermediate results to the dlog basis IBP coefficients. This would reduce the number of sampling points.

The size of the IBP coefficients in the dlog basis is, however, still large. In ref. [60], we discovered that with an improved version of Leinartas' algorithm implemented in Singular, the coefficient size can be dramatically reduced to only 19 MB, with symbolic s_{12}. Comparing with the original IBP coefficients of size 2.4 GB, this is a 100-fold simplification of the IBP reduction coefficients.

5. Summary and Outlook

We have illustrated some recent progress that has been made on tackling the IBP reduction problem. In our approach, the problem is simplified in different algorithmic sub-steps: Firstly by using methods from algebraic geometry to simplify the generated IBP system. A Mathematica interface for generating IBPs with restricted propagator degrees is presented. Secondly by using the Singular-GPI-Space framework to perform Gaussian elimination relying on interpolation of parameters in a massively parallel way. Finally, we have seen how a good choice of basis, in our case the dlog basis, and an efficient algebraic representation of the data can significantly simplify the output of the reduction.

With these new developments, we could hope to tackle higher points or higher loop reductions and amplitudes with increasing number of off-shell external points or internal masses, thus increasing the precision of the theoretical predictions for processes of phenomenological interest. Moreover, we could apply the Singular-GPI-Space framework to other problems than IBP reduction, for example, in recent years some progress has been made into studying the Bethe ansatz equations and partition functions of integrable models [75] or for the study of Grassmanians and their tropical varieties [76].

References

[1] S. Badger, H. Frellesvig, and Y. Zhang, *A Two-Loop Five-Gluon Helicity Amplitude in QCD*, JHEP **12** (2013) 045, [arXiv:1310.1051].

[2] T. Gehrmann, J. M. Henn, and N. A. Lo Presti, *Analytic form of the two-loop planar five-gluon all-plus-helicity amplitude in QCD*, Phys. Rev. Lett. **116** (2016), no. 6 062001, [arXiv:1511.05409]. [Erratum: Phys. Rev. Lett.116,no.18,189903(2016)].

[3] S. Badger, C. Brønnum-Hansen, H. B. Hartanto, and T. Peraro, *First look at two-loop five-gluon scattering in QCD*, Phys. Rev. Lett. **120** (2018), no. 9 092001, [arXiv:1712.02229].

[4] S. Abreu, F. Febres Cordero, H. Ita, B. Page, and M. Zeng, *Planar Two-Loop Five-Gluon Amplitudes from Numerical Unitarity*, Phys. Rev. **D97** (2018), no. 11 116014, [arXiv:1712.03946].

[5] S. Abreu, L. J. Dixon, E. Herrmann, B. Page, and M. Zeng, *The two-loop five-point amplitude in $\mathcal{N} = 4$ super-Yang-Mills theory*, Phys. Rev. Lett. **122** (2019), no. 12 121603, [arXiv:1812.08941].

[6] S. Abreu, F. Febres Cordero, H. Ita, B. Page, and V. Sotnikov, *Planar Two-Loop Five-Parton Amplitudes from Numerical Unitarity*, JHEP **11** (2018) 116, [arXiv:1809.09067].

[7] R. H. Boels, Q. Jin, and H. Luo, *Efficient integrand reduction for particles with spin*, arXiv:1802.06761.

[8] T. Gehrmann, J. M. Henn, and N. A. Lo Presti, *Pentagon functions for massless planar scattering amplitudes*, JHEP **10** (2018) 103, [arXiv:1807.09812].

[9] S. Badger, C. Brønnum-Hansen, H. B. Hartanto, and T. Peraro, *Analytic helicity amplitudes for two-loop five-gluon scattering: the single-minus case*, JHEP **01** (2019) 186, [arXiv:1811.11699].

[10] S. Abreu, J. Dormans, F. Febres Cordero, H. Ita, and B. Page, *Analytic Form of Planar Two-Loop Five-Gluon Scattering Amplitudes in QCD*, Phys. Rev. Lett. **122** (2019), no. 8 082002, [arXiv:1812.04586].

[11] D. Chicherin, T. Gehrmann, J. M. Henn, P. Wasser, Y. Zhang, and S. Zoia, *Analytic result for a two-loop five-particle amplitude*, Phys. Rev. Lett. **122** (2019), no. 12 121602, [arXiv:1812.11057].

[12] D. Chicherin, T. Gehrmann, J. M. Henn, P. Wasser, Y. Zhang, and S. Zoia, *The two-loop five-particle amplitude in $\mathcal{N} = 8$ supergravity*, JHEP **03** (2019) 115, [arXiv:1901.05932].

[13] S. Abreu, L. J. Dixon, E. Herrmann, B. Page, and M. Zeng, *The two-loop five-point amplitude in $\mathcal{N} = 8$ supergravity*, JHEP **03** (2019) 123, [arXiv:1901.08563].

[14] S. Abreu, J. Dormans, F. Febres Cordero, H. Ita, B. Page, and V. Sotnikov, *Analytic Form of the Planar Two-Loop Five-Parton Scattering Amplitudes in QCD*, JHEP **05** (2019) 084, [arXiv:1904.00945].

[15] H. B. Hartanto, S. Badger, C. Brønnum-Hansen, and T. Peraro, *A numerical evaluation of planar two-loop helicity amplitudes for a W-boson plus four partons*, arXiv:1906.11862.

[16] K. Chetyrkin and F. Tkachov, *Integration by parts: The algorithm to calculate β-functions in 4 loops*, Nuclear Physics B **192** (1981), no. 1 159 – 204.

[17] A. V. Kotikov, *Differential equations method: New technique for massive Feynman diagrams calculation*, Phys. Lett. **B254** (1991) 158–164.

[18] A. V. Kotikov, *Differential equation method: The Calculation of N point Feynman diagrams*, Phys. Lett. **B267** (1991) 123–127.

[19] Z. Bern, L. J. Dixon, and D. A. Kosower, *Dimensionally regulated pentagon integrals*, Nucl. Phys. **B412** (1994) 751–816, [hep-ph/9306240].

[20] E. Remiddi, *Differential equations for Feynman graph amplitudes*, Nuovo Cim. **A110** (1997) 1435–1452, [hep-th/9711188].

[21] T. Gehrmann and E. Remiddi, *Differential equations for two loop four point functions*, Nucl. Phys. **B580** (2000) 485–518, [hep-ph/9912329].

[22] J. M. Henn, *Multiloop integrals in dimensional regularization made simple*, Phys. Rev. Lett. **110** (2013) 251601, [arXiv:1304.1806].

[23] C. G. Papadopoulos, *Simplified differential equations approach for Master Integrals*, JHEP **07** (2014) 088, [arXiv:1401.6057].

[24] R. N. Lee, *Reducing differential equations for multiloop master integrals*, JHEP **04** (2015) 108, [arXiv:1411.0911].

[25] J. Ablinger, A. Behring, J. Blümlein, A. De Freitas, A. von Manteuffel, and C. Schneider, *Calculating Three Loop Ladder and V-Topologies for Massive Operator Matrix Elements by Computer Algebra*, Comput. Phys. Commun. **202** (2016) 33–112, [arXiv:1509.08324].

[26] C. G. Papadopoulos, D. Tommasini, and C. Wever, *The Pentabox Master Integrals with the Simplified Differential Equations approach*, JHEP **04** (2016) 078, [arXiv:1511.09404].

[27] X. Liu, Y.-Q. Ma, and C.-Y. Wang, *A Systematic and Efficient Method to Compute Multi-loop Master Integrals*, Phys. Lett. **B779** (2018) 353–357, [arXiv:1711.09572].

[28] L. J. Dixon, J. M. Drummond, and J. M. Henn, *Bootstrapping the three-loop hexagon*, JHEP **11** (2011) 023, [arXiv:1108.4461].

[29] L. J. Dixon, J. M. Drummond, M. von Hippel, and J. Pennington, *Hexagon functions and the three-loop remainder function*, JHEP **12** (2013) 049, [arXiv:1308.2276].

[30] L. J. Dixon and M. von Hippel, *Bootstrapping an NMHV amplitude through three loops*, JHEP **10** (2014) 065, [arXiv:1408.1505].

[31] S. Caron-Huot, L. J. Dixon, A. McLeod, and M. von Hippel, *Bootstrapping a Five-Loop Amplitude Using Steinmann Relations*, Phys. Rev. Lett. **117** (2016), no. 24 241601, [arXiv:1609.00669].

[32] L. J. Dixon, M. von Hippel, and A. J. McLeod, *The four-loop six-gluon NMHV ratio function*, JHEP **01** (2016) 053, [arXiv:1509.08127].

[33] L. J. Dixon, J. Drummond, T. Harrington, A. J. McLeod, G. Papathanasiou, and M. Spradlin, *Heptagons from the Steinmann Cluster Bootstrap*, JHEP **02** (2017) 137, [arXiv:1612.08976].

[34] D. Chicherin, J. Henn, and V. Mitev, *Bootstrapping pentagon functions*, JHEP **05** (2018) 164, [arXiv:1712.09610].

[35] S. Caron-Huot, L. J. Dixon, F. Dulat, M. von Hippel, A. J. McLeod, and G. Papathanasiou, *Six-Gluon Amplitudes in Planar $\mathcal{N} = 4$ Super-Yang-Mills Theory at Six and Seven Loops*, arXiv:1903.10890.

[36] H. Ita, *Two-loop Integrand Decomposition into Master Integrals and Surface Terms*, Phys. Rev. **D94** (2016), no. 11 116015, [arXiv:1510.05626].

[37] S. Abreu, F. Febres Cordero, H. Ita, M. Jaquier, B. Page, and M. Zeng, *Two-Loop Four-Gluon Amplitudes from Numerical Unitarity*, Phys. Rev. Lett. **119** (2017), no. 14 142001, [arXiv:1703.05273].

[38] S. Badger, D. Chicherin, T. Gehrmann, G. Heinrich, J. M. Henn, T. Peraro, P. Wasser, Y. Zhang, and S. Zoia, *Analytic form of the full two-loop five-gluon all-plus helicity amplitude*, Phys. Rev. Lett. **123** (2019), no. 7 071601, [arXiv:1905.03733].

[39] P. Mastrolia and S. Mizera, *Feynman Integrals and Intersection Theory*, JHEP **02** (2019) 139, [arXiv:1810.03818].

[40] H. Frellesvig, F. Gasparotto, M. K. Mandal, P. Mastrolia, L. Mattiazzi, and S. Mizera, *Vector Space of Feynman Integrals and Multivariate Intersection Numbers*, arXiv:1907.02000.

[41] H. Frellesvig, F. Gasparotto, S. Laporta, M. K. Mandal, P. Mastrolia, L. Mattiazzi, and S. Mizera, *Decomposition of Feynman Integrals on the Maximal Cut by Intersection Numbers*, JHEP **05** (2019) 153, [arXiv:1901.11510].

[42] H. Frellesvig, F. Gasparotto, S. Laporta, M. K. Mandal, P. Mastrolia, L. Mattiazzi, and S. Mizera, *Decomposition of Feynman Integrals by Multivariate Intersection Numbers*, arXiv:2008.04823.

[43] X. Liu and Y.-Q. Ma, *Determining arbitrary Feynman integrals by vacuum integrals*, Phys. Rev. **D99** (2019), no. 7 071501, [arXiv:1801.10523].

[44] X. Guan, X. Liu, and Y.-Q. Ma, *Complete reduction of two-loop five-light-parton scattering amplitudes*, arXiv:1912.09294.

[45] P. Zhang, C.-Y. Wang, X. Liu, Y.-Q. Ma, C. Meng, and K.-T. Chao, *Semi-analytical calculation of gluon fragmentation into* $^1S_0^{[1,8]}$ *quarkonia at next-to-leading order*, JHEP **04** (2019) 116, [arXiv:1810.07656].

[46] Y. Wang, Z. Li, and N. Ul Basat, *Direct Reduction of Amplitude*, arXiv:1901.09390.

[47] S. Laporta, *High precision calculation of multiloop Feynman integrals by difference equations*, Int. J. Mod. Phys. **A15** (2000) 5087–5159, [hep-ph/0102033].

[48] A. V. Smirnov, *Algorithm FIRE – Feynman Integral REduction*, JHEP **10** (2008) 107, [arXiv:0807.3243].

[49] A. V. Smirnov, *FIRE5: a C++ implementation of Feynman Integral REduction*, Comput. Phys. Commun. **189** (2015) 182–191, [arXiv:1408.2372].

[50] A. V. Smirnov and F. S. Chuharev, *FIRE6: Feynman Integral REduction with Modular Arithmetic*, arXiv:1901.07808.

[51] P. Maierhoefer, J. Usovitsch, and P. Uwer, *Kira - A Feynman Integral Reduction Program*, Comput. Phys. Commun. **230** (2018) 99–112, [arXiv:1705.05610].

[52] P. Maierhöfer and J. Usovitsch, *Kira 1.2 Release Notes*, arXiv:1812.01491.

[53] A. von Manteuffel and C. Studerus, *Reduze 2 - Distributed Feynman Integral Reduction*, arXiv:1201.4330.

[54] J. Klappert, F. Lange, P. Maierhöfer, and J. Usovitsch, *Integral Reduction with Kira 2.0 and Finite Field Methods*, arXiv:2008.06494.

[55] R. N. Lee and A. A. Pomeransky, *Critical points and number of master integrals*, JHEP **11** (2013) 165, [arXiv:1308.6676].

[56] A. Georgoudis, K. J. Larsen, and Y. Zhang, *Azurite: An algebraic geometry based package for finding bases of loop integrals*, Comput. Phys. Commun. **221** (2017) 203–215, [arXiv:1612.04252].

[57] D. Bendle, J. Böhm, W. Decker, A. Georgoudis, F.-J. Pfreundt, M. Rahn, P. Wasser, and Y. Zhang, *Integration-by-parts reductions of Feynman integrals using Singular and GPI-Space*, JHEP **02** (2020) 079, [arXiv:1908.04301].

[58] J. Usovitsch, *Factorization of denominators in integration-by-parts reductions*, arXiv:2002.08173.

[59] A. Smirnov and V. Smirnov, *How to choose master integrals*, arXiv:2002.08042.

[60] J. Böhm, M. Wittmann, Z. Wu, Y. Xu, and Y. Zhang, *IBP reduction coefficients made simple*, arXiv:2008.13194.

[61] J. Böhm, A. Georgoudis, K. J. Larsen, M. Schulze, and Y. Zhang, *Complete sets of logarithmic vector fields for integration-by-parts identities of Feynman integrals*, Phys. Rev. **D98** (2018), no. 2 025023, [arXiv:1712.09737].

[62] J. Gluza, K. Kajda, and D. A. Kosower, *Towards a Basis for Planar Two-Loop Integrals*, Phys.Rev. **D83** (2011) 045012, [arXiv:1009.0472].

[63] R. M. Schabinger, *A New Algorithm For The Generation Of Unitarity-Compatible Integration By Parts Relations*, JHEP **01** (2012) 077, [arXiv:1111.4220].

[64] K. J. Larsen and Y. Zhang, *Integration-by-parts reductions from unitarity cuts and algebraic geometry*, Phys. Rev. **D93** (2016), no. 4 041701, [arXiv:1511.01071].

[65] J. Böhm, A. Georgoudis, K. J. Larsen, H. Schönemann, and Y. Zhang, *Complete integration-by-parts reductions of the non-planar hexagon-box via module intersections*, JHEP **09** (2018) 024, [arXiv:1805.01873].

[66] H. A. Chawdhry, M. A. Lim, and A. Mitov, *Two-loop five-point massless QCD amplitudes within the IBP approach*, arXiv:1805.09182.

[67] A. von Manteuffel and R. M. Schabinger, *A novel approach to integration by parts reduction*, Phys. Lett. **B744** (2015) 101–104, [arXiv:1406.4513].

[68] T. Peraro, *Scattering amplitudes over finite fields and multivariate functional reconstruction*, JHEP **12** (2016) 030, [arXiv:1608.01902].

[69] T. Peraro, *FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs*, JHEP **07** (2019) 031, [arXiv:1905.08019].

[70] J. Böhm, W. Decker, A. Frühbis-Krüger, F.-J. Pfreundt, M. Rahn, and L. Ristau, *Towards massively parallel computations in algebraic geometry*, Foundations of Computational Mathematics (2020) [arXiv:1808.09727].

[71] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "Singular 4-1-3 — A computer algebra system for polynomial computations." http://www.singular.uni-kl.de, 2020.

[72] F.-J. Pfreundt and M. Rahn, *GPI-Space*, 2018. Fraunhofer ITWM Kaiserslautern, http://www.gpi-space.de/.

[73] D. Bendle, *Massively parallel computation of integration-by-parts relations for Feynman integrals*, 2020. Master's Thesis.

[74] D. Chicherin, T. Gehrmann, J. M. Henn, P. Wasser, Y. Zhang, and S. Zoia, *All master integrals for three-jet production at NNLO*, Phys. Rev. Lett. **123** (2019), no. 4 041603, [arXiv:1812.11160].

[75] Z. Bajnok, J. L. Jacobsen, Y. Jiang, R. I. Nepomechie, and Y. Zhang, *Cylinder partition function of the 6-vertex model from algebraic geometry*, JHEP **06** (2020) 169, [arXiv:2002.09019].

[76] D. Bendle, J. Böhm, Y. Ren, and B. Schröter, *Parallel Computation of tropical varieties, their positive part, and tropical Grassmannians*, arXiv e-prints (Mar., 2020) arXiv:2003.13752, [arXiv:2003.13752].