Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

CARAVEL: A C++ framework for the computation of multi-loop amplitudes with numerical unitarity [☆,☆☆]

S. Abreu [a], J. Dormans [b], F. Febres Cordero [c,*], H. Ita [b], M. Kraus [c], B. Page [d,*], E. Pascual [b], M.S. Ruf [b], V. Sotnikov [e,*]

[a] Center for Cosmology, Particle Physics and Phenomenology (CP3), Université Catholique de Louvain, 1348 Louvain-La-Neuve, Belgium
[b] Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, Hermann-Herder-Str. 3, D-79104 Freiburg, Germany
[c] Physics Department, Florida State University, 77 Chieftan Way, Tallahassee, FL 32306, USA
[d] Institut de Physique Théorique, CEA, CNRS, Université Paris-Saclay, F-91191 Gif-sur-Yvette cedex, France
[e] Max-Planck-Institut für Physik, Werner-Heisenberg-Institut, 80805 München, Germany

A R T I C L E   I N F O

Article history:
Received 7 October 2020
Received in revised form 12 April 2021
Accepted 1 June 2021
Available online 17 June 2021

Keywords:
Numerical unitarity
Multi-loop scattering amplitudes
Gauge theories
Automated tools
QCD
LHC
Gravity

A B S T R A C T

We present the first public version of CARAVEL, a C++17 framework for the computation of multi-loop scattering amplitudes in quantum field theory, based on the numerical unitarity method. CARAVEL is composed of modules for the D-dimensional decomposition of integrands of scattering amplitudes into master and surface terms, the computation of tree-level amplitudes in floating point or finite-field arithmetic, the numerical computation of one- and two-loop amplitudes in QCD and Einstein gravity, and functional reconstruction tools. We provide programs that showcase CARAVEL's main functionalities and allow to compute selected one- and two-loop amplitudes.

Program summary
Program Title: CARAVEL
CPC Library link to program files: https://doi.org/10.17632/rfjrxrb3rk.1
Developer's repository link: https://gitlab.com/caravel-public/caravel.git
Licensing provisions: GPLv3
Programming language: C++
External dependencies:
• *Required:* `Python3` [1], `meson` [2]
• *Optional:* `Doxygen` [3], `Eigen` [4], `GiNaC` [5], `GMP` [6], `Lapack` [7], `MPFR` [8], `MPI` [9], `Pentagon-Library` [10,11], `QD` [12]
Nature of problem: The computation of multi-loop multi-particle scattering amplitudes in quantum field theory
Solution method: The multi-loop numerical unitarity method, functional reconstruction algorithms
Additional comments including restrictions and unusual features: Current version includes tools employed in previous calculations, with the aim of showcasing details of the algorithms employed. Computations are organized by provided data files.

References

[1] http://www.python.org/
[2] https://mesonbuild.com/
[3] http://www.doxygen.nl/
[4] http://eigen.tuxfamily.org/
[5] https://ginac.de/
[6] https://gmplib.org/
[7] http://www.netlib.org/lapack/
[8] https://www.mpfr.org/
[9] https://www.open-mpi.org/

☆ The review of this paper was arranged by Prof. Z. Was.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding authors.
E-mail addresses: ffebres@hep.fsu.edu (F. Febres Cordero), bpage@ipht.fr (B. Page), sotnikov@mpp.mpg.de (V. Sotnikov).

https://doi.org/10.1016/j.cpc.2021.108069

[10] T. Gehrmann, J. M. Henn and N. A. Lo Presti, JHEP 1810 (2018) 103, arXiv:1807.09812 [hep-ph]
[11] https://gitlab.com/caravel-public/pentagon-library
[12] QD: A double-double and quad-double package for Fortran and C++, https://www.davidhbailey.com/dhbsoftware/

Contents

1. Introduction

The computation of scattering amplitudes in quantum field theory is crucial in our quest to describe high-energy particle interactions. Indeed, these objects allow one to make theoretical predictions which can then be compared with experimental measurements, be it at particle colliders such as the Large Hadron Collider (LHC), which are testing the Standard Model of particle physics, or at experiments such as LIGO or VIRGO, which are testing our understanding of gravity. Computing these amplitudes remains a challenge, in particular when they contain many external particles and when higher-order corrections in their perturbative expansion are necessary. The former implies a dependence on a large number of physical scales, and the latter a number of unconstrained *loop momenta* which must be integrated over. Combined, these two aspects lead to a considerable level of complexity in the computation of these amplitudes. This is nevertheless a very timely and important problem to tackle. For example, two-loop five-particle amplitudes as well as three-loop four-particle amplitudes are already relevant for phenomenological studies at the LHC, and will become even more so over the next years.

A major obstacle to overcome when evaluating loop amplitudes is the complexity of intermediate computational steps. In order to bypass this issue, it has proven fruitful to consider numerical approaches. An outstanding example of this is the current possibility to compute general one-loop amplitudes, which has been powered by the introduction of robust numerical techniques. Among many other developments, these techniques are based on integrand reduction approaches [1,2], on the one-loop numerical unitarity method [3–5], and on recursive approaches [6,7]. More recently, there has been great progress in the numerical computation of two-loop amplitudes at high multiplicities. By now, all five-parton amplitudes [8–11] and the amplitudes for four partons and a W boson [12] have been computed numerically at leading color. Through the use of finite fields and multivariate functional reconstruction techniques [13,14], the frameworks powering these numerical computations have furthermore allowed the calculation of the analytic form of all five-parton leading-color amplitudes [15–17], the five-point all-plus amplitude at full color [18] and the four-graviton amplitude in Einstein gravity [19]. In related work, the amplitudes for three-photon production at the LHC have also been computed [20].

In this article we present the Caravel C++ framework. It provides an implementation of many algorithms necessary to perform computations of multi-loop scattering amplitudes within the multi-loop numerical unitarity method. This is the first publicly available code of its kind. It is based on the (generalized) unitarity approach, which was first developed for the analytic computation of one-loop amplitudes [21–24] and later adapted for numeric calculations [3–5]. An extension of the method beyond one loop has been developed recently [25–27]. In a nutshell, in this framework the amplitude is computed starting from a parametrization of its integrand. The corresponding free parameters are numerically computed at each phase-space point by constructing systems of linear equations in which the parameters are the unknowns and the numerical entries are associated to products of tree-level amplitudes. With a suitable choice of integrand parametrization [25], this directly gives a decomposition of the amplitude in terms of master integrals. Finally, after inserting the value of the integrals at the required phase-space point we obtain the value of the amplitude.

The current release of Caravel includes a module for computing products of tree-level amplitudes in several theories through off-shell recursion relations [28], and tools that allow the efficient construction and solution of the systems of linear equations that determine the integrand. Whilst these components work for generic multi-loop amplitudes, other components such as the construction of the parametrization are required as input. In this release we showcase the different available tools by providing a series of example programs. Discussions regarding full automation of the calculation of two-loop massless amplitudes, computations beyond two-loop, as well as the treatment of massive particles, are left to future work. The example programs give a first-hand account of the procedures employed for the calculations presented in refs. [8,10,16,17,19,27].

The rest of this article is organized as follows. Section 2 provides a brief description of our computational methodology, section 3 gives a description of the organization of the internal modules in the Caravel framework, section 4 describes the procedure of installation and setup of the libraries. In section 5 we give details on how to run the example programs that we provide and we conclude in section 6. Appendix A and Appendix B contain technical details about our conventions for color-ordered helicity amplitudes and phase-space parametrizations.

2. Computational methodology

In this section we briefly review the main features of the multi-loop numerical unitarity method. This framework allows for the numerical evaluation as well as the analytical reconstruction of multi-loop scattering amplitudes. Our approach is generic, in that it facilitates the computation of amplitudes in different quantum field theories. When dealing with QCD amplitudes, we consider the amplitudes to be vectors $\mathcal{M}$ in a generic color space. In this section we will mostly be concerned with the components of these vectors, that is with the $\mathcal{A}(\sigma)$ defined as

$$\mathcal{M} = \sum_{\sigma} C(\sigma)\mathcal{A}(\sigma)\,, \tag{1}$$

where the $C(\sigma)$ span the relevant color space. For QCD amplitudes, this is the usual $SU(N_c)$ color space, which can be specialized to $N_c = 3$. For pure gravity processes, no color is present and in eq. (1) there is a single σ with $C(\sigma) = 1$. In the remaining of this section, we will always discuss the $\mathcal{A}(\sigma)$ defined in eq. (1), which for brevity we will call amplitudes, and for simplicity we will drop the σ dependence. In later sections, when discussing the calculation of amplitudes in specific theories, we will be more explicit about the $C(\sigma)$. As is standard in perturbative quantum field theory, we will consider the expansion of $\mathcal{A}$ around a small coupling constant, which is associated to an expansion in the number of loops of the contributing Feynman integrals. We refer the reader to Appendix A for more details on the definition of the objects we compute. For more details on the techniques outlined in this section we refer to previous publications [8,10,16,17,19,27].

2.1. Integrand parametrization

In full generality, an L-loop amplitude $\mathcal{A}^{(L)}$ can be decomposed as a linear combination of master integrals according to

$$\mathcal{A}^{(L)} = \sum_{\Gamma\in\Delta}\sum_{i\in M_\Gamma} c_{\Gamma,i}\, \mathcal{I}_{\Gamma,i}\,. \tag{2}$$

Here, Γ defines a propagator structure associated with the amplitude, and we will often refer to it as a *diagram* (indeed, they are in one-to-one correspondence with scalar Feynman integrals). The set Δ contains all propagator structures Γ, and can be organized hierarchically according to whether a propagator structure $\Gamma_1 \in \Delta$ can be obtained from another $\Gamma_2 \in \Delta$ by removing some propagators in the latter. In this case we write $\Gamma_1 < \Gamma_2$. The set M_Γ denotes the full set of master integrals associated to Γ. Each master integral $\mathcal{I}_{\Gamma,i}$ is usually expressed as a Laurent series in the dimensional-regularization parameter $\epsilon = (4-D)/2$. The coefficients in the Laurent expansion involve multi-valued functions with non-trivial branch-cut structures, such as multiple polylogarithms. The coefficients $c_{\Gamma,i}$ are algebraic functions of the kinematic invariants and rational functions of ϵ.

At its core, the unitarity method is a way to compute the integrand of an amplitude. We therefore start with a parametrization of the integrand $\mathcal{A}^{(L)}(\ell_l)$, where ℓ_l represents the set of L loop momenta, of the form

$$\mathcal{A}^{(L)}(\ell_l) = \sum_{\Gamma\in\Delta}\sum_{k\in Q_\Gamma} c_{\Gamma,k} \frac{m_{\Gamma,k}(\ell_l)}{\prod_{j\in P_\Gamma}\rho_j(\ell_l)}\,. \tag{3}$$

The multiset P_Γ labels all inverse propagators ρ_j in the diagram Γ, and the basis of numerators $\mathcal{Q}_\Gamma = \{m_{\Gamma,k}(\ell_l) | k \in Q_\Gamma\}$ parametrizes every possible integrand up to the maximum allowed power of loop momenta, which is theory specific.

The numerator basis $\mathcal{Q}_\Gamma$ is not unique. Let us highlight two natural classes of bases. First, the simplest choice is what we call a *tensor basis*, denoted by $\mathcal{T}_\Gamma$. It can be built out of independent monomials in a set of variables α_j, which parametrize the loop momenta $\ell_l(\vec{\alpha})$ [29–32]. This type of basis is straightforward to build for generic $\Gamma \in \Delta$. However, with this choice, the relation between eqs. (2) and (3) is not explicit, as it would require solving large systems of integration-by-part (IBP) relations (see e.g. [13,33–41]). A second class of bases is what we call a *master-surface basis* $\mathcal{M}_\Gamma$, and it is crucial to our multi-loop numerical unitarity method. It was observed in ref. [25] that one can parametrize the integrand of a multi-loop amplitude by functions $\mathcal{M}_\Gamma = \{m_{\Gamma,i}(\ell_l) | i \in M_\Gamma \cup S_\Gamma\}$ such that the associated integrands either integrate to zero or correspond to the integrand of a master integral:

$$\int \left(\prod_{j=1}^{L} \frac{d^D \ell_j}{(2\pi)^D} \right) \frac{m_{\Gamma,i}(\ell_l)}{\prod_{k \in P_\Gamma} \rho_k(\ell_l)} = \begin{cases} \mathcal{I}_{\Gamma,i} & \text{for} \quad i \in M_\Gamma \,, \\ 0 & \text{for} \quad i \in S_\Gamma \,. \end{cases} \tag{4}$$

The numerators $m_{\Gamma,i}(\ell_l)$ with $i \in M_\Gamma$ are called *master integrands* and the ones with $i \in S_\Gamma$ *surface terms*. A master-surface basis of functions thus makes the relation between eqs. (2) and (3) explicit. As discussed in [25–27] the construction of master-surface bases of integrands can be efficiently performed by using unitarity-compatible IBP relations [42,43], employing computational algebraic geometry techniques. Notice that the D-dependence of the master/surface terms generates the aforementioned D-dependence of the master-integral coefficients in eq. (2).

2.2. Integrand factorization and cut equations

In order to compute the coefficients $c_{\Gamma,k}$ in eq. (3) we exploit the factorization properties of multi-loop integrands for *on-shell* configurations ℓ_l^Γ of the loop momenta. For a given propagator structure P_Γ, these are defined by

$$\rho_j(\ell_l^\Gamma) = 0 \;\text{ iff }\; j \in P_\Gamma \,. \tag{5}$$

In most cases, this does not fix the ℓ_l^Γ completely, and there is some residual degree of freedom. In this limit, the leading pole of eq. (3) is given by

$$\sum_{\text{states}} \prod_{i \in T_\Gamma} \mathcal{A}_i^{\text{tree}}(\ell_l^\Gamma) = \sum_{\substack{\Gamma' \geq \Gamma \\ k \in Q_{\Gamma'}}} \frac{c_{\Gamma',k} m_{\Gamma',k}(\ell_l^\Gamma)}{\prod_{j \in (P_{\Gamma'}/P_\Gamma)} \rho_j(\ell_l^\Gamma)} \,, \tag{6}$$

where T_Γ represents the set of all tree-level amplitudes corresponding to the vertices of the diagram Γ, and the state sum runs over all D_s-dimensional particle states that can appear in the loop. On the right-hand side the sum runs over all propagator structures Γ' with equal or more propagators than Γ (hence $P_\Gamma \subseteq P_{\Gamma'}$). Eq. (6) is a so-called *cut equation*.

The cut equations allow one to numerically compute the coefficients $c_{\Gamma,k}$ in eq. (3) by sampling a sufficient set of on-shell momenta ℓ_l^Γ and solving the resulting system of linear equations. Importantly, some of these coefficients may be identically zero for all phase-space points. To account for this, we identify zero coefficients during a “warm-up run” on a single phase-space point, and remove the corresponding terms from the ansatz for all subsequent evaluations. In order to construct this system of equations we must have an efficient way to evaluate the tree amplitudes on the left-hand side of the cut equations. This is achieved with an implementation of the Berends-Giele off-shell recursion relations [28], which allows one to recursively compute tree amplitudes with an arbitrary number of legs, and where the particles have D_s-dimensional states. Being a very general approach, it provides a straightforward way to add new types of fields. Beyond one-loop we also need to consider subleading singular contributions for certain propagator structures for which no generic integrand factorization is known (at two loops, this happens for propagator structures where the same propagator appears twice). To address these cases, we employ cut equations with fewer on-shell constraints and then solve for the corresponding coefficients [26].

Another important aspect in sampling the cut equations is the construction of the on-shell momenta ℓ^Γ. These configurations of loop momenta are constructed by solving the quadratic equations in eq. (5), and depending on which number field we use they might not have solutions. To be more precise, we will often do calculations in a field $\mathbb{F}$ that is not algebraically closed, such as the field of rational numbers or a finite field (see section 2.5 below), which makes the construction of on-shell momenta with components in $\mathbb{F}$ a non-trivial problem. However, it turns out that the square roots originating from the solutions of eq. (5) are only present at intermediate steps of the calculation, and any D-dimensional Lorentz-invariant scalar product of the loop momenta lives in $\mathbb{F}$. In particular, the product of trees in eq. (6) is free of square roots and representable in $\mathbb{F}$. Therefore, we (temporarily) employ an algebraic extension of $\mathbb{F}$ for evaluating off-shell currents contributing to the left-hand side of eq. (6). We refer to refs. [10,17,44] for details. In some cases, e.g. for Yang-Mills theory, it is possible to avoid the appearance of square roots altogether [16]. In this case we compute cuts directly in $\mathbb{F}$.

Solving for all coefficients in an amplitude can be efficiently organized in a block-triangular way, using the hierarchical structure of the set Δ. In two-loop five-particle QCD amplitudes each block of equations can have up to a few hundreds of unknowns [8,10], while in Einstein gravity this number is typically an order of magnitude larger. We solve these equations by employing standard linear algebra techniques, such as PLU or QR factorization. Through this procedure, we can compute the coefficients $c_{\Gamma,i}$ in eq. (2) at a numerical phase-space point, and for numerical values of D and D_s.

2.3. Special functions

As stated below eq. (2), the master integrals have a Laurent expansion around $\epsilon = 0$, whose coefficients can be written as linear combinations of multivalued special functions. For all the cases currently implemented in Caravel, the special functions are (linear combinations of) multiple polylogarithms. Using modern mathematical techniques [45,46], we can find a basis B for this space of functions, and find an alternative decomposition of the amplitude in terms of the elements $h_i \in B$. That is, up to a given order k in the ϵ expansion we can write

$$\mathcal{A}^{(L)} = \sum_{j=-2L}^{k} \sum_{i \in B} r_{i,j}\, h_i\, \epsilon^j + \mathcal{O}(\epsilon^{k+1}), \tag{7}$$

where the functions $r_{i,j}$ do not depend on ϵ. This decomposition presents a major difference compared to the one of eq. (2): it allows one to write one- and two-loop amplitudes as a linear combination of the same basis of functions. In turn, this then makes it possible to

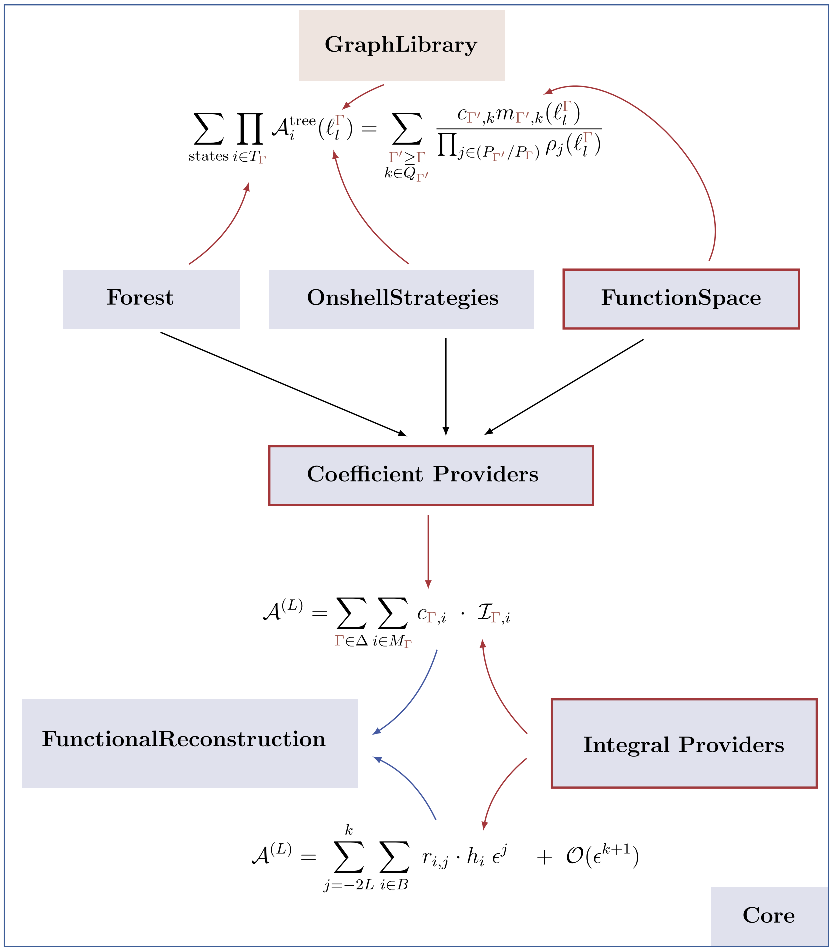
GraphLibrary
$\sum_{\text{states}} \prod_{i \in T_\Gamma} \mathcal{A}_i^{\text{tree}}(\ell_l^\Gamma) = \sum_{\substack{\Gamma' \geq \Gamma \\ k \in Q_{\Gamma'}}} \frac{c_{\Gamma',k} m_{\Gamma',k}(\ell_l^\Gamma)}{\prod_{j \in (P_{\Gamma'}/P_\Gamma)} \rho_j(\ell_l^\Gamma)}$
Forest
OnshellStrategies
FunctionSpace
Coefficient Providers
$\mathcal{A}^{(L)} = \sum_{\Gamma \in \Delta} \sum_{i \in M_\Gamma} c_{\Gamma,i} \cdot \mathcal{I}_{\Gamma,i}$
FunctionalReconstruction
Integral Providers
$\mathcal{A}^{(L)} = \sum_{j=-2L}^{k} \sum_{i \in B} r_{i,j} \cdot h_i \, \epsilon^j + \mathcal{O}(\epsilon^{k+1})$
Core

Fig. 1. Illustration of the interplay of the different modules in Caravel in generic calculations of scattering amplitudes. A black arrow indicates a dependence, a blue arrow means input for a module, and a red arrow the capacity of a module to deliver a given component of the calculation. Modules surrounded by a red box rely on external input to operate. All modules depend on the **Core** module. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

write quantities derived from amplitudes, such as finite remainders, in terms of this basis of functions. This observation was fundamental in reconstructing two-loop five-parton QCD amplitudes [16,17], using the basis of ref. [47], as the coefficients in a decomposition of the form of eq. (7) are much simpler for a two-loop finite remainder than for a two-loop amplitude.

2.4. Analytic structure in the dimensional regulators

The cut equation in eq. (6) depends on dimensional regulators, namely on D, the dimension of the loop momenta, and D_s, the dimension of the states of loop particles. For convenience, we keep these quantities separate until the final stages of the computation. The coefficient functions $c_{\Gamma,i}$ in eq. (2) are rational functions in D and polynomials in D_s (in some very special cases, this dependence can also be rational, see e.g. [19]).

We evaluate the products of tree-level amplitudes in eq. (6) in integer D_s dimensions. To reconstruct the analytic D_s dependence we can employ two different approaches. In the first approach, known as dimensional reconstruction [4,8,10,27,48,49], we extract the polynomial dependence by evaluating the tree-level amplitudes for various integer dimensions D_s and fit the resulting coefficients of the D_s polynomial. This procedure is conceptually straightforward. However, its numerical inefficiency can become a bottleneck for amplitudes with external fermions due to the exponential scaling of the dimension of spinor representations with D_s [10,50]. To address this issue, we employ a second approach, which bypasses the dimensional reconstruction method and provides a diagrammatic representation of the coefficients of the D_s polynomial [17,44,50]. As an example, this strategy reduces the evaluation time of two-loop five-parton amplitudes with two external fermion lines by about two orders of magnitude.

Regarding the dependence on D, we sample at a sufficient number of values of D in order to reconstruct the rational dependence of each master-integral coefficient using Thiele's formula [14,51]. This procedure can be computationally intensive. We note nevertheless that the D-dependence in the denominator of the coefficients is rather simple and independent of the phase-space point. When evaluating the same amplitude over several phase-space points, we thus usually perform a warm-up evaluation, dedicated to determining this dependence for each $c_{\Gamma,k}$ in eq. (3).

The l.h.s. of eq. (6) is evaluated as many times as there are unknowns on the r.h.s. (typically $\mathcal{O}(100)$ in massless two-loop QCD amplitudes and $\mathcal{O}(1000)$ in two-loop gravity amplitudes). On the other hand, the basis of integrands on the r.h.s. is evaluated over $\mathcal{O}(10)$ values of D. It is worth noting that since the l.h.s. of eq. (6) does not depend on D and the r.h.s. of eq. (6) does not depend on D_s, we reconstruct the dependence of each $c_{\Gamma,k}$ on both D and D_s by sampling the sides of the equation independently.[1] In particular, this allows us to significantly reduce the number of tree-amplitude evaluations in cut equations.

2.5. *Finite fields and functional reconstruction*

In ref. [13] it was shown that finite fields can be applied to the Laporta algorithm [34] for the IBP reduction of multi-loop Feynman integrals. The authors were able to not only efficiently perform numerical IBP reductions of integrals, but also to reconstruct the analytic rational dependence of the master integral coefficients in the dimensional regularization parameter ϵ from those numerical reductions. It was later shown in ref. [14] that through a recursive approach one could more generically reconstruct multivariate rational functions. In the same paper, this idea was applied to the computation of scattering amplitudes in generalized unitarity methods. In Caravel, we apply the reconstruction approach to rational coefficient functions in the numerical unitarity method. To do this, we evaluate the amplitudes at rational phase-space points and perform all calculations in a finite field, obtaining exact numerical values for the coefficients. These evaluations can then be used to reconstruct the analytic dependence on the kinematic variables which parametrize the appropriate Lorentz-invariant phase space associated to the amplitude. Caravel contains all the functionalities for the numerical evaluation of scattering amplitudes in a finite field, as well as for the reconstruction of generic rational functions [16,17].[2] These tools have been fundamental for the computation of the planar two-loop five-parton amplitudes [8,10,16,17], as well as the two-loop four-graviton amplitudes [19].

2.6. *Numerical evaluation of scattering amplitudes*

Here we summarize the steps involved in the numerical evaluation of a typical two-loop amplitude with Caravel. As input, Caravel requires a set of analytic information specific to the kinematics/process at hand:

1. A hierarchical organization of diagrams Γ of the color-decomposed amplitude (see. Eq. (1)), the "Process Library".
2. Master-surface integrand parametrizations for all topologically inequivalent diagrams in the process library (see Eqs. (3) and (4)).
3. The relevant master integrals for the process.

This preparatory analytic work is not automated and relies on private computer-algebra programs or explicit calculations. For the processes considered by the example programmes of section 5, this information is provided in the current release of Caravel. Caravel then undertakes the following, two-phase numerical procedure.

A. Warm-up phase
   1. Load the process library and construct the hierarchy of cut equations.
   2. Employ finite fields and a random phase-space point, traverse the hierarchy and perform the following for each entry Γ:
      a. Solve for coefficients $c_{\Gamma,k}$, maintaining full D_s dependence on the l.h.s. of eq. (6) and a fixed value of D in the r.h.s. of eq. (6). Record the set of vanishing integrand coefficients.
      b. If $c_{\Gamma,k}$ is a master integral coefficient, use Thiele's formula to compute the degree of D-dependence.
      c. Store this information in a *warmup* file.

B. Numerical Evaluation
   1. Load process library and construct all cut equations according to the warmup file.
   2. With the requested phase-space point, traverse the hierarchy and determine the $c_{\Gamma,k}$. As described in section 2.4, sample the l.h.s. of eq. (6) for as many times as there are non-zero coefficients on the r.h.s.. Evaluate the r.h.s. on as many values of D as required to obtain all master integral coefficients.
   3. If evaluation is performed using finite field arithmetic, proceed with rational reconstruction of coefficients.
   4. Having determined master integral coefficients, evaluate master integrals and combine with coefficients. Taylor expand in $\epsilon = (4 - D)/2$ and return the value of the amplitude as a Laurent series.

3. Internal modules

The Caravel framework is organized in a modular fashion. The structure of the multi-loop numerical unitarity method outlined in the previous section naturally lends itself to this modular implementation. In Fig. 1 we show how the different modules of Caravel relate to the main equations of the numerical unitarity method as well as to each other. The red arrows show which module constructs each of the different components of these equations. A black arrow from A to B represents a dependence of B on A. The blue arrows highlight an input that a module receives. The modules highlighted with red boxes receive external input to operate, either in the form of data files or as machine-generated source code.

In the following we list the modules of Caravel, such that the reader gets a general view of the source code of the library. We do not give details on application programming interfaces, as in this release we only include specific example programs for the user, as presented in section 5. The modules are:

- **Core**: This module implements general tools for debugging, arithmetic, kinematics, as well as utilities for linear algebra, rational reconstruction, type traits, and more general operations such as Laurent expansions. Among the arithmetic tools, we include interfaces to the `QD` [55] library for double-double and quad-double precision floating-point, and the `GMP` [56] library for arbitrary precision floating-point as well as arbitrary size rational numbers. Furthermore, we use an in-house implementation of 32-bit finite fields based

[1] We do not sample D_s when we compute the coefficients of the D_s polynomial directly.
[2] Two publicly available packages for functional reconstruction are `Firefly` [52,53] and `FiniteFlow` [54].

on Barrett reduction [57] which allows finite fields of cardinality in the range $(2^{30}, 2^{31} - 1]$. The module also provides linear algebra facilities for solving the cut equations. These include in-house implementations of linear algebra algorithms for finite fields, as well as interfaces to `Lapack` [58] (for double precision) and to `Eigen` [59] (for the double and the high-precision floating-point numbers provided by `QD` and `GMP`). Furthermore, the module contains implementations of D-dimensional vector and spinor representations, designed to work with both floating-point and exact number types. There is also a parser to process headed lists like those employed in `Mathematica` [60].

- **GraphLibrary**: A module for the classification and canonicalization of multi-loop graphs. In Caravel, many objects can naturally be associated to graphs, such as the propagator structures Γ in eq. (2). Graph isomorphisms are identified by building a partial order in the representation of the graph (which is ultimately based on the standard C++ function `std::lexicographical_compare`).
- **FunctionalReconstruction**: Implementations of algorithms for analytic reconstruction of univariate and multivariate rational functions from exact numerical evaluations (see section 2.5). The reconstruction algorithms are parallelized using either native C++ threads or using `MPI`. The latter is useful for use on computer clusters.
- **OnShellStrategies**: Tools to generate on-shell loop momenta for one- and two-loop diagrams, as required for the construction of the linear system of equations in eq. (6).
- **Forest**: Implementation of the Berends-Giele off-shell recursion [28] for the computation of general tree-level amplitudes and the products of trees on the left-hand side of eq. (6) in arbitrary D_s dimensions. The recursions can be constructed from any given set of Feynman rules[3] and can be evaluated over an arbitrary numerical type (e.g. floating point of different sizes, finite fields, etc.). This release of Caravel includes the Feynman rules for massless QCD and Einstein gravity.
- **FunctionSpace**: Module for the construction of the integrand ansätze of eq. (3), both for tensor bases $\mathcal{T}_\Gamma$ and master-surface bases $\mathcal{M}_\Gamma$. The former can be constructed for an arbitrary two-loop diagram Γ and are used for development/testing purposes. The latter are provided for arbitrary one-loop diagrams and only for the two-loop diagrams required for the calculations in [8,10,16,17,19,27]. Master-surface bases have been produced with private computer-algebra programs, and transformed into C++ code to be handled by this module.
- **Integral providers**: Two separate modules are included to handle analytic expressions of master integrals. The main module is the **IntegralLibrary**, which provides access to the master integrals associated to four- and five-point one- and two-loop planar massless master integrals. In the five-point case, integrals are written in terms of *pentagon functions* [47]. Internally, all master integrals are normalized as
$$\mathcal{I}_{\Gamma,i} = \left(\frac{e^{\gamma_E \epsilon}}{i\pi^{D/2}}\right)^L \int \prod_{j=1}^{L} d^D\ell_j \, \frac{m_{\Gamma,i}(\ell_l)}{\prod_{k\in P_\Gamma} \rho_k(\ell_l)} \tag{8}$$
where γ_E is the Euler-Mascheroni constant. In the four-point case, master integrals are evaluated with `GiNaC` [61–63] and `CLN` [64]. In the five-point case, we currently employ a modified version of the library provided in [47]. All master integrals are stored in a format that allows on-the-fly expression of an amplitude in terms of a set of basis functions as described in section 2.3, see eq. (7). In the current version all master integrals are implemented in the Euclidean region. Additionally, Caravel contains the **IntegralsBH** module, which provides a large collection of one-loop integrals up to $O(\epsilon^0)$ (including massive and massless propagators) which can be evaluated in up-to quad-double precision using the `QD` [55] package. This library has been adapted from the `BHlibMassive` library employed in [65].
- **Coefficient providers**: Two different modules perform the hierarchical extraction of master-integrand coefficients via the cut equations (see section 2.2). The **AmpEng** provider computes general one-loop integral coefficients, building up the hierarchy of diagrams Δ in an automated fashion. For two-loop calculations, the **AmplitudeCoefficients** module is employed. For a given amplitude, it requires an input data file, which we call the *process library*. These process libraries contain all hierarchical relations between the propagator structures in the amplitude, as well as information about the color decomposition [66,67] and the D_s-dependence based on the particle content [17,44,50]. In this release, we provide the process libraries employed for the calculations in [8,10,16,17,19,27], which were produced with private computer-algebra code.
- **Other modules**: further minor modules include miscellaneous functionalities, for example some phase-space generators (including momentum twistor parametrizations [68]) and information on the pole structure of relevant amplitudes.

All modules in Caravel are implemented according to the concept of generic programming, in which algorithms are designed to operate on any data type satisfying certain (minimal) requirements. In particular, our algorithms are well equipped to work with any numerical type, such as floating point number (of fixed or variable size) or numbers in an algebraic field (the rational numbers or numbers in a finite field). This allows us to perform evaluations of amplitudes with different fixed-size floating point numbers and the reconstruction of their analytic form from exact evaluations over finite fields with essentially a single implementation.

4. Installation and setup

The source code of Caravel can be obtained from a `git` repository at

https://gitlab.com/caravel-public/caravel.git.

Caravel employs the `meson` [69] build system, which relies on `pkg-config` for resolving dependencies. For more details on dependence resolution, configuration options and the installation of optional libraries see the `README.md` and `INSTALL.md` files in the repository. To build Caravel in the default configuration and install it to the directory `<install dir>` one can proceed as shown in Listing 1. All available test suites can be run with the command

[3] In particular, there is no restriction on the number of particles in vertices.

```
1  > git clone https://gitlab.com/caravel-public/caravel.git
2  > cd caravel
3  > mkdir build
4  > cd build
5  > meson .. -D prefix=<install dir>
6  > ninja
7  > ninja install
```

Listing 1: Obtaining and building CARAVEL in its default configuration.

```
> ninja test
```

executed in the build directory. Note that, depending on the hardware and build configuration, running all test suites can take a considerable amount of time. This is particularly noticeable for the first time tests are run which produce and store warm-up information.

The default configuration of CARAVEL provides very limited functionalities. This allows users to customize the installation to suit their particular needs, and additional configuration options and their current values can be queried by running

```
> meson configure
```

in the build directory. An option `<option>` can be set to a particular value `<value>` with the command

```
> meson configure -D <option>=<value>
```

These options can be set either at the configuration stage (step 5 of Listing 1), or any time before the building is initiated (step 6 of Listing 1). It is possible to specify several options at the same time.

Some of the options, listed first when running `meson configure`, are related to generic C++ compiler and linker options, which are automatically provided by the `meson` build system. These options are intended mostly for developers.

Options specific to CARAVEL can be found in the section *Project options*, which enable additional features. We first describe the set of options most relevant for a user of the example programs.

- **double-inverter**: Switch between `Eigen` and `Lapack` for solving linear systems in double precision.
- **finite-fields**: Enable computation using finite fields. Requires the external library `GMP`. This option can be set to `false` (default) or `true`.
- **field-ext-fermions**: Enable the exact computation of master integral coefficients for amplitudes involving fermions. Off-shell currents are then evaluated in an algebraic extension of the number field in order to handle square roots originating from the solutions of quadratic constraints for on-shell momenta (see section 2.2). The evaluation of cuts in the algebraic extension is slower than in the corresponding field. For this reason, if only amplitudes in Yang-Mills theory are of interest, the option should be left at the default value `false`.
- **gravity-model**: Enable/select gravity models. Possible choices are `none`, `Cubic`, `EH`, `EH-GB-R3` and `all`. The last two options increase compilation times considerably. These options give access to the computations of ref. [19], in particular to our implementation of the cubic formulation of the Einstein-Hilbert Feynman rules of [70].
- **precision-QD**: Enable the computation of master-integral coefficients and integrals in double-double (`-D precision-QD=HP`) or quad-double (`-D precision-QD=VHP`) floating-point precision. In this case floating-point types are provided by the `QD` library [55] and linear systems are solved with `Eigen` [59]. Allowing computations in both double-double and quad-double precision requires setting `-D precision-QD=all`. The default is `none`.
- **integrals**: Choose whether or not to compile the master integrals of the module **IntegralLibrary**. The default is `none` which means that no master integrals are compiled. If `goncharovs` is selected, `GiNaC` [63] is required. If `pentagons` is selected the `PentagonLibrary` [47] is required.[4] The choice `all` compiles both representations of the master integrals.
- **lapack-path**: If necessary, specifies the path of `Lapack` if `meson` is unable to find the path to the library.

Beyond these, there are a number of options mainly useful for development:

- **caravel-debug**: Enable a dynamic handling of debugging information from specific source files. To use this feature, simply place a file named *debug.dat* in the directory where the corresponding binary is run. The file should contain the filenames of the source files (without the full path to it), which should provide additional debugging information. One filename should be listed per line and lines starting with # are ignored. This option can be set to `false` (default) or `true`.
- **doxygen**: Enable the generation of HTML API documentation. This requires `Doxygen` [72]. This option can be set to `false` (default) or `true`.
- **ds-strategy**: Select the algorithm for the reconstruction of the dependence of the two-loop amplitudes on the dimensional regulator D_s (see section 2.4). Possible values are `decomposition` (default), referring to the decomposition by particle content, and `sampling`, referring to the reconstruction of the D_s polynomial coefficients from the sample values. The former provides a significant efficiency boost so we recommend not to change its default value. The option `decomposition` is currently not supported for gravity amplitudes.
- **instantiate-rational**: Enable selected computations with rational numbers. This option requires **finite-fields** set to `true`. This option can be set to `false` (default) or `true`.

[4] The modified version of the `PentagonLibrary` that is employed in CARAVEL can be obtained from https://gitlab.com/caravel-public/pentagon-library.git. Notice the recent release of the new `PentagonFunctions++` library [71].

- **precision-arbitrary**: Enable computation with arbitrary-precision floating-point types included in the `GMP` and `MPFR` libraries. This option can be set to `false` (default) or `true`.
- **timing**: Enable the printout of the time spent in different contributions to amplitude's calculations at the end of each program. This option can be set to `false` (default) or `true`.

Enabling some of these features introduces dependencies on third-party libraries, which the user should make available before the start of the building process. Since certain options may significantly increase build times, we suggest to enable only the features necessary for each calculation. Depending on the chosen configuration, building Caravel can take from a few minutes up to half an hour on a modern multi-core processor.

5. Example programs

In this section we introduce the example programs provided with this release, which demonstrate the main features of Caravel. For each one we specify the required configuration setup, a brief explanation of the computations it performs and instructions for execution. More information about these programs can be found in the file `examples/README.md`, contained in the Caravel repository. All programs can be found in `build/examples`, where `build` is the build directory created in Listing 1. Before turning to the description of each example, we first establish some conventions and explain the general structure of the command-line input that the user must provide.

5.1. Helicity amplitudes

Each particle q in a multi-particle helicity amplitude is labeled by the particle type f_q (here, f_q can be a gluon, an (anti)quark, or a graviton), the helicity state h_q and the color index c_q (for color-charged particles). An n-particle amplitude depends on all this data, that is

$$\mathcal{M}_n \equiv \mathcal{M}_n(1_{f_1}^{c_1,h_1}, \ldots, n_{f_n}^{c_n,h_n}). \tag{9}$$

We assign a momentum index q to particle q. All particles and their momenta are considered outgoing. Note that the color indices c_q are not present in pure graviton amplitudes. We consider the perturbative expansion of the (bare) helicity amplitudes and write

$$\mathcal{M}_n = a_0^{\lambda}\left(\mathcal{M}_n^{(0)} + \left(\frac{a_0}{4\pi}\right)^2 \mathcal{M}_n^{(1)} + \left(\frac{a_0}{4\pi}\right)^4 \mathcal{M}_n^{(2)} + \cdots\right), \tag{10}$$

where a_0 is a generic bare coupling constant and λ denotes the power of the leading-order amplitude. For QCD amplitudes, the expansion is in powers of the strong coupling, i.e. $a_0 = g_0$, where $g_0 = \sqrt{4\pi\alpha_s}$. For Einstein gravity amplitudes, instead, $a_0 = \kappa_0/2$, where $\kappa_0 = \sqrt{32\pi G_N}$ and G_N is Newton's constant. In this section we will often refer to the index L in $\mathcal{M}_n^{(L)}$ as the loop order, as for the examples we will consider the two numbers are aligned. $L = 0$ corresponds to tree-level amplitudes.

As already stated in section 2, see in particular eq. (1), Caravel computes the coefficients $\mathcal{A}_n^{(L)}$ of the decomposition of $\mathcal{M}_n^{(L)}$ in terms of a set of color structures. For gravity amplitudes, this decomposition is trivial and Caravel directly computes the $\mathcal{M}_n^{(L)}$. More generally, to properly define the helicity amplitudes $\mathcal{A}_n^{(L)}$ we must specify several of our conventions and this is done in detail in Appendix A. These conventions are important for defining the output of the example programs described in this section.

The example programs we provide compute tree-level, one-loop and two-loop amplitudes. While the tree-level example program allows one to evaluate amplitudes with an arbitrary number of particles, the one-loop and two-loop example programs evaluate at most five-point amplitudes, up to order ϵ^2 for one loop and ϵ^0 for two loops. The integral normalization for the amplitudes computed numerically in the example programs is defined by equation (10). We stress that this differs from the internal normalization in eq. (8). Indeed, the integrals in the example programs are normalized as

$$\mathcal{I}_{\Gamma,i} = \int \left(\prod_{j=1}^{L} \frac{d^D \ell_j}{(2\pi)^D}\right) \frac{m_{\Gamma,i}(\ell_l)}{\prod_{k\in P_\Gamma} \rho_k(\ell_l)}. \tag{11}$$

Additionally, the QCD loop amplitudes are evaluated in the leading-color limit of QCD. In this limit, we keep the leading term of eq. (1) in the limit of a large number of colors N_c, but consider the ratio N_f/N_c to be fixed, where N_f is the number of massless flavors. The leading-color amplitudes have a decomposition in terms of powers of this ratio, specifically

$$\mathcal{A}^{(L)}(1_{p_1}^{h_1}, \ldots, n_{p_n}^{h_n}) = \mathcal{A}^{(L)[0]} + \frac{N_f}{N_c}\mathcal{A}^{(L)[1]} + \ldots + \left(\frac{N_f}{N_c}\right)^L \mathcal{A}^{(L)[L]}. \tag{12}$$

5.2. Specifying program input

Many of the programs that we provide can be run for a variety of different scattering amplitudes. To evaluate different amplitudes, we provide a uniform interface by passing (arbitrarily-ordered) command-line arguments. These arguments are formatted similarly to `Mathematica` lists with heads.

Specifying particles. In the command-line interface, a particle is specified as

```
Particle[field, index, state]
```

Table 1
Allowed `field/state` pairings in a `Particle` list.

Type	`field`	`state`
Gluon	gluon	p, m
Quark	q, u, d, c, s, b	qbp, qm
Anti-quark	qb, ub, db, cb, sb, bb	qbp, qm
Graviton	G	hpp, hmm

Table 2
Examples of valid `PartialAmplitudeInput` lists, all related to leading-color QCD amplitudes. The quotation marks are required to keep the line breaks in a shell execution.

Amplitude	`PartialAmplitudeInput`
$\mathcal{A}^{(2)[2]}(1_g^+, 2_g^+, 3_g^+, 4_g^+, 5_g^+)$	`"PartialAmplitudeInput[ Particles[ Particle[gluon,1,p], Particle[gluon,2,p], Particle[gluon,3,p], Particle[gluon,4,p], Particle[gluon,5,p] ], NfPower[2] ]"`
$\mathcal{A}^{(L)[0]}(1_q^+, 2_{\overline{q}}^-, 3_g^+, 4_g^-)$	`"PartialAmplitudeInput[ Particles[ Particle[q,1,qbp], Particle[qb,2,qm], Particle[gluon,3,p], Particle[gluon,4,m] ] ]"`
$\mathcal{A}^{(L)[0]}(1_u^+, 2_{\overline{u}}^-, 3_d^+, 4_{\overline{d}}^-)$	`"PartialAmplitudeInput[ Particles[ Particle[u,1,qbp], Particle[ub,2,qm], Particle[d,3,qbp], Particle[db,4,qm] ] ]"`

A (momentum) `index` needs to be provided for each particle, starting from 1 up to the number of particles in the scattering process. Gluons are specified by setting the `field` to `gluon`, and `state` should be set to either `p` (+ helicity) or `m` (− helicity). For massless quarks, CARAVEL offers multiple possibilities. Generic unflavored quarks and their anti-particles are input as `q` and `qb` respectively. Fields of definite flavor can also be specified with `u,d,c,s,b` and `ub,db,cb,sb,bb`. We note that scattering processes with identical fermion flavors are currently not supported. However, these can be obtained from anti-symmetrizing distinct flavor amplitudes (see for example [73–75]). The associated `states` for fermions are labeled with `qbp` (+ helicity) and `qm` (− helicity). The graviton field is labeled by `G` and its polarization states are given by `hmm` (−− helicity) and `hpp` (++ helicity). In Table 1 we summarize the available `field` and `state` options to define particles in CARAVEL.

Specifying amplitudes. The simplest scattering amplitudes are color-ordered tree-level helicity amplitudes $\mathcal{A}^{(0)}(1_{f_1}^{h_1}, \ldots, n_{f_n}^{h_n})$. Such a tree-level amplitude is specified in CARAVEL by an ordered list of particles

```
Particles[Particle[..],Particle[..],Particle[..],..]
```

where each particle is defined as in the previous section. Note that color-ordered amplitudes are invariant under cyclic permutations of the external particles. For gravity the same interface is used, however the ordering of the external gravitons does not matter.

For loop amplitudes, the partial amplitude $\mathcal{A}^{(L)[k]}$ of equation (12) can be specified in the command-line interface by

```
PartialAmplitudeInput[Particles[Particle[..], ..], NfPower[k]]
```

where `NfPower[k]` corresponds to the desired power of N_f. This entry is optional, and if it is omitted then the N_f^0 contribution is computed. For gravity amplitudes this entry is meaningless and should be omitted. In Table 2 we give examples of valid `PartialAmplitudeInput`.

Specifying kinematics. By default, most of the examples we provide evaluate amplitudes on phase-space points which allow one to reproduce the results of ref. [10]. Nevertheless, the user can request evaluations at different phase-space points. Since internally most of the example programs perform calculations in a finite field, one has to be sure that the momenta associated with the chosen phase-space point can be represented in a finite field. Finding such points is in general a non-trivial problem. For all the examples we will be concerned with, however, it can be solved by using a special parametrization of phase-space, called *momentum twistor* parametrization [68]. In Appendix B, we give more details on this parametrization for four- and five-point massless kinematics. In particular, we give equations that relate our choice of twistor parameters to Mandelstam variables $s_{ij} = (p_i + p_j)^2$, where p_i denotes the momentum of the external particle with particle index i.

To evaluate amplitudes at a chosen phase-space point, the user should provide a list with the head `TwistorParameters`. For four-point kinematics, the phase space is directly parametrized by the Mandelstam variables `s12` and `s23` which are passed as

```
TwistorParameters[s12, s23]
```

For five-point kinematics the parametrization is given in terms of the 5 independent twistor parameters `x0`, ..., `x4`, see Appendix B for more details. These are passed as

```
TwistorParameters[x0, x1, x2, x3, x4]
```

5.3. *Numerical amplitude evaluation*

In this subsection we present a series of programs which allow the numerical evaluation of a number of QCD scattering amplitudes at tree level, one loop and two loop as well as graviton amplitudes at tree level.

5.3.1. *Tree level*

The program `treeamp` evaluates tree-level amplitudes for a variety of processes. It can be executed by specifying the corresponding amplitude with a `Particles` list (see section 5.2). For example:

```
 > ./treeamp "Particles[Particle[..],...,Particle[..]]"
```

The program randomly generates a phase-space point and prints the value of the specified tree-level amplitude as well as the point. In Appendix A.2 we provide details about the normalization employed for external helicity states, which are necessary to specify our phase conventions.

If `treeamp` is to be used to evaluate N-graviton scattering amplitudes, Caravel must be configured with the option

```
-D gravity-model=Cubic
```

Instead of evaluating the amplitudes on randomly-generated phase-space points the user can also provide external momenta by placing a file with the name `treeampPSP.dat` into the same directory as the executable. The file should list one momentum per line in the format:

```
E PX PY PZ
```

`treeamp` will find this file, read the momenta and, after performing on-shell and momentum-conservation checks, will compute the corresponding amplitude.

If Caravel has been configured to include high-precision floating-point types (with the option `precision-QD` set to `HP`, `VHP` or `all`), then high-precision amplitudes can be computed by passing an additional command-line input

```
 > ./treeamp "Particles[...]" "HighPrecision[prec]"
```

where `prec` is either `HP` for double-double or `VHP` for quad-double precision.

Finally, the program can be used to compute tree amplitudes using finite-field arithmetic (`-D finite-fields=true` required). In this case the cardinality of the finite field has to be passed as a parameter to the program

```
> ./treeamp "Particles[...]" "Cardinality[p]"
```

where `p` is a prime number smaller than 2^{31} and larger than 2^{30}, such as $\texttt{p} = 2^{31} - 1 = 2147483647$. For finite-field evaluations, the program does not randomly generate phase-space points and so the user must provide a set of valid external momenta using a file named `treeampPSP.dat` as explained above. Notice that in the case of finite-field evaluations we use a space-time metric with alternating signature $(+, -, +, -)$ to enhance performance by rendering spinors real (see Appendix A.2 for details). Thus, the momentum components should be provided in this signature. For example, using `treeampPSP.dat` we can pass the rational phase-space point

```
  1/3        1/3    -2        2
 -5/16       1/4    -9/64     15/64
329/144  -355/144   17/16    -25/48
-83/36    271/144   69/64   -329/192
```

to evaluate four-point amplitudes for the momenta

$$p_1 = \left(\frac{1}{3}, \frac{1}{3}, -2, 2\right), \qquad p_2 = \left(-\frac{5}{16}, \frac{1}{4}, -\frac{9}{64}, \frac{15}{64}\right),$$
$$p_3 = \left(\frac{329}{144}, -\frac{355}{144}, \frac{17}{16}, -\frac{25}{48}\right), \qquad p_4 = \left(-\frac{83}{36}, \frac{271}{144}, \frac{69}{64}, -\frac{329}{192}\right),$$

corresponding to $s_{12} = -3/4$ and $s_{23} = -1/4$. The program converts the rational momentum components into their image in the chosen finite field.

5.3.2. One-loop amplitude to $\mathcal{O}(\epsilon^2)$

The program `amplitude_evaluator_1l` numerically evaluates one-loop four- and five-parton helicity amplitudes up to order $\mathcal{O}(\epsilon^2)$ in the dimensional regulator. The master-integral coefficients are rationally reconstructed from finite-field evaluations. The integrals are then computed (in double precision) to obtain the numerical value for the amplitude. If the corresponding tree-level amplitude is non-vanishing the one-loop result is normalized by this tree-level amplitude. Otherwise, the result is normalized by a spinor weight as defined in Appendix A.3, see in particular eq. (A.17). The Laurent expansion of the amplitude is printed to the standard output.

By default, the program runs on the phase-space points defined in ref. [10] so that the user can easily reproduce the results in tables 3 and 4. In the current implementation, this example program runs only in the Euclidean region of phase space as Caravel does not include the analytic continuation of the integrals. We note that the evaluation of five-point amplitudes takes a considerable amount of time because of the evaluation of the one-loop pentagon integral through order ϵ^2.

In order to enable this example, Caravel has to be configured with the following options:

```
-D finite-fields=true
-D field-ext-fermions=true
-D integrals=all
```

The program can be executed by passing the appropriate amplitude input and kinematic point specifications. For example:

```
> ./amplitude_evaluator_1l "PartialAmplitudeInput[Particles[
 Particle[gluon,1,m], Particle[gluon,2,m],
 Particle[gluon,3,p], Particle[gluon,4,p]]]" \
 "TwistorParameters[-1/3, -1/5]"
```

evaluates the one-loop color-ordered helicity amplitude $\mathcal{A}^{(1)[0]}(1_g^-, 2_g^-, 3_g^+, 4_g^+)$ at $s_{12} = -\frac{1}{3}$ and $s_{23} = -\frac{1}{5}$.

5.3.3. Leading-color two-loop amplitude

The program `amplitude_evaluator_2l` numerically evaluates two-loop four- and five-parton helicity amplitudes to $\mathcal{O}(\epsilon^0)$ in the dimensional regulator. It works in the same way as the one-loop program described above. The master-integral coefficients are rationally reconstructed from finite-field evaluations and subsequently combined with the master integrals into a semi-analytic object, which is expressed in terms of unevaluated special functions. Next, those special functions are evaluated to produce the amplitude whose Laurent expansion is then printed to the standard output. The computation of the master-integral coefficients is performed in parallel, using all available threads in the CPU, then the special functions are evaluated sequentially in a single thread. As at one-loop, the amplitude is normalized either by the corresponding non-vanishing tree-level amplitude or by the spinor-weight defined in Appendix A.3, see in particular eq. (A.17). On the first evaluation of each amplitude, the program performs a warm-up run, as described in sections 2.2 and section 2.4.

By default, the program runs on phase-space points defined in ref. [10] and can be used to reproduce the results presented in the tables 1 and 2. As at one-loop, this example program is restricted to the Euclidean region of phase space, as master integrals are so far included only for this region.

In order to enable this example the Caravel library has to be configured with the following options:

```
-D finite-fields=true
-D field-ext-fermions=true
-D integrals=all
```

The program can be executed in an analogous way to what was described in the previous example in section 5.3.2. Additionally, the user can pass the argument `Verbosity[All]` in order to print to the standard output extra information on the computations performed. For example:

```
> ./amplitude_evaluator_2l "PartialAmplitudeInput[Particles[
 Particle[gluon,1,m], Particle[gluon,2,m],
 Particle[gluon,3,p], Particle[gluon,4,p]],NfPower[1]]" \
 "TwistorParameters[-1/3, -1/5]"
```

evaluates the two-loop color-ordered helicity amplitude $\mathcal{A}^{(2)[1]}(1_g^-, 2_g^-, 3_g^+, 4_g^+)$ for $s_{12} = -\frac{1}{3}$ and $s_{23} = -\frac{1}{5}$.

The evaluation of two-loop amplitudes is considerably more involved than the corresponding one-loop amplitudes. For example, the runtime of the most complex two-loop five-parton amplitude is of the order of 12 minutes to rationally reconstruct all master-integral coefficients on the default phase-space point (using 22 finite-field evaluations), while the computation of the pentagon functions in double precision by the external library provided with ref. [47] takes about 23 minutes (employing a modern 12-core Intel i7 processor).

5.3.4. Leading-color five-point two-loop finite remainder

The program `finite_remainder_2l` numerically computes the finite remainder of planar two-loop five-parton amplitudes. The numerical calculation of finite remainders was instrumental in order to reconstruct the analytic form of the planar two-loop five-parton amplitudes in refs. [16,17]. The program proceeds by building the requested two-loop amplitude and its corresponding infrared subtraction, which also requires the one-loop amplitude to $\mathcal{O}(\epsilon^2)$. The finite remainder obtained in this way is decomposed in terms of the special functions introduced in ref. [47]. As in the two-loop amplitude evaluation example, the computation of the two-loop master integral coefficients is performed in parallel, using all available threads in the CPU and, on the first evaluation, a warm-up run will be performed for the two-loop amplitude. The program automatically verifies that all poles in ϵ cancel exactly, rationally reconstructs the special function coefficients in the finite remainder and subtraction term, and evaluates the special functions. The program then prints the values of the subtraction term, the remainder and the amplitude. We refer to ref. [17] for the precise definition of our subtraction conventions. We note that the run-time of this program is comparable to that of the two-loop amplitude evaluation program.

Table 3
Table of numerator labels used in output files and their explicit expression in momentum space. The conventions for `dbTensor` are specified in Fig. 2.

`NumeratorLabel`	Master-integral numerator
`scalar`	1
`mu2`	μ^2
`mu4`	$(\mu^2)^2$
`dbTensor`	$(p_1+\ell_2)^2$

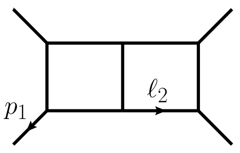

Fig. 2. Specifying momentum routing for the double-box tensor integral.

In order to enable this example Caravel has to be configured with the following options:

```
-D finite-fields=true
-D field-ext-fermions=true
-D integrals=pentagons
```

This example can also be enabled with the `integrals` option set to `-D integrals=all`. The program has similar command-line arguments as in the examples above. For example, it can be executed with

```
> ./finite_remainder_2l "PartialAmplitudeInput[Particles[
 Particle[gluon,1,p],Particle[gluon,2,p],Particle[gluon,3,p],
 Particle[gluon,4,p],Particle[gluon,5,p]],NfPower[2]]"
```

in order to compute the finite remainder for the color-ordered two-loop amplitude $\mathcal{A}^{(2)[2]}(1_g^+,2_g^+,3_g^+,4_g^+,5_g^+)$. As in the two previous examples, the phase-space point can be specified in the command-line input as a list with the head `TwistorParameters`. The optional argument `Verbosity[All]` or `Verbosity[Remainder]` can be passed in the command line to request the printing of additional information.

5.4. Analytic reconstruction of amplitudes

An important application of the numerical techniques for amplitude calculation that Caravel provides is the reconstruction of analytic results from numerical evaluations. Here we present two programs which reconstruct either two-loop four-parton or one-loop five-parton amplitudes from numerical evaluations. These examples rely on `MPI` for parallelization.

5.4.1. Program output

The two analytic reconstruction example programs share common output features, which we describe here. Both programs perform the analytic reconstruction over a single finite field and attempt to rationally reconstruct the result. The computation in the finite field is saved as a text file in the local directory `analytics/amplitudes_XY/`, where `XY` refers to the relevant cardinality. The master-integral coefficients are then rationally reconstructed employing only the single finite-field evaluation performed. This rational reconstruction is cross checked against a numerical computation at a single phase-space point in a second finite field. In the case of a successful reconstruction, the amplitude is saved in a text file under `analytics/amplitudes_rational/`. The reconstructed amplitudes are normalized either by the corresponding non-vanishing tree-level amplitude or by the spinor weight defined in Appendix A.3, see in particular eq. (A.17).

The files produced by each program are named according to the requested `PartialAmplitudeInput`. They contain a string that encodes the decomposition of the amplitude as a linear combination of master integral coefficients and master integrals.[5] Each master integral is specified by a string of the form

```
Topology[NumeratorLabel, {D1, D2, .., Dm}].
```

Here, `Topology` is a human readable name for the topology, e.g. `Triangle` or `DoubleBox`. The list of `Di` is the list of inverse propagators of the master integral, written in terms of the loop momenta (`l1`, `l2`) and external momenta (`k1`, `k2`, ...) of the amplitude. Our conventions in Caravel are that all external momenta are outgoing. The label `NumeratorLabel` denotes the numerator of the master integral. For the example programs provided with this release, there are four possible values for this label, which we list in Table 3. For one-loop integrals, these include numerators built from powers of μ^2, which is the scalar product of the $(D-4)$ dimensional components of the loop momentum, i.e.

$$\mu^2 = \ell^{(D-4)} \cdot \ell^{(D-4)}. \tag{13}$$

[5] The format used in this string allows it to be directly imported into Mathematica for usage if desired.

In order to aid reading of the output, we provide a collection of Mathematica routines in `math/CaravelGraph.m` which produce graphical representations of the integrals. The output format as described above is not appropriate for these routines. The example programs therefore also writes a textfile in the directory `analytics/integral_info/`, which contains a list of replacement rules that allow one to use these routines.

5.4.2. Univariate amplitude reconstruction

As a simple example of the two-loop analytic reconstruction capabilities of Caravel, the program `4parton_2loop_analytics_MPI` analytically computes the reduction to master integrals. The massless four-point amplitudes depend only on the Mandelstam variables $s = (p_1 + p_2)^2$ and $t = (p_2 + p_3)^2$. By setting $s = 1$ and $x = t/s$ the amplitude depends only on a single parameter. Its analytic dependence on x is reconstructed from exact numerical evaluations of the master-integral coefficients over a finite field, which are then fed into Thiele's interpolation formula. The dependence on s is then recovered by dimensional analysis.

To enable this example, Caravel has to be configured with the options

```
-D finite-fields=true
-D field-ext-fermions=true
```

To compute the two-loop four-gluon all-plus-helicity amplitude, for example, the program should be executed with

```
> mpirun -np <ncores> ./4parton_2loop_analytics_MPI \
  "PartialAmplitudeInput[Particles[Particle[gluon,1,p],
  Particle[gluon,2,p],Particle[gluon,3,p],
  Particle[gluon,4,p]]]"
```

5.4.3. Multivariate amplitude reconstruction

The program computes the analytic form of five-parton one-loop amplitudes, using multivariate functional reconstruction. The five-parton amplitudes depend on five twistor parameters $x_0, \ldots, x_4$, see Appendix B. The problem is reduced to a four-dimensional reconstruction by setting $x_4 = 1$. The dependence on x_4 is recovered from dimensional analysis.

In order to enable this example, Caravel has to be configured with the options

```
-D finite-fields=true
-D field-ext-fermions=true
```

To compute the one-loop five gluon all-plus-helicity amplitude, for example, the program should be executed with

```
> mpirun -np <ncores> ./5parton_1loop_analytics_MPI \
  "PartialAmplitudeInput[Particles[Particle[gluon,1,p],
  Particle[gluon,2,p],Particle[gluon,3,p],Particle[gluon,4,p],
  Particle[gluon,5,p]]]"
```

6. Conclusions

We have presented Caravel, a C++ framework for the computation of multi-loop amplitudes through the multi-loop numerical unitarity method. This is the first publicly available program of its kind. We have provided a series of example programs which showcase the main functionalities of Caravel. In particular, these examples give access to all details of the calculation of the planar two-loop five-parton scattering amplitudes [16,17] and the two-loop four-graviton scattering amplitude in Einstein gravity [19].

In its current form, Caravel is not meant to be able to automatically evaluate arbitrary two-loop multi-leg amplitudes. Rather, it is meant as a framework to do so, which should be complemented with process-specific information such as, for instance, the master integrals. The modular fashion in which Caravel is constructed allows one to easily add these new features. We aim to continue the development of Caravel, to increase the pool of multi-loop amplitude computations that it can perform, while also extending its autonomy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We gratefully acknowledge contributions to Caravel made by Matthieu Jaquier during the initial stages of the development. We also wish to thank Mao Zeng for discussions. The work of S.A. is supported by the Fonds De La Recherche Scientifique - FNRS, Belgium. The work of F.F.C. is supported by the U.S. Department of Energy under grant DE-SC0010102. The work of V.S. is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, *Novel structures in scattering amplitudes* (grant agreement No. 725110). The work of B.P. is supported by the French Agence Nationale pour la Recherche, under grant ANR-17-CE31-0001-01. M.S.R.'s work is funded by the German Research Foundation (DFG) within the Research Training Group GRK 2044. The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

Appendix A. Helicity amplitudes in Caravel

In this appendix we summarize several of our conventions, that allow us to precisely define the helicity amplitudes computed by Caravel. In Appendix A.1 we discuss the conventions regarding the color decomposition. In Appendix A.2 we present our conventions for external helicity states. Finally, in Appendix A.3 we discuss the spinor-weight normalization.

A.1. Color decomposition

As already stated in section 2, see in particular eq. (1), Caravel computes the coefficients of $\mathcal{M}_n^{(L)}$ in a decomposition in terms of color structures. For Einstein gravity amplitudes, this decomposition is trivial and Caravel directly computes the $\mathcal{M}_n^{(L)}$. For QCD amplitudes, however, to properly define the objects computed in the example programs we must specify the color decomposition and our conventions for the color algebra. We follow the conventions of ref. [76] and denote the fundamental generators of the $SU(N_c)$ group by $(T^a)_i^{\;\bar{j}}$, where the adjoint index a runs over N_c^2-1 values and the (anti-) fundamental indices i and $\bar{\imath}$ run over N_c values. We use the normalization $\mathrm{Tr}(T^aT^b)=\delta^{ab}$. Finally, we define $F^{abc}=\mathrm{Tr}\big([T^a,T^b,]T^c\big)$, which is closely related to the $SU(N_c)$ structure constants.

Tree-level QCD amplitudes. In order to define in a unified way the color-ordered tree-level amplitudes that we compute, we shall consider QCD with all fields (that is, both quarks and gluons) in the adjoint representation. In this 'adjoint QCD' theory, amplitudes with any number of partons can be expressed in terms of $(n-2)!$ color-ordered partial amplitudes using the decomposition of ref. [77],

$$\mathcal{M}^{(0)}=\sum_{\sigma\in S_{n-2}} C(a_{n-1},a_{\sigma_1},\ldots,a_{\sigma_{n-2}},a_n)\mathcal{A}^{(0)}(n-1^{h_{n-1}}_{p_{n-1}},\sigma 1^{h_{\sigma_1}}_{p_{\sigma_1}},\ldots,\sigma n-2^{h_{\sigma_{n-2}}}_{p_{\sigma_{n-2}}},n^{h_n}_{p_n}). \tag{A.1}$$

Here, S_n denotes all permutations of n indices and C is a color structure given by

$$C(a_1,\ldots,a_n)=F^{a_1a_2x_1}F^{x_1a_3x_2}\cdots F^{x_{n-4}a_{n-2}x_{n-3}}F^{x_{n-3}a_{n-1}a_n}, \tag{A.2}$$

with the F^{abc} defined above. We stress that we simply use adjoint QCD to define quark and gluon amplitudes in a unified way. It is a well understood procedure to assemble the multi-parton QCD amplitude from these color-ordered amplitudes, see for example [67,77,78].

Leading-color QCD loop amplitudes. Beyond tree-level, the example programs compute the color ordered amplitudes relevant for the leading-color limit of QCD. In this limit, we keep the leading term for a large number of colors N_c, but consider the ratio N_f/N_c to be fixed, where N_f is the number of massless flavors.

We first discuss the four-point amplitudes and consider amplitudes for the scattering of four gluons, one quark pair and two gluons, and two distinct quark pairs. In the leading-color approximation we write

$$\mathcal{M}^{(L)}(1_g,2_g,3_g,4_g)\big|_{\text{leading color}}=N_c^L\sum_{\sigma\in S_4/Z_4}\mathrm{Tr}\left(T^{a_{\sigma(1)}}T^{a_{\sigma(2)}}T^{a_{\sigma(3)}}T^{a_{\sigma(4)}}\right)\mathcal{A}^{(L)}(\sigma(1)_g,\sigma(2)_g,\sigma(3)_g,\sigma(4)_g), \tag{A.3}$$

$$\mathcal{M}^{(L)}(1_q,2_{\bar{q}},3_g,4_g)\big|_{\text{leading color}}=N_c^L\sum_{\sigma\in S_2}\left(T^{a_{\sigma(3)}}T^{a_{\sigma(4)}}\right)_{i_1}^{\bar{\imath}_2}\mathcal{A}^{(L)}(1_q,2_{\bar{q}},\sigma(3)_g,\sigma(4)_g), \tag{A.4}$$

$$\mathcal{M}^{(L)}(1_q,2_{\bar{q}},3_Q,4_{\bar{Q}})\big|_{\text{leading color}}=N_c^L\,\delta_{i_3}^{\bar{\imath}_2}\delta_{i_1}^{\bar{\imath}_4}\,\mathcal{A}^{(L)}(1_q,2_{\bar{q}},3_Q,4_{\bar{Q}}), \tag{A.5}$$

where S_n/Z_n denotes all non-cyclic permutations of n indices, and L corresponds to the number of loops of the amplitudes. In the five-point case we write

$$\mathcal{M}^{(L)}(1_g,2_g,3_g,4_g,5_g)\big|_{\text{leading color}}=N_c^L\sum_{\sigma\in S_5/Z_5}\mathrm{Tr}\left(T^{a_{\sigma(1)}}\ldots T^{a_{\sigma(5)}}\right)\mathcal{A}^{(L)}(\sigma(1)_g,\sigma(2)_g,\sigma(3)_g,\sigma(4)_g,\sigma(5)_g), \tag{A.6}$$

$$\mathcal{M}^{(L)}(1_q,2_{\bar{q}},3_g,4_g,5_g)\big|_{\text{leading color}}=N_c^L\sum_{\sigma\in S_3}\left(T^{a_{\sigma(3)}}T^{a_{\sigma(4)}}T^{a_{\sigma(5)}}\right)_{i_1}^{\bar{\imath}_2}\mathcal{A}^{(L)}(1_q,2_{\bar{q}},\sigma(3)_g,\sigma(4)_g,\sigma(5)_g), \tag{A.7}$$

$$\mathcal{M}^{(L)}(1_q,2_{\bar{q}},3_Q,4_{\bar{Q}},5_g)\big|_{\text{leading color}}=N_c^L\,(T^{a_5})_{i_3}^{\bar{\imath}_2}\delta_{i_1}^{\bar{\imath}_4}\,\mathcal{A}^{(L)}(1_q,2_{\bar{q}},5_g,3_Q,4_{\bar{Q}})+N_c^L\,(T^{a_5})_{i_1}^{\bar{\imath}_4}\delta_{i_3}^{\bar{\imath}_2}\,\mathcal{A}^{(L)}(1_q,2_{\bar{q}},3_Q,4_{\bar{Q}},5_g). \tag{A.8}$$

A.2. External helicity states

In this section we collect the conventions used in Caravel for the spinors and polarization states. For floating-point computations, the spinors must be constructed so that they are numerically stable, which means different conventions are chosen depending on the phase-space point. As an example, if p_+ is not small, we take

$$\begin{aligned}u_+(p)=v_-(p)=|p\rangle&=\frac{\sqrt{|p_+|}}{p_+}\begin{pmatrix}p_+\\ p_+^\perp\end{pmatrix},\\ u_-(p)=v_+(p)=|p]&=\frac{1}{\sqrt{|p_+|}}\begin{pmatrix}p_+\\ p_-^\perp\end{pmatrix},\end{aligned} \tag{A.9}$$

where we defined

$$p_+=p^0+p^3,\qquad p_+^\perp=p^1+ip^2,\qquad p_-^\perp=p^1-ip^2. \tag{A.10}$$

We will often use the particle index instead of its momentum to denote the spinors. That is, for a particle with index i of momentum p_i we will write $|i\rangle$ and $|i]$.

For massless external vector bosons with four-momentum p, the polarization states are defined in terms of a light-like reference vector n^μ as

$$\epsilon_+^\mu(p,n) = \frac{\langle n|\bar\sigma^\mu|p]}{\sqrt{2}\,\langle n|p\rangle}\,, \qquad \epsilon_-^\mu(p,n) = -\frac{[n|\sigma^\mu|p\rangle}{\sqrt{2}[n|p]}\,, \tag{A.11}$$

where $\sigma^\mu = (1,\sigma^i)$, $\bar\sigma^\mu = (1,-\sigma^i)$ and σ^i are the Pauli matrices. Amplitudes are independent of the specific choice of auxiliary vector n. Finally, for the computation of gravity amplitudes, the external graviton states are constructed from the polarization vectors given above. The two transverse polarization states read

$$h_{--}^{\mu\nu}(p,n) = \epsilon_-^\mu(p,n)\epsilon_-^\nu(p,n)\,, \qquad h_{++}^{\mu\nu}(p,n) = \epsilon_+^\mu(p,n)\epsilon_+^\nu(p,n)\,, \tag{A.12}$$

where as before p is the four momentum of the graviton and n is a massless auxiliary vector.

In the case of finite-field evaluations, we would like the external helicity states to be rational functions of the momentum components. To achieve this, we exploit the fact that Weyl spinors are defined up to a little group scaling

$$|p\rangle \to z_p\,|p\rangle\,, \qquad |p] \to \frac{1}{z_p}\,|p]\,. \tag{A.13}$$

With $z_p = \sqrt{p_+}$, we obtain

$$\begin{aligned} u_+(p) = v_-(p) = |p\rangle &= \begin{pmatrix} p^0+p^3 \\ p^1-p^2 \end{pmatrix}\,, \\ u_-(p) = v_+(p) = |p] &= \frac{1}{p^0+p^3}\begin{pmatrix} p^0+p^3 \\ p^1+p^2 \end{pmatrix}\,, \end{aligned} \tag{A.14}$$

where the components are given for the alternating signature $(+,-,+,-)$, in which the spinors can be rendered explicitly real. This choice of metric does not affect the value of Lorentz-invariant quantities, such as the normalized amplitudes $A^{(L)}$ defined in eq. (A.17) below.

A.3. Spinor weights of helicity amplitudes

In this section we present our conventions for a spinor-weight normalization which allows one to construct Lorentz-invariant objects from helicity amplitudes in Caravel. We use a generalization of the normalization factor introduced in ref. [79] with the conventions specified in the previous section.

For an n-point amplitude, let $\mathcal{C} = \{h_1,\ldots,h_n\}$ be the sequence of the helicity states of n_v vector bosons and n_f fermion pairs ($n = n_v + 2n_f$), labeled by their particle index. We then construct the spinor weight $\Phi_\mathcal{C}$ associated to $\mathcal{C}$ as

$$\Phi_\mathcal{C} = \prod_{i=1}^{n_v} \omega_{\mathrm{v}_i}^{\mathrm{sign}(h_{\mathrm{v}_i})} \prod_{i=1}^{n_f} \eta_{\mathrm{f}_i^-\mathrm{f}_i^+}\,. \tag{A.15}$$

Here, v and $\mathrm{f}^\pm$ are the (order-preserving) subsequences of the index sequence of $\mathcal{C}$ corresponding to vector-boson states and fermion states with $h_i = \pm\frac{1}{2}$ respectively.[6] The weights $\omega_i^\pm$ and η_{ij} are given by

$$\begin{aligned} &\omega_1^+ = \frac{[12]\,\langle 32\rangle}{\langle 13\rangle}, \qquad \omega_i^+ = \frac{\langle 13\rangle}{\langle i1\rangle^2\,[12]\,\langle 32\rangle} \quad \text{for} \quad i \geq 2, \\ &\omega_i^- = \frac{1}{\omega_i^+}, \qquad \eta_{ij} = \langle i k_{ij}\rangle\,\big[k_{ij}\,j\big]\,, \end{aligned} \tag{A.16}$$

where k_{ij} is the smallest positive integer such that $k_{ij} \neq i$ and $k_{ij} \neq j$. The spinor weights for the graviton helicity states are $\left(\omega_i^\pm\right)^2$. We can then write the amplitudes as

$$\mathcal{A}^{(L)}(1_{p_1}^{h_1},\ldots,n_{p_n}^{h_n}) = \Phi_\mathcal{C}\,A^{(L)}(1_{p_1}^{h_1},\ldots,n_{p_n}^{h_n}) \tag{A.17}$$

where $A^{(L)}(1_{p_1}^{h_1},\ldots,n_{p_n}^{h_n})$ is Lorentz invariant.

[6] Note that this pairs fermion states based strictly on the ordering of $\mathcal{C}$ and disregards any other quantum numbers of the corresponding particles. This might not be what is intuitively anticipated.

Appendix B. Momentum-twistor parametrizations

Here we list explicitly the momentum-twistor parametrization [68] used to rationalize the external on-shell momenta. We provide explicit relations with the Mandelstam invariants $s_{ij} = (p_i + p_j)^2$, where p_i denotes the momentum of the external particle with index i. We recall that the conventions in Caravel are that all external momenta are outgoing.

Four-point kinematics

In order to construct four-point rational momenta we use the momentum-twistor parametrization of ref. [80], which is given by

$$Z = \begin{pmatrix} |1\rangle & |2\rangle & |3\rangle & |4\rangle \\ |\mu_1] & |\mu_2] & |\mu_3] & |\mu_4] \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\frac{1}{s_{12}} & -\frac{1}{s_{12}} - \frac{1}{s_{23}} \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{B.1}$$

With this parametrization the resulting momenta are rational in s_{12} and s_{23}.

Five-point kinematics

For five-point kinematics we use the parametrization given in refs. [9,14]

$$Z = \begin{pmatrix} |1\rangle & |2\rangle & |3\rangle & |4\rangle & |5\rangle \\ |\mu_1] & |\mu_2] & |\mu_3] & |\mu_4] & |\mu_5] \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{1}{x_4} & \frac{1+x_0}{x_0 x_4} & \frac{1+x_1(1+x_0)}{x_0 x_1 x_4} \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & \frac{x_2}{x_0} & 1 \\ 0 & 0 & 1 & 1 & \frac{x_2 - x_3}{x_2} \end{pmatrix} . \tag{B.2}$$

The Mandelstam variables are then parametrized by

$$\begin{aligned} s_{12} &= x_4 \, , \\ s_{23} &= x_2 x_4 \, , \\ s_{34} &= x_4 \left[\frac{(1+x_1)x_2}{x_0} + x_1(x_3 - 1) \right] , \\ s_{45} &= x_3 x_4 \, , \\ s_{51} &= x_1 x_4 (x_0 - x_2 + x_3) \, , \\ \mathrm{tr}_5 &= i\epsilon(p_1, p_2, p_3, p_4) \\ &= x_4^2 \left[x_2(1 + 2x_1) + x_0 x_1 (x_3 - 1) - \frac{x_2(1+x_2)(x_2 - x_3)}{x_0} \right] . \end{aligned} \tag{B.3}$$

References

[1] F. del Aguila, R. Pittau, J. High Energy Phys. 07 (2004) 017, https://doi.org/10.1088/1126-6708/2004/07/017, arXiv:hep-ph/0404120.
[2] G. Ossola, C.G. Papadopoulos, R. Pittau, Nucl. Phys. B 763 (2007) 147–169, https://doi.org/10.1016/j.nuclphysb.2006.11.012, arXiv:hep-ph/0609007.
[3] R.K. Ellis, W.T. Giele, Z. Kunszt, J. High Energy Phys. 03 (2008) 003, https://doi.org/10.1088/1126-6708/2008/03/003, arXiv:0708.2398.
[4] W.T. Giele, Z. Kunszt, K. Melnikov, J. High Energy Phys. 04 (2008) 049, https://doi.org/10.1088/1126-6708/2008/04/049, arXiv:0801.2237.
[5] C.F. Berger, Z. Bern, L.J. Dixon, F. Febres Cordero, D. Forde, H. Ita, D.A. Kosower, D. Maitre, Phys. Rev. D 78 (2008) 036003, https://doi.org/10.1103/PhysRevD.78.036003, arXiv:0803.4180.
[6] F. Cascioli, P. Maierhofer, S. Pozzorini, Phys. Rev. Lett. 108 (2012) 111601, https://doi.org/10.1103/PhysRevLett.108.111601, arXiv:1111.5206.
[7] S. Actis, A. Denner, L. Hofer, A. Scharf, S. Uccirati, J. High Energy Phys. 04 (2013) 037, https://doi.org/10.1007/JHEP04(2013)037, arXiv:1211.6316.
[8] S. Abreu, F. Febres Cordero, H. Ita, B. Page, M. Zeng, Phys. Rev. D 97 (11) (2018) 116014, https://doi.org/10.1103/PhysRevD.97.116014, arXiv:1712.03946.
[9] S. Badger, C. Brønnum-Hansen, H.B. Hartanto, T. Peraro, Phys. Rev. Lett. 120 (9) (2018) 092001, https://doi.org/10.1103/PhysRevLett.120.092001, arXiv:1712.02229.
[10] S. Abreu, F. Febres Cordero, H. Ita, B. Page, V. Sotnikov, J. High Energy Phys. 11 (2018) 116, https://doi.org/10.1007/JHEP11(2018)116, arXiv:1809.09067.
[11] S. Badger, C. Brønnum-Hansen, T. Gehrmann, H.B. Hartanto, J. Henn, N.A. Lo Presti, T. Peraro, PoS LL2018 (2018) 006, https://doi.org/10.22323/1.303.0006, arXiv:1807.09709.
[12] H.B. Hartanto, S. Badger, C. Brønnum-Hansen, T. Peraro, J. High Energy Phys. 09 (2019) 119, https://doi.org/10.1007/JHEP09(2019)119, arXiv:1906.11862.
[13] A. von Manteuffel, R.M. Schabinger, Phys. Lett. B 744 (2015) 101–104, https://doi.org/10.1016/j.physletb.2015.03.029, arXiv:1406.4513.
[14] T. Peraro, J. High Energy Phys. 12 (2016) 030, https://doi.org/10.1007/JHEP12(2016)030, arXiv:1608.01902.
[15] S. Badger, C. Brønnum-Hansen, H.B. Hartanto, T. Peraro, J. High Energy Phys. 01 (2019) 186, https://doi.org/10.1007/JHEP01(2019)186, arXiv:1811.11699.
[16] S. Abreu, J. Dormans, F. Febres Cordero, H. Ita, B. Page, Phys. Rev. Lett. 122 (8) (2019) 082002, https://doi.org/10.1103/PhysRevLett.122.082002, arXiv:1812.04586.
[17] S. Abreu, J. Dormans, F. Febres Cordero, H. Ita, B. Page, V. Sotnikov, J. High Energy Phys. 05 (2019) 084, https://doi.org/10.1007/JHEP05(2019)084, arXiv:1904.00945.
[18] S. Badger, D. Chicherin, T. Gehrmann, G. Heinrich, J. Henn, T. Peraro, P. Wasser, Y. Zhang, S. Zoia, Phys. Rev. Lett. 123 (7) (2019) 071601, https://doi.org/10.1103/PhysRevLett.123.071601, arXiv:1905.03733.
[19] S. Abreu, F. Febres Cordero, H. Ita, M. Jaquier, B. Page, M. Ruf, V. Sotnikov, Phys. Rev. Lett. 124 (21) (2020) 211601, https://doi.org/10.1103/PhysRevLett.124.211601, arXiv:2002.12374.
[20] H.A. Chawdhry, M.L. Czakon, A. Mitov, R. Poncelet, J. High Energy Phys. 02 (2020) 057, https://doi.org/10.1007/JHEP02(2020)057, arXiv:1911.00479.
[21] Z. Bern, L.J. Dixon, D.C. Dunbar, D.A. Kosower, Nucl. Phys. B 425 (1994) 217–260, https://doi.org/10.1016/0550-3213(94)90179-1, arXiv:hep-ph/9403226.
[22] Z. Bern, L.J. Dixon, D.C. Dunbar, D.A. Kosower, Nucl. Phys. B 435 (1995) 59–101, https://doi.org/10.1016/0550-3213(94)00488-Z, arXiv:hep-ph/9409265.
[23] Z. Bern, L.J. Dixon, D.A. Kosower, Nucl. Phys. B 513 (1998) 3–86, https://doi.org/10.1016/S0550-3213(97)00703-7, arXiv:hep-ph/9708239.
[24] R. Britto, F. Cachazo, B. Feng, Nucl. Phys. B 725 (2005) 275–305, https://doi.org/10.1016/j.nuclphysb.2005.07.014, arXiv:hep-th/0412103.
[25] H. Ita, Phys. Rev. D 94 (11) (2016) 116015, https://doi.org/10.1103/PhysRevD.94.116015, arXiv:1510.05626.
[26] S. Abreu, F. Febres Cordero, H. Ita, M. Jaquier, B. Page, Phys. Rev. D 95 (9) (2017) 096011, https://doi.org/10.1103/PhysRevD.95.096011, arXiv:1703.05255.

[27] S. Abreu, F. Febres Cordero, H. Ita, M. Jaquier, B. Page, M. Zeng, Phys. Rev. Lett. 119 (14) (2017) 142001, https://doi.org/10.1103/PhysRevLett.119.142001, arXiv:1703.05273.
[28] F.A. Berends, W.T. Giele, Nucl. Phys. B 306 (1988) 759–808, https://doi.org/10.1016/0550-3213(88)90442-7.
[29] P. Mastrolia, G. Ossola, J. High Energy Phys. 11 (2011) 014, https://doi.org/10.1007/JHEP11(2011)014, arXiv:1107.6041.
[30] S. Badger, H. Frellesvig, Y. Zhang, J. High Energy Phys. 04 (2012) 055, https://doi.org/10.1007/JHEP04(2012)055, arXiv:1202.2019.
[31] Y. Zhang, J. High Energy Phys. 09 (2012) 042, https://doi.org/10.1007/JHEP09(2012)042, arXiv:1205.5707.
[32] P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro, Phys. Lett. B 718 (2012) 173–177, https://doi.org/10.1016/j.physletb.2012.09.053, arXiv:1205.7087.
[33] K. Chetyrkin, F. Tkachov, Nucl. Phys. B 192 (1981) 159–204, https://doi.org/10.1016/0550-3213(81)90199-1.
[34] S. Laporta, Int. J. Mod. Phys. A 15 (2000) 5087–5159, https://doi.org/10.1016/S0217-751X(00)00215-7, arXiv:hep-ph/0102033, https://doi.org/10.1142/S0217751X00002157.
[35] C. Anastasiou, A. Lazopoulos, J. High Energy Phys. 07 (2004) 046, https://doi.org/10.1088/1126-6708/2004/07/046, arXiv:hep-ph/0404258.
[36] C. Studerus, Comput. Phys. Commun. 181 (2010) 1293–1300, https://doi.org/10.1016/j.cpc.2010.03.012, arXiv:0912.2546.
[37] A. von Manteuffel, C. Studerus, Reduze 2 - distributed feynman integral reduction, arXiv:1201.4330.
[38] R.N. Lee, J. Phys. Conf. Ser. 523 (2014) 012059, https://doi.org/10.1088/1742-6596/523/1/012059, arXiv:1310.1145.
[39] A. Smirnov, V. Smirnov, Comput. Phys. Commun. 184 (2013) 2820–2827, https://doi.org/10.1016/j.cpc.2013.06.016, arXiv:1302.5885.
[40] A.V. Smirnov, Comput. Phys. Commun. 189 (2015) 182–191, https://doi.org/10.1016/j.cpc.2014.11.024, arXiv:1408.2372.
[41] P. Maierhoefer, J. Usovitsch, P. Uwer, Comput. Phys. Commun. 230 (2018) 99–112, https://doi.org/10.1016/j.cpc.2018.04.012, arXiv:1705.05610.
[42] J. Gluza, K. Kajda, D.A. Kosower, Phys. Rev. D 83 (2011) 045012, https://doi.org/10.1103/PhysRevD.83.045012, arXiv:1009.0472.
[43] R.M. Schabinger, J. High Energy Phys. 01 (2012) 077, https://doi.org/10.1007/JHEP01(2012)077, arXiv:1111.4220.
[44] V. Sotnikov, Scattering amplitudes with the multi-loop numerical unitarity method, Ph.D. thesis, Freiburg U, 2019.
[45] A.B. Goncharov, M. Spradlin, C. Vergu, A. Volovich, Phys. Rev. Lett. 105 (2010) 151605, https://doi.org/10.1103/PhysRevLett.105.151605, arXiv:1006.5703.
[46] C. Duhr, J. High Energy Phys. 08 (2012) 043, https://doi.org/10.1007/JHEP08(2012)043, arXiv:1203.0454.
[47] T. Gehrmann, J.M. Henn, N.A. Lo Presti, J. High Energy Phys. 10 (2018) 103, https://doi.org/10.1007/JHEP10(2018)103, arXiv:1807.09812.
[48] R.K. Ellis, W.T. Giele, Z. Kunszt, K. Melnikov, Nucl. Phys. B 822 (2009) 270–282, https://doi.org/10.1016/j.nuclphysb.2009.07.023, arXiv:0806.3467.
[49] R. Boughezal, K. Melnikov, F. Petriello, Phys. Rev. D 84 (2011) 034044, https://doi.org/10.1103/PhysRevD.84.034044, arXiv:1106.5520.
[50] F.R. Anger, V. Sotnikov, On the dimensional regularization of QCD helicity amplitudes with quarks, arXiv:1803.11127.
[51] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables, vol. 55, Courier Corporation, 1964.
[52] J. Klappert, F. Lange, Reconstructing rational functions with `FireFly`, arXiv:1904.00009.
[53] J. Klappert, S.Y. Klein, F. Lange, Interpolation of dense and sparse rational functions and other improvements in `FireFly`, arXiv:2004.01463.
[54] T. Peraro, J. High Energy Phys. 07 (2019) 031, https://doi.org/10.1007/JHEP07(2019)031, arXiv:1905.08019.
[55] Y. Hida, S. Li, D. Bailey, Quad-double arithmetic: algorithms, implementation, and application.
[56] T. Granlund, the GMP development team, GNU MP: The GNU Multiple Precision Arithmetic Library, 5th edition, 2012, http://gmplib.org/.
[57] P. Barrett, in: A.M. Odlyzko (Ed.), Advances in Cryptology — CRYPTO' 86, Springer, Berlin, Heidelberg, 1987, pp. 311–323.
[58] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[59] Eigen, http://eigen.tuxfamily.org/index.php?title=Main_Page.
[60] W.R. Inc, Mathematica, Version 12.1, champaign, IL, 2020, https://www.wolfram.com/mathematica.
[61] C.W. Bauer, A. Frink, R. Kreckel, J. Symb. Comput. 33 (2000) 1, arXiv:cs/0004015.
[62] J. Vollinga, S. Weinzierl, Comput. Phys. Commun. 167 (2005) 177, https://doi.org/10.1016/j.cpc.2004.12.009, arXiv:hep-ph/0410259.
[63] J. Vollinga, Nucl. Instrum. Methods A 559 (2006) 282–284, https://doi.org/10.1016/j.nima.2005.11.155, arXiv:hep-ph/0510057.
[64] B. Haible, CLN - class library for numbers, https://www.ginac.de/CLN/.
[65] F.R. Anger, F. Febres Cordero, H. Ita, V. Sotnikov, Phys. Rev. D 97 (3) (2018) 036018, https://doi.org/10.1103/PhysRevD.97.036018, arXiv:1712.05721.
[66] A. Ochirov, B. Page, J. High Energy Phys. 02 (2017) 100, https://doi.org/10.1007/JHEP02(2017)100, arXiv:1612.04366.
[67] A. Ochirov, B. Page, J. High Energy Phys. 10 (2019) 058, https://doi.org/10.1007/JHEP10(2019)058, arXiv:1908.02695.
[68] A. Hodges, J. High Energy Phys. 05 (2013) 135, https://doi.org/10.1007/JHEP05(2013)135, arXiv:0905.1473.
[69] Meson, https://mesonbuild.com/.
[70] C. Cheung, G.N. Remmen, J. High Energy Phys. 09 (2017) 002, https://doi.org/10.1007/JHEP09(2017)002, arXiv:1705.00626.
[71] D. Chicherin, V. Sotnikov, Pentagon functions for scattering of five massless particles, arXiv:2009.07803.
[72] Doxygen, http://www.doxygen.nl/.
[73] Z. Bern, L.J. Dixon, D.A. Kosower, S. Weinzierl, Nucl. Phys. B 489 (1997) 3–23, https://doi.org/10.1016/S0550-3213(96)00703-1, arXiv:hep-ph/9610370.
[74] E. Glover, J. High Energy Phys. 04 (2004) 021, https://doi.org/10.1088/1126-6708/2004/04/021, arXiv:hep-ph/0401119.
[75] A. De Freitas, Z. Bern, J. High Energy Phys. 09 (2004) 039, https://doi.org/10.1088/1126-6708/2004/09/039, arXiv:hep-ph/0409007.
[76] L.J. Dixon, in: Theoretical Advanced Study Institute in Elementary Particle Physics (TASI 95): QCD and Beyond, 1996, pp. 539–584, arXiv:hep-ph/9601359.
[77] V. Del Duca, L.J. Dixon, F. Maltoni, Nucl. Phys. B 571 (2000) 51–70, https://doi.org/10.1016/S0550-3213(99)00809-3, arXiv:hep-ph/9910563.
[78] H. Johansson, A. Ochirov, J. High Energy Phys. 01 (2016) 170, https://doi.org/10.1007/JHEP01(2016)170, arXiv:1507.00332.
[79] S. Badger, J. Phys. Conf. Ser. 762 (1) (2016) 012057, https://doi.org/10.1088/1742-6596/762/1/012057, arXiv:1605.02172.
[80] S. Badger, H. Frellesvig, Y. Zhang, J. High Energy Phys. 12 (2013) 045, https://doi.org/10.1007/JHEP12(2013)045, arXiv:1310.1051.