UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea Magistrale in Fisica

Tesi di Laurea

The natural structure of scattering amplitudes

Relatore

Prof. Pierpaolo Mastrolia

Correlatore

Dr. Tiziano Peraro

Laureando

Manuel Accettulli Huber

Anno Accademico 2017/2018

Contents

Abstract

In this thesis we present all the techniques needed for the functional reconstruction of univariate rational functions over finite fields and apply them to tree-level scattering amplitudes. In particular the reconstruction of scattering amplitudes on finite fields using the BCFW recursion as black-box algorithm is presented here for the first time. Furthermore, as a second novel result, we discuss some new aspects of the relation among BCFW tree-level recursion and the maximum-cut of multi-loop diagrams.

Given a black-box algorithm which computes numerical values of a rational function at given values of the variable, the reconstruction procedure allows to obtain the associated analytical expression. We begin by describing the reconstruction on the field $\mathbb{Q}$ of univariate polynomials, via Newton's polynomial representation, and of univariate rational functions, via Thiele's interpolation formula. Then we define finite fields $\mathbb{Z}_p$ along with the addition and multiplication operations characterizing them and the map $\mathbb{Q} \to \mathbb{Z}_p$, discussing the hypothesis under which the latter map can be inverted. We describe how the reconstruction procedure can be performed on $\mathbb{Z}_p$, and discuss how the map $\mathbb{Z}_p \to \mathbb{Q}$ can be complemented by the use of the Chinese remainder theorem, allowing for numerical calculations to be always performed in the domain of machine-size integers.

The application of rational reconstruction to tree-level scattering amplitudes is then presented, using as black-box algorithm the Berends-Giele and BCFW recursions, which are both described in detail. While the second is the main focus of our work, the first is used as a comparison and for cross-checks. We introduce the momentum-twistor parametrization for the amplitude, describing also how the latter can be factorized and how the recurrence relations adapted in order for rational reconstruction to be applicable. An example reconstruction of a four-point amplitude using BCFW recursion is then given.

Finally we explore some of the relations among BCFW tree-level recurrence and multi-loop maximum-cut graphs. In particular we present a novel strategy allowing the direct construction of all the multi-loop graphs related to BCFW recursion, starting only from the number of external legs.

Introduction

Among the many fascinating mysteries of nature, the microscopic structure of matter is one of the most intriguing ones. More than one hundred years after Rutherford's "gold foil experiment", which led to the discovery of the atomic nucleus and laid the foundations for the modern atomic model, the investigation techniques are based on the same principle: particle scattering. However what has changed are the length and energy scales of the processes physicists are interested in, the first extremely small and the second extremely large.

At the Large Hadron Collider (LHC) high precision experiments based on colliding hadrons are performed, in order to investigate the nature of subnuclear constituents of those which once were thought to be the most fundamental building blocks of matter. High precision experiments require high precision theoretical predictions, which will either be confirmed or rejected shaping as a consequence the mathematical model of nature we accept as true. The theoretical framework for these predictions is Quantum Field Theory (QFT), which combines the founding principles of Quantum Mechanics and Special Relativity, allowing to describe objects as small as subatomic particles moving almost at speed of light. In QFT the probability of a certain particle collision to take place is described by the scattering amplitude, an analytical function of the momenta of the involved particles. A part from special cases, the exact expression of a scattering amplitude cannot be found, thus it is expanded in terms of a small perturbative parameter and then computed through successive approximation until the required precision of the result is achieved.

Even when the determination of the leading order contribution is simple, the calculation of higher order corrections becomes soon more and more involved, its complexity growing exponentially with the number of particles involved and the order of the correction one wants to compute. This is due to the fact that many different but quantum-mechanically indistinguishable processes contribute to each scattering amplitude, and at each order of the perturbative expansion many new such processes appear involving not only the real but virtual particles as well. Since the latter are not bound to have physical mass or specific momenta one needs to sum over all the infinitely many possibilities, i.e. integrate over what is called loop-momentum.

These integrals are often highly non-trivial, and powerful techniques have been developed to deal with them. Among these, striking results were obtained by expressing the amplitude in terms of a universal integral basis, whose elements are called Master Integrals (MIs). However, writing scattering amplitudes as linear combinations of integrals is currently one of the main bottlenecks of high-multiplicity multi-loop calculations.

An issue which is common to analytic calculations in high-energy physics is the large size of intermediate expressions, which can often be orders of magnitude larger than the final result. The latter in fact often enjoys properties, in particular satisfies certain symmetries, which are not shared by the intermediate step of the computation. Moreover these are in general described also by a larger number of variables (such as loop momenta) which do not appear in the result. The problem can be mitigated through the use of unitarity based methods [7], which involve gauge-invariant intermediate expressions which are thus simpler.

Unitarity as well as analiticity are qualitative properties of scattering amplitudes which, when combined with complex momenta for propagating particles, can be turned into powerful tools for computing them. Analiticity yields the possibility of identifying in each amplitude sub-reactions that correspond to simpler collisions, and unitarity, the probability conservation across each of these sub-processes. In particular, for diagrams containing just one loop, these ideas proved to be very effective.

It is in this framework that the so called on-shell methods originated. They owe their name to the fact that the appearance of unphysical degrees of freedom, whose effects are not observable in the final physical result, are avoided by restricting the propagating states to the physical ones, i.e. particles which are on mass-shell. Diagrams which present common kinematic structures, as for example multi-particle poles at tree-level or branch-cuts at loop-level, can be treated simultaneously using on-shell techniques. These allow the extraction of analytic information from simpler amplitudes in a recursive fashion, since the singularity structure defining a scattering amplitude is determined by lower-point on-shell amplitudes, in the case of poles, and by lower-loop ones, in the case of cuts. The price to pay for intermediate states of tree-level amplitudes to go on-shell, is the introduction of suitable external complex momenta. More precisely, starting from a set of real momenta, associated to a given tree-level scattering amplitude, a corresponding amplitude can be defined where some of the external momenta get an appropriate complex component injected in their spinorial representation. The use of complex momenta combined with the residue theorem, allows to determine the tree-level amplitude from its singularity structure, which in turn leads to a factorization into lower-point on-shell amplitudes corresponding to simpler processes with complex external momenta. Each sub-process can then again be constructed from simpler amplitudes, all the way down until only three-point functions remain. Noticeably these three-point amplitudes are allowed to be non-vanishing only because complex kinematics is involved. In other words glueing together amplitudes corresponding to simpler processes, connected by on-shell, complex propagating particles, allows to reconstruct the given parent amplitude. The recursive application of this technique is known as BFCW recurrence relation [9].

Complex kinematics is also useful for the fulfilment of generalized unitarity conditions. Generalized unitarity is an extension of the idea behind the optical theorem, and at one-loop corresponds to requiring more than two internal particles to be on-shell. In general such constraints cannot be realized with real Minkowski momenta. The application of unitarity as an on-shell method of calculation is based on two principles [6, 7]:

1. Sewing tree amplitudes together to form one-loop amplitudes. The products of tree-level amplitudes produces functions with the correct branch cuts in all channels. Using (on-shell) amplitudes inside the cuts, instead of cutting Feynman diagrams, offers the advantage of working with simplified expressions which contain only physical degrees of freedom, resulting in compact and clean expressions.

2. Decomposing loop-amplitudes in terms of a basis of loop-integrals (Master Integrals). Matching the generalized cuts of the amplitude with the cuts of the basic integrals provides an efficient way to extract the coefficients of the decomposition in terms of MI.

On the more mathematical side, in the framework of on-shell and unitarity-based methods, the theory of multivariate complex fuctions can be seen to play an important role in order to compute the generalized cuts efficiently.

Combining unitarity and analyticity with the idea of reduction under the integral sign, integrand-decomposition methods are born. These allow the extractions of the coefficients of the MI by exploiting the multi-particle pole expansion for the integrand of the scattering amplitude, i.e. a representation where the numerator of each Feynman integral is expressed as a combination of products of the corresponding denominators, with polynomial coefficients. The crucial aspect is the shape of the residue on the multi-particle pole. Each residue is a multivariate polynomial in the irreducible scalar products (ISPs) of loop momenta and either external momenta or polarization vectors. In the residues reducible scalar products, i.e. products which can be written in terms of denominators, are not allowed to appear: else further simplification would be possible, and this contradicts the definition of residue. In the context of an integrand-reduction, any explicit integration procedure and/or any matching procedure between cuts of amplitudes and cuts of MIs is replaced by polynomial fitting, which is a simpler operation. The parametric form of the polynomial residues is process-independent and can be determined once and for all from the structure of the corresponding multiple cut. The actual value of the coefficients which appear in the residues is instead process-dependent and, in the framework of the integrand-reduction their determination is achieved by sampling the known integrand at values of the loop-momenta fulfilling the cut conditions. Decomposing the amplitudes in terms of MIs amounts to reconstructing the full polynomiality of the residues, i.e. it amounts to determining all the coefficients of each polynomial [18, 23].

In [21], it is shown that the shape of the residues is uniquely determined by the on-shell conditions alone, without any additional constraint. A simple integrand recurrence relation can then be derived, which generates the required multi-particle pole decomposition for arbitrary amplitudes, independently of the number of loops. The algorithm treats the numerator and the denominators of any Feynman integrand, as multivariate polynomials in the components of the loop variables. The method uses both the weak Nullstellensatz theorem and the multivariate polynomial division modulo Gröbner basis, two powerful tools of algebraic geometry, the first allowing a simple formulation of a reducibility criterion of the integrand, and the second granting that the residues of polynomial division are uniquely defined. More recently, an improved integrand decomposition algorithm has been proposed, called Adaptive Integrand Decomposition [19]. This exploits the simplifications arising when splitting the space-time dimensions of dimensionally regulated Feynman integrals into parallel and orthogonal dimensions. The splitting is performed basing on the number of legs of the individual diagram, hence the name Adaptive Integrand Decomposition.

Besides the so far addressed techniques, in [25] a novel approach to high-energy physics calculations is presented, which allows to side-step the issue of large intermediate expressions , by reconstructing analytic results from their numerical evaluation, where each intermediate step is a relatively small number (or a set of numbers). This is done by the use of finite fields, a tool quite commonly adopted in computer algebra, but whose application to a realistic problem in high energy physics involving the reconstruction of multivariate rational functions, was presented in [25] for the first time. Any algorithm which can be implemented via a sequence of rational elementary operations is suited for the use of these reconstruction techniques. Multi-loop integrand reduction via generalized unitarity falls into this category, but we will focus on its application to tree-level amplitudes, which can be shown to be rational functions when expressed in terms of so called momentum-twistor variables [3, 14, 32].

The functional reconstruction of an amplitude requires many numerical evaluations, which in principle could be performed in various ways. The most obvious choice would be a floating-point calculation, however this would be affected by numerical inaccuracies. To perform exact computations rational numbers should be used, however by doing so the typically large (analytic) intermediate step expressions mentioned above traduce into rationals whose numerator and denominator are generally described by a large number of digits. This requires extensive use of arbitrary-precision arithmetic, which is computationally expensive. What one can do is to perform the numerical evaluation over a finite field called $\mathbb{Z}_p$, i.e. a field composed of a finite number of elements represented by consecutive integers [16]. This allows for exact computations but at the same time, keeping the order of the field under machine-size, no arbitrary-precision arithmetic is needed.

Once the functional reconstruction over finite fields was performed, the resulting analytical expression on $\mathbb{Z}_p$ needs to be mapped back to the field $\mathbb{Q}$. Clearly the mapping among these fields cannot be bijective, since $\mathbb{Z}_p$ has only a finite number of elements whereas $\mathbb{Q}$ has infinitely many. Nevertheless under appropriate hypothesis [30] a unique rational corresponding to the reconstructed result can be identified.

The first aim of this thesis is the presentation of all the tools and methods needed for the univariate rational reconstruction over finite fields, the transposition of the results to $\mathbb{Q}$, and its application to tree-level scattering amplitudes. In particular our goal is to perform a complete reconstruction of a tree-level scattering amplitude over finite fields using as black-box algorithm the BCFW recursion, which will be one of the novel results of this thesis.

The BCFW recursion is particularly interesting because it presents a link to the leading singularity of appropriate one-loop amplitudes [7]. What is still to be explored is the link it may have with the residue of the maximum-cut graphs appearing in integrand reduction methods [21]. As has been shown [21] this residue can be expressed as an univariate polynomial of degree $n_s - 1$, where n_s is the number of solutions of the system defined by the cut conditions. Considering the similarities among the analytic structure of BCFW tree-level recursion and multi-loop maximum-cut graphs, the result of the recurrence may present some form of connection (yet to be uncovered) in particular with the constant term of this polynomial. In the last part of this thesis we started to explore this connection, obtaining as a second main result a general strategy allowing the direct construction of the maximum-cut graphs related to the BCFW recursion.

This thesis is organised as follows. First of all a brief review of computation techniques for scattering amplitudes is given in chapter 1. In particular the so called spinor-helicity formalism is introduced [13, 27] which allows to write amplitudes in terms of left- and right-handed Weyl spinors. Then color-ordered amplitudes are presented [13], gauge invariant objects from which the color structure has been factorized allowing for simpler computations. From there on we will always consider color-ordered amplitudes only. Among these a special class is represented by the Maximum Helicity Violating (MHV) amplitudes [24, 28], whose analytic expressions in the so far presented formalism are simple and known for an arbitrary number of legs. Then we move on to discuss rational parametrizations of the amplitude [1–3, 26], without which rational reconstruction methods could not be applied. In the last section of the chapter the Berends-Giele [4] and BCFW [9] tree-level recurrence relations are treated. These will provide the black-box algorithm for the successive rational reconstruction procedure. While the second relation is the main focus of the thesis, the first is used for comparison and cross-checks.

In chapter 2 the mathematical tools needed for rational reconstruction over finite fields are introduced. First we review functional reconstruction algorithms for univariate polynomials and rational functions [12, 25]. These algorithms allow to reconstruct the analytic expression of functions from their repeated numerical evaluation, and they are independent of the specific algorithm used for the evaluation. More specifically we will be using Newton's polynomial representation for the reconstruction of univariate polynomials and Thiele's interpolation formula for univariate rational functions. Then we will briefly outline the reconstruction procedure for multivariate polynomials and rational functions [25], whose implementation was however beyond the scope of this thesis. In the second half of the chapter finite fields will be defined [16] and the mapping from $\mathbb{Q}$ to them will be described. Operations over finite fields are introduced immediately after the mapping, and finally the so called Chinese remainder theorem (e.g. [25]) is reported along with its proof.

In chapter 3 we start discussing how the so far presented theoretical machinery is put to work. First our code implementation of the reconstruction of univariate polynomials and rational functions on the field $\mathbb{Q}$ is described in detail, followed by the description of the analogous procedure over a finite field. This is then complemented by the description of how the Chinese remainder theorem can be used to effectively convert multiple reconstructions over different finite fields into the sought analytic expression over $\mathbb{Q}$ of the unknown function.

The goal of chapter 4 is the application of rational reconstruction on finite fields to a tree-level scattering amplitude computed via BCFW recursion, which represents the first main result of the thesis. We describe how the Berends-Giele and BCFW tree-level recurrences can be adapted so that only rational numbers appear in the intermediate step expressions as well as in the output, thus becoming rational functions. At this point numerical evaluation of scattering amplitudes over finite fields is discussed, i.e. the transposition of Berends-Giele and BCFW recursions over finite fields. The chapter ends with a description of how all the presented tools are put together to achieve the reconstruction of the analytic expression of a tree-level color-ordered scattering amplitude. A four-point example is then presented.

In chapter 5 we present a novel idea about the relation between on-shell recurrence relations and maximum-cuts of multi-loop amplitudes [21], which constitute the second main result of this work. First analogies among the analytical structure of the BCFW recursion and the quadruple-cut of a one-loop amplitude are explored. These analogy is then converted into a new diagrammatic representation of the recursion relation. Finally, after some explicit examples are presented, we outline a general strategy which allows the direct determination of the multi-loop diagrams whose poles contribute to the BCFW tree-level recursion.

Chapter 1

Scattering amplitudes in gauge theories

In this chapter we are going to review some powerful methods for calculations of scattering amplitudes in QCD. The first topic we are going to discuss is the so called spinor-helicity formalism which allows to represent scattering amplitudes in terms of the spinor components instead of mandelstam invariants or scalar products of the momenta of the external particles. Then follows a short section on color-ordered and MHV amplitudes. These techniques are well known and an exhaustive introduction to them can be found for example in [27] and [13].

Finally we will be discussing momentum-twistor variables, a particular parametrization in which the amplitude is a rational function and thus suitable for the functional reconstruction algorithms presented in chapter 2.

1.1 Spinor-helicity formalism

Perturbative QCD is primarily concerned with the interactions of quarks and gluons at momentum scales for which the masses of these particles can be ignored. Considering massless states we have that the helicity of the particles becomes a definite Lorentz-invariant quantity, and is thus an intrinsic property of the particles themselves. What we are going to compute are scattering amplitudes in the massless limit and thus in terms of definite helicity states.

1.1.1 General idea

Recall first of all that given a spin-$\frac{1}{2}$ particle described by the field

$$\psi(x) = \int \frac{d^3p}{(2\pi)^3} \frac{1}{\sqrt{2E_{\mathbf{p}}}} \sum_{spin=s} \left(a^s_{\mathbf{p}} U^s(p) e^{-ip\cdot x} + b^s_{\mathbf{p}} V^s(p) e^{ip\cdot x}\right) \tag{1.1}$$

the massless Dirac equation in momentum space reads

$$i\partial\!\!\!/\psi(x) = 0 \Rightarrow \begin{cases} p\!\!\!/ U^s(p) = 0 & (1.2) \\ p\!\!\!/ V^s(p) = 0 & (1.3) \end{cases}$$

and for the conjugate field

$$i\partial_\mu \bar{\psi}(x)\gamma^\mu = 0 \Rightarrow \begin{cases} \overline{U}^s(p) p\!\!\!/ = 0 & (1.4) \\ \overline{V}^s(p) p\!\!\!/ = 0 & (1.5) \end{cases}$$

Now we are going to choose an explicit representation for the gamma matrices. Since we are dealing with massless particles, for which spin and helicity (and chirality) eigenstates coincide, a particularly well suited choice is the Weyl representation in which

$$\gamma^\mu = \begin{pmatrix} 0 & \sigma^\mu \\ \bar{\sigma}^\mu & 0 \end{pmatrix} \qquad \gamma^5 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.6}$$

where $\sigma^\mu = (\mathbb{1}_2, \vec{\sigma})$, $\bar{\sigma}^\mu = (\mathbb{1}_2, -\vec{\sigma})$ and the solutions to the above equations, which can be shown to be unique, appear in a particularly simple form. In this representation the chirality projectors

$$P_R = \frac{\mathbb{1} + \gamma^5}{2} \qquad P_L = \frac{\mathbb{1} - \gamma^5}{2} \tag{1.7}$$

project respectively onto the first two and the second two of the four components of the Dirac spinors, and since in the considered limit chirality and helicity coincide we get a simple form for the helicity eigenstates. Labeling the different eigenstates in terms of helicity left and right we will call the solutions to eq. (1.2):

$$U_R(p) = \begin{pmatrix} 0 \\ u_R(p) \end{pmatrix} \qquad U_L(p) = \begin{pmatrix} u_L(p) \\ 0 \end{pmatrix} \tag{1.8}$$

From here on we will use the upper vs lower case notation to distinguish the four-component Dirac spinors, e.g. $U_L(p)$, from the two-component Weyl spinors, e.g. $u_L(p)$. In terms of u_L and u_R eq. (1.2) reads

$$\begin{cases} p \cdot \sigma u_R = 0 & (1.9) \\ p \cdot \bar{\sigma} u_L = 0 & (1.10) \end{cases}$$

Notice that there is a relation between the two solutions u_L and u_R, in fact, if eq. (1.10) is satisfied

$$p_\mu \bar{\sigma}^\mu u_L(p) = 0 \tag{1.11a}$$
$$i\sigma^2 (p_0 \mathbb{1}_2 - p_1\sigma^1 - p_2\sigma^2 - p_3\sigma^3) u_L(p) = 0 \tag{1.11b}$$
$$(-i(-\sigma^2))(p_0 \mathbb{1}_2 - p_1\sigma^1 - p_2(-\sigma^2) - p_3\sigma^3) u_L^*(p) = 0 \tag{1.11c}$$
$$i(p_0 \mathbb{1}_2\sigma^2 + p_1\sigma^1\sigma^2 + p_2\sigma^2\sigma^2 + p_3\sigma^3\sigma^2) u_L^*(p) = 0 \tag{1.11d}$$
$$p_\mu \sigma^\mu (i\sigma^2 u_L^*(p)) = 0 \tag{1.11e}$$

where we multiplied eq. (1.11a) by $i\sigma^2$ and expanded the contracted μ index, then we took the complex conjugate of both members of eq. (1.11b) , and using the anticommutation relations $\{\sigma^i, \sigma^j\} = 2\delta^{ij}\mathbb{1}_2$ we get eq. (1.11e). From uniqueness of the solution to eq. (1.9) we have that

$$u_R(p) = i\sigma^2 u_L^*(p) \tag{1.12}$$

which can easily be recognized as the charge conjugation relation.

In order to describe antiparticles the same reasoning applies to eq. (1.3) leading to a general solution for $V(p)$. However, considering the massless case the equation satisfied by $V(p)$ is identical to that satisfied by $U(p)$, so we already now the two unique solutions: $u_L(p)$ and $u_R(p)$. In any case one has to be careful because for antiparticles in the massless case chirality left coincides with helicity right and vice versa, so we have

$$V_L(p) = \begin{pmatrix} 0 \\ u_R(p) \end{pmatrix} = U_R(p) \qquad V_R(p) = \begin{pmatrix} u_L(p) \\ 0 \end{pmatrix} = U_L(p) \tag{1.13}$$

Keeping this in mind all the building blocks we need to describe a scattering amplitude with only external fermions are U_L,U_R, $\overline{U}_L$ and $\overline{U}_R$ for which we will use the compact notation

$$U_R(p) = |p\rangle\ , \quad U_L(p) = |p]\ , \qquad \overline{U}_L(p) = \langle p|\ , \quad \overline{U}_R(p) = [p| \tag{1.14}$$

Given these the only non-vanishing Lorentz-invariant spinor products are those of the form

$$\overline{U}_L(p)U_R(q) = \langle p|q\rangle\ , \quad \overline{U}_R(p)U_L(q) = [p|q] \tag{1.15}$$

which will be simply called angle brackets and square brackets respectively. The scattering amplitudes will be products of these quantities.

Before proceeding further there is another simplification that can be made. It consists in considering all the external particles as outgoing (or incoming), and then in order to recover the physical amplitude flip the sign of the time component of the incoming (outgoing) particles. Moreover the three-momentum we assign to these particles for the computation needs to be opposite to their physical three-momentum, thus leading the physical helicity to be opposite to the assigned one[1].

From here on we are going to consider all the external particles as outgoing.

1.1.2 Useful identities

In this section we are going to present some identities which will allow us to reduce complex spinor expressions to functions of angle brackets and square brackets.

Since it will prove useful in the following we are going to rewrite eq. (1.15), expanding the product on the left hand side, as

$$\langle p|q\rangle = u_L^\dagger(p)u_R(q) \qquad [p|q] = u_R^\dagger(p)u_L(q) \tag{1.16}$$

First of all notice that the angle and square brackets are antisymmetric

$$\begin{aligned}
\langle p|q\rangle &= u_L^\dagger(p)u_R(q) && \text{(1.17a)}\\
&= u_R^T(q)u_L^*(p) && \text{(1.17b)}\\
&= (i\sigma^2 u_L^*(q))^T(-i\sigma^2 u_R(p)) && \text{(1.17c)}\\
&= u_L^\dagger(q)(-\sigma^2\sigma^2)u_R(p) && \text{(1.17d)}\\
&= -\langle q|p\rangle && \text{(1.17e)}
\end{aligned}$$

where we used the fact that $\langle p|q\rangle$ being a number is equal to its own transpose and then eq. (1.12). Similarly one gets the analogous result for the square brackets.

$$\langle p|q\rangle = -\langle q|p\rangle \qquad [p|q] = -[q|p] \tag{1.18}$$

Moreover there is a relation between $\langle p|q\rangle$ and $[p|q]$:

$$\begin{aligned}
[p|q]^* &= (u_R^\dagger(p)u_L(q))^* && \text{(1.19a)}\\
&= u_R^T(p)u_L^*(q) && \text{(1.19b)}\\
&= u_L^\dagger(q)u_R(p) && \text{(1.19c)}\\
&= \langle q|p\rangle && \text{(1.19d)}
\end{aligned}$$

[1]For a more technical though brief discussion of this aspect see for example [29] page 365.

having again used the fact that $[p|q]$ is a number and so invariant under transposition, leading to

$$[p|q]^* = \langle q|p\rangle \tag{1.20}$$

Recalling now that

$$U(p)\overline{U}(p) = \not{p} \tag{1.21}$$

we can easily see that

$$|p\rangle[p| = \not{p}\left(\frac{\mathbb{1}_2 - \gamma^5}{2}\right) \qquad |p]\langle p| = \not{p}\left(\frac{\mathbb{1}_2 + \gamma^5}{2}\right) \tag{1.22}$$

Take for example the first relation, using $\{\gamma^\mu, \gamma^5\} = 0$ and the idempotence of the chirality projectors we have

$$\begin{aligned}
|p\rangle[p| &= U_R(p)\overline{U}_R(p) && (1.23a)\\
&= P_R U(p)\overline{U}(p)P_L && (1.23b)\\
&= \not{p}(P_L)^2 && (1.23c)\\
&= \not{p}\left(\frac{\mathbb{1}_2 - \gamma^5}{2}\right) && (1.23d)
\end{aligned}$$

As an immediate consequence one gets

$$\not{p} = |p\rangle[p| + |p]\langle p| \tag{1.24}$$

Instead if we want to express p^μ in terms of the angle and square brackets:

$$\begin{aligned}
\langle p|\gamma^\mu|p] &= \overline{U}_L(p)\gamma^\mu U_L(p) && (1.25a)\\
&= (\overline{U}(p)P_R)_i\gamma^\mu_{ij}(P_L U(p))_j && (1.25b)\\
&= \overline{U}(p)_i(P_R\gamma^\mu P_L)_{ij}U(p)_j && (1.25c)\\
&= \not{p}_{ji}(\gamma^\mu(P_L)^2)_{ij} && (1.25d)\\
&= p_\nu Tr\left[\gamma^\nu\gamma^\mu\left(\frac{\mathbb{1} - \gamma^5}{2}\right)\right] && (1.25e)\\
&= p_\nu Tr[\gamma^\nu\gamma^\mu] && (1.25f)\\
&= 2p^\mu && (1.25g)
\end{aligned}$$

with i and j spinor indexes and having used again the anti-commutation of γ^5 with γ^μ and the idempotence of the projectors. So

$$p^\mu = \frac{1}{2}\langle p|\gamma^\mu|p] \tag{1.26}$$

This expression for p^μ can be cast into a different form using another identity:

$$\begin{aligned}
\langle p|\gamma^\mu|q] &= u_L^\dagger(p)\bar{\sigma}^\mu u_L(q) && (1.27a)\\
&= u_L^\dagger(p)(-(i\sigma^2)^2)\bar{\sigma}^\mu u_L(q) && (1.27b)\\
&= u_L^\dagger(p)(-i\sigma^2)(\sigma^\mu)^T(i\sigma^2)u_L(p) && (1.27c)\\
&= u_R^T(p)(\sigma^\mu)^T u_R^*(q) && (1.27d)\\
&= u_R^\dagger(q)\sigma^\mu u_R(p) && (1.27e)\\
&= [q|\gamma^\mu|p\rangle && (1.27f)
\end{aligned}$$

where we used the relation $\sigma^2\bar{\sigma}^\mu = (\sigma^\mu)^T\sigma^2$ and then simply rearranged terms. So

$$\langle p|\gamma^\mu|q] = [q|\gamma^\mu|p\rangle \tag{1.28}$$

leading to the alternative expression for the four momentum

$$p^\mu = \frac{1}{2}[q|\gamma^\mu|p\rangle \tag{1.29}$$

Equation (1.28) can be generalized, for an odd number of γ matrices

$$\langle p|\gamma^{\mu_1}\gamma^{\mu_2}\cdots\gamma^{\mu_{2n}}\gamma^{\mu_{2n+1}}|q] = [q|\gamma^{\mu_{2n+1}}\gamma^{\mu_{2n}}\cdots\gamma^{\mu_2}\gamma^{\mu_1}|p\rangle \tag{1.30}$$

and

$$\langle p|\gamma^{\mu_1}\cdots\gamma^{\mu_{2n+1}}|q\rangle = 0 \tag{1.31}$$

$$[p|\gamma^{\mu_1}\cdots\gamma^{\mu_{2n+1}}|q] = 0 \tag{1.32}$$

whereas, for an even number of gamma matrices, one has

$$\langle p|\gamma^{\mu_1}\cdots\gamma^{\mu_{2n}}|q\rangle = -\langle q|\gamma^{\mu_{2n}}\cdots\gamma^{\mu_1}|p\rangle \tag{1.33}$$

$$[p|\gamma^{\mu_1}\cdots\gamma^{\mu_{2n}}|q] = -[q|\gamma^{\mu_{2n}}\cdots\gamma^{\mu_1}|p] \tag{1.34}$$

From eq. (1.22) we get, by tacking the trace of $|p\rangle[p|$ multiplied with $|q]\langle q|$

$$\langle p|q\rangle[q|p] = Tr\left[\not{q}\not{p}\left(\frac{\mathbb{1}+\gamma^5}{2}\right)\right] = 2p\cdot q \tag{1.35}$$

and considering eq. (1.20) we have

$$|\langle p|q\rangle|^2 = |[q|p]|^2 = 2p\cdot q \tag{1.36}$$

and defining the invariant s_{ij}

$$s_{ij} \equiv (p_i+p_j)^2 = 2p_i\cdot p_j = \langle i|j\rangle[j|i] \tag{1.37}$$

Another very useful identity is called Fierz rearrangement, basing on the identity of sigma matrices

$$(\bar{\sigma}^\mu)_{ab}(\bar{\sigma}_\mu)_{cd} = 2(i\sigma^2)_{ac}(i\sigma^2)_{bd} \tag{1.38}$$

one has

$$\langle p|\gamma^\mu|q]\langle k|\gamma_\mu|l] = \overline{U}_L(p)\gamma^\mu U_L(q)\overline{U}_L(k)\gamma_\mu U_L(l) \tag{1.39a}$$
$$= u_L^\dagger(p)_a(\bar{\sigma}^\mu)_{ab}u_L(q)_b u_L^\dagger(k)_c(\bar{\sigma}_\mu)_{cd}u_L(l)_d \tag{1.39b}$$
$$= u_L^\dagger(p)_a u_L(q)_b u_L^\dagger(k)_c u_L(l)_d 2(i\sigma^2)_{ac}(i\sigma^2)_{bd} \tag{1.39c}$$
$$= 2u_L^\dagger(p)_a i\sigma^2_{ac}u_L^*(k)_c u_L^T(q)_b i\sigma^2_{bd}u_L(l)_d \tag{1.39d}$$
$$= -2u_L^\dagger(p)_a u_R(k)_a u_R^\dagger(q)_b u_L(l)_b \tag{1.39e}$$
$$= -2\overline{U}_L(p)U_R(k)\overline{U}_R(q)U_L(l) \tag{1.39f}$$
$$= 2\langle p|k\rangle[l|q] \tag{1.39g}$$

where we used eq. (1.38), eq. (1.12) and eq. (1.12), the latter in the rearranged form

$$u_L^T(q)i\sigma^2 = (-i\sigma^2 u_L(q))^T = (-i\sigma^2 u_L^*(q))^\dagger = -u_R^\dagger(q) \tag{1.40}$$

Using eq. (1.28) we can cast it also in a different form, so the Fierz identity reads

$$\langle p|\gamma^\mu|q]\langle k|\gamma_\mu|l] = 2\langle p|k\rangle[l|q] \qquad \langle p|\gamma^\mu|q][k|\gamma_\mu|l\rangle = 2\langle p|l\rangle[k|q] \tag{1.41}$$

Finally, spinor products obey the Schouten identity

$$\langle i|j\rangle\langle k|l\rangle + \langle i|k\rangle\langle l|j\rangle + \langle i|l\rangle\langle j|k\rangle = 0 \tag{1.42}$$

$$[i|j][k|l] + [i|k][l|j] + [i|l][j|k] = 0 \tag{1.43}$$

this can be shown noticing that the expressions on the left are totally antisymmetric in j, k, l, but totally antisymmetrizing three two-component objects gives zero.

1.1.3 Polarization vectors

Until now we only focused on spin-$\frac{1}{2}$ particles, i.e. quarks, but in order to describe QCD using spinor-helicity formalism we also need an expression for the polarization vectors of the gluons.

First of all, one may wonder if it is even possible to write the polarization vectors in terms of angle and square brackets. The answer is yes, the proof of this statement comes from group theory: $|p\rangle$ and $|p]$ correspond to elements respectively of the $(\frac{1}{2}$,0) and (0,$\frac{1}{2})$ representation of the proper Lorentz group. It can be shown that all other finite dimensional representations of the Lorentz group can be obtained from direct products of these two "fundamental" representations, e.g. four vectors transform in the $(\frac{1}{2}, \frac{1}{2})$ representation. We already implicitly used this fact when writing the 4-momentum p^μ as eq. (1.26), so we expect the polarization vectors to have a similar form, and in fact they can be written as[2]

$$\epsilon_R^{*\mu}(k, r) = \frac{1}{\sqrt{2}}\frac{\langle r|\gamma^\mu|k]}{\langle r|k\rangle} \qquad \epsilon_L^{*\mu}(k, r) = \frac{1}{\sqrt{2}}\frac{\langle k|\gamma^\mu|r]}{[k|r]} \tag{1.44}$$

Here r is an arbitrary fixed lightlike 4-vector called reference vector. The reference vector cannot be collinear with k, as if it was eq. (1.44) would be singular. We will show later that the choice of r does not influence the value of the physical amplitude, as it should be since we claimed r to be arbitrary. This freedom of choice will prove quite powerful in the following.

It is easy to see that eq. (1.44) satisfy all the properties of polarization vectors:

$$[\epsilon_R^*(k)]^* = \epsilon_L^*(k) \qquad k_\mu\epsilon_{R,L}^{*\mu}(k) = 0 \tag{1.45}$$

having used the Dirac equation in the form $\not{k}|k] = 0$ in the second and $\gamma^0\gamma^\mu\gamma^0 = (\gamma^\mu)^\dagger$ in the first.Using this last identity one can see that

$$(\langle r|\gamma^\mu|k])^* = \langle k|\gamma^\mu|r] \tag{1.46}$$

and from this and eq. (1.41)

$$|\epsilon_R^*(k)|^2 = \frac{1}{2}\frac{\langle r|\gamma^\mu|k]\langle k|\gamma_\mu|r]}{\langle r|k\rangle[k|r]} = \frac{2\langle r|k\rangle[r|k]}{2\langle r|k\rangle[k|r]} = -1 \tag{1.47}$$

Similarly, using also the antisymmetry of the angle and square brackets

$$|\epsilon_L^*(k)|^2 = -1 \qquad \epsilon_L^*(k)\cdot(\epsilon_R^*(k))^* = \epsilon_R^*(k)\cdot(\epsilon_L^*(k))^* = 0 \tag{1.48}$$

[2]From here on we are going to use interchangeably the notation R, L and $+, -$.

as required for polarization vectors.

Furthermore for an appropriate choice of the reference vector it is possible to see that computing the polarization vector's components explicitly[3], for example for $\epsilon_R^*(k)$, it reduces to the well known form

$$\epsilon_R^*(k) = -\frac{1}{\sqrt{2}}(0, 1, -i, 0) \tag{1.49}$$

and analogously for $\epsilon_L^*(k)$.

Now let us turn to the issue of what happens if we change the reference vector r to another vector s. Computing the difference of the resulting polarization vectors, consider for example $\epsilon_R^*(k, r)$, we get

$$\begin{aligned}
\epsilon_R^*(k,r) - \epsilon_R^*(k,s) &= \frac{1}{\sqrt{2}}\left(\frac{\langle r|\gamma^\mu|k]}{\langle r|k\rangle} - \frac{\langle s|\gamma^\mu|k]}{\langle s|k\rangle}\right) && \text{(1.50a)}\\
&= \frac{1}{\sqrt{2}}\frac{1}{\langle r|k\rangle\langle s|k\rangle}(-\langle r|\gamma^\mu|k]\langle k|s\rangle + \langle s|\gamma^\mu|k]\langle r|k\rangle) && \text{(1.50b)}\\
&= \frac{1}{\sqrt{2}}\frac{1}{\langle r|k\rangle\langle s|k\rangle}(-\langle r|\gamma^\mu \not{k}|s\rangle + \langle s|\gamma^\mu \not{k}|r\rangle) && \text{(1.50c)}\\
&= \frac{1}{\sqrt{2}}\frac{1}{\langle r|k\rangle\langle s|k\rangle}\langle s|(\not{k}\gamma^\mu + \gamma^\mu \not{k})|r\rangle && \text{(1.50d)}\\
&= \frac{1}{\sqrt{2}}\frac{1}{\langle r|k\rangle\langle s|k\rangle}2k^\mu\langle s|r\rangle && \text{(1.50e)}
\end{aligned}$$

where in eq. (1.50c) we used eq. (1.33) and in eq. (1.50d) the anticommutation relation $\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu}$ defining the Clifford algebra of the gamma matrices. We see that the difference of the two polarization vectors is a scalar function of r and s times k^μ

$$\epsilon_R^*(k,r) = \epsilon_R^*(k,s) + f(r,s)k^\mu \tag{1.51}$$

This means that as soon as we contract eq. (1.51) with an on-shell amplitude, due to Ward identity which leads to cancellation of the $f(r,s)k^\mu$ term, the two different reference vectors will lead to physical indistinguishable results. Stated differently, gauge-invariant quantities do not depend by any means on the chosen reference vector.

Finally we are going to prove, using the formalism presented so far, the completeness relation

$$\sum_{\lambda=\pm} \epsilon_\lambda^{*\mu}(k,q)(\epsilon_\lambda^{*\rho}(k,q))^* = -\eta^{\mu\rho} + \frac{k^\mu q^\rho + k^\rho q^\mu}{k\cdot q} \tag{1.52}$$

which we will need when discussing the BCFW recurrence relation. Notice that

$$\begin{aligned}
\epsilon_+^\mu(k,q)\epsilon_-^\rho(k,q) &= -\frac{1}{2}\frac{\langle q\gamma^\mu k][q\gamma^\rho k\rangle}{\langle qk\rangle[qk]} && \text{(1.53a)}\\
&= \frac{1}{4}\frac{[k\gamma^\mu \not{q}\left(\mathbb{1}-\gamma^5\right)\gamma^\rho k\rangle}{2q\cdot k} && \text{(1.53b)}\\
&= \frac{1}{2}\frac{q_\nu}{2q\cdot k}[k\gamma^\mu\gamma^\nu\gamma^\rho k\rangle && \text{(1.53c)}\\
&= \frac{1}{2}\frac{q_\nu}{2q\cdot k}\overline{U}_R(k)_i\left(\gamma^\mu\gamma^\nu\gamma^\rho\right)_{ij}U_L(k)_j && \text{(1.53d)}\\
&= \frac{1}{2}\frac{q_\nu}{2q\cdot k}\overline{U}(k)_i\left(\gamma^\mu\gamma^\nu\gamma^\rho\left(\frac{\mathbb{1}-\gamma^5}{2}\right)\right)_{ij}U(k)_j && \text{(1.53e)}
\end{aligned}$$

[3]The needed formulae will be presented in the following and can be found for example in [13].

$$= \frac{1}{2}\frac{q_\nu k_\alpha}{2q\cdot k} Tr\left[\gamma^\alpha\gamma^\mu\gamma^\nu\gamma^\rho\left(\frac{\mathbb{1}-\gamma^5}{2}\right)\right] \tag{1.53f}$$

$$= -\frac{1}{2}\left(\eta^{\mu\rho} - \frac{1}{q\cdot k}(q^\rho k^\mu + q^\mu k^\rho) - \frac{i}{q\cdot k} q_\nu k_\alpha \epsilon^{\alpha\mu\nu\rho}\right) \tag{1.53g}$$

where in eq. (1.53a) we used eq. (1.28) and in eq. (1.53b) eq. (1.22), then in eq. (1.53c) the anticommutation of γ^5 and γ^μ and the fact that $P_R|k\rangle = |k\rangle$, and in eq. (1.53f) we substituted eq. (1.21). From this we can immediately get the analogous expression with opposite helicity simply renaming the μ and ρ indexes

$$\epsilon^\mu_-(k,q)\epsilon^\rho_+(k,q) = -\frac{1}{2}\left(\eta^{\mu\rho} - \frac{1}{q\cdot k}(q^\rho k^\mu + q^\mu k^\rho) + \frac{i}{q\cdot k} q_\nu k_\alpha \epsilon^{\alpha\mu\nu\rho}\right) \tag{1.54}$$

and finally adding eq. (1.53) and eq. (1.54) one gets eq. (1.52).

Alternative proof of the completeness relation

A more elegant way of getting to eq. (1.53) is through Lorentz-invariance arguments. Defining

$$F^{\mu\rho} = \epsilon^\mu_+(k,q)\epsilon^\rho_-(k,q) \tag{1.55}$$

the most general form of $F^{\mu\rho}$ allowed by its Lorentz structure is[4]

$$F^{\mu\rho} = A\eta^{\mu\rho} + Bk^\mu k^\rho + Cq^\mu q^\rho + D_1 q^\mu k^\rho + D_2 q^\rho k^\mu + E\epsilon^{\mu\rho\nu\sigma}k_\nu q_\sigma \tag{1.56}$$

Using then the Dirac equation in momentum space and eq. (1.47),eq. (1.48) it can be easily seen that

$$k_\mu k_\rho F^{\mu\rho} = 0 \tag{1.57a}$$
$$q_\mu q_\rho F^{\mu\rho} = 0 \tag{1.57b}$$
$$k_\mu q_\rho F^{\mu\rho} = 0 \tag{1.57c}$$
$$q_\mu k_\rho F^{\mu\rho} = 0 \tag{1.57d}$$
$$\eta_{\mu\rho} F^{\mu\rho} = -1 \tag{1.57e}$$
$$F_{\mu\rho} F^{\rho\mu} = +1 \tag{1.57f}$$

Solving this system of equations we get the value of the constants A,B,C,D_1,D_2 and E. Now due to the antisymmetry of the Levi-Civita tensor the E contribution vanishes in the first five equations which then can be solved independently leading to

$$A = -\frac{1}{2} \qquad B = C = 0 \qquad D_1 = D_2 = -\frac{A}{k\cdot q} \tag{1.58}$$

Solving the final one leads to

$$\begin{aligned} F_{\mu\rho}F^{\rho\mu} &= -A + E^2 k^\nu q^\sigma k_\alpha q_\beta \epsilon_{\mu\rho\nu\sigma}\epsilon^{\rho\mu\alpha\beta} \\ &= -A + 2E^2(\delta^\alpha_\nu\delta^\beta_\sigma - \delta^\beta_\nu\delta^\alpha_\sigma)k^\nu q^\sigma k_\alpha q_\beta \\ &= -A - 2E^2(k\cdot q)^2 \\ &\overset{!}{=} 1 \end{aligned} \tag{1.59}$$

[4]We considered also the fact that $\epsilon_\pm$ are Lorentz vectors and so no terms containing γ^5 are allowed.

where we used $\epsilon_{\mu\rho\nu\sigma}\epsilon^{\mu\rho\alpha\beta} = -2(\delta^\alpha_\nu\delta^\beta_\sigma - \delta^\beta_\nu\delta^\alpha_\sigma)$. Using the value of A previously found we get

$$E^2 = -\frac{1}{4(k\cdot q)^2} \qquad \Rightarrow \qquad E = \pm\frac{i}{2k\cdot q} \tag{1.60}$$

As can be seen there is an indetermination left in the sign of E due to the fact that eq. (1.57f) is quadratic in E. However it is possible to find another linear relation instead of eq. (1.57f): one can project out the E component by contracting $F^{\mu\rho}$ with $\epsilon_{\mu\rho\alpha\beta}k^\alpha q^\beta$, we then get

$$\epsilon_{\mu\rho\alpha\beta}k^\alpha q^\beta F^{\mu\rho} = 2E(k\cdot q)^2 \tag{1.61}$$

Then using the identity

$$\epsilon_{\mu\rho\alpha\beta}k^\alpha q^\beta F^{\mu\rho} = ik\cdot q \tag{1.62}$$

we can completely fix E and get the desired result, eq. (1.52).

1.1.4 2-component vs 4-component spinors

Up until now we used the four component Dirac spinors to describe a spin $\frac{1}{2}$ particle. However, as could already be seen looking at the proofs of many of the identities presented in section 1.1.2, there are instances where the two component Weyl spinors may be better suited in order to deal with the task at hand. Since the introduction of momentum-twistor variables is precisely one of these instances, from here on we are going to use Weyl spinors.

In terms of the angle and square brackets we are then going to write

$$u_R(p) = |p\rangle\,, \quad u_L(p) = |p]\,, \qquad \overline{u}_L(p) = \langle p|\,, \quad \overline{u}_R(p) = [p| \tag{1.63}$$

This redefinition can be easily related back to the previous one in terms of Dirac spinors through eq. (1.8).

The identities derived in section 1.1.2 do not undergo major changes, one simply has to substitute the γ^μ with σ^μ or $\overline{\sigma}^\mu \equiv (1, -\overrightarrow{\sigma})$, depending on the situation. For example eq. (1.28) becomes

$$\langle p|\sigma^\mu|q] = [q|\overline{\sigma}^\mu|p\rangle \tag{1.64}$$

which can be easily read of from eq. (1.27).

Most of the time though what we know about a given particle is neither the Dirac nor the Weyl form of the spinor components, but its momentum. So we need the expression relating spinor components to momentum components. This expression for a massive particle can be obtained by first solving the equation of motion in the particle's rest frame and then performing a Lorentz transformation to a generic reference frame. Considering massless particles one has also to take the massless limit, having chosen a suitable normalization condition.

Because of the fact that $\overline{u}_{L,R}$ belong to the dual space of $u_{R,L}$ we only need to know two of the $|p\rangle$, $|p]$,$\langle p|$, $[p|$ as a function of $p^\mu = (p^0, p^1, p^2, p^3)$, recovering the two remaining expression through the ϵ^{ij} tensor which gives the mapping to the dual space. We have for example

$$p^+ = p^0 + p^3\,, \quad p^- = p^0 - p^3 \tag{1.65}$$

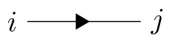
$$|p\rangle = \begin{pmatrix} \sqrt{p^+} \\ \frac{p^1+ip^2}{\sqrt{p^+}} \end{pmatrix} \qquad |p] = \begin{pmatrix} \frac{p^1-ip^2}{\sqrt{p^+}} \\ -\sqrt{p^+} \end{pmatrix} \tag{1.66}$$

It should be pointed out that, as can be seen from eq. (1.66), the spinor components are not rational functions of the momentum. Since we will be concerned with rational reconstruction of amplitudes, this factor of $\sqrt{p^+}$ needs to be removed. This can be achieved through a little group transformation, but we will come back to this issue later on.

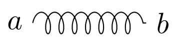
1.2 Color-ordered and MHV amplitudes

Color-ordered amplitudes

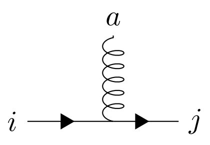
When computing scattering amplitudes usually a great number of diagrams contributes to any single process. Some of these diagrams are topologically independent, but others can be obtained from one another by simply switching the momenta of two or more particles and if needed reversing helicities or particle states (incoming/outgoing): this is called crossing. So not all of the diagrams associated to a given process need to be computed, but only the topologically independent ones, the remaining then can be immediately obtained by crossing. When dealing with QCD any particle carries also a color charge which results in some color structure inside the amplitude, and when performing either a computation from scratch or the crossing procedure one needs to consider that additional structure to. Fortunately the color structure of an amplitude always factorizes from the remaining kinematic part, moreover it has the same form for each diagram contributing to a given process[5]. Typically it is a product of the form $T^{a_1} \cdots T^{a_n}$ or a product of traces of the form $Tr[T^{a_1} \cdots T^{a_n}]$ with T^a generators of the $SU(3)$ group [6]. Why this is the case and the exact form of the color structure can be seen most efficiently in a diagrammatic way, starting from the Feynman rules of the theory. Considering only the color factors we have



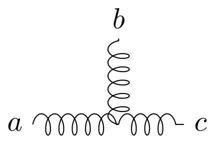
$$\propto \delta^{ij} \tag{1.67a}$$



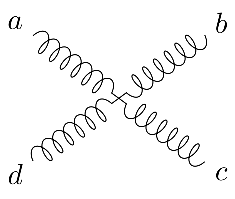
$$\propto \delta^{ab} \tag{1.67b}$$



$$\propto (T^a)^i_j \tag{1.67c}$$



$$\begin{aligned}\propto\ & f^{abc}T^a(\text{kinematics})\\ & +f^{abc}T^b(\text{kinematics})\\ & +f^{abc}T^c(\text{kinematics})\end{aligned} \tag{1.67d}$$



$$\propto f^{abe}f^{cde}(\text{kin.}) + f^{ace}f^{bde}(\text{kin.}) + f^{ade}f^{bce}(\text{kin.}) \tag{1.67e}$$

with f^{abc} structure constants of the group, i,j transforming in the fundamental/antifundamental representation (the upper vs lower index notation is used to distinguish among the two cases) and a,b,c,d,e transforming in the adjoint representation.

The first manipulation one can perform is the substitution of $f^{abc}T^c$ in the three-gluon vertex with a combination of products of T's using the fundamental relation of this Lie algebra

$$[T^a, T^b] = i\sqrt{2}f^{abc}T^c \tag{1.68}$$

[5]For a proof of these statements see for example [29].

[6]These generators are normalized in order to have $Tr[T^aT^b] = \delta^{ab}$.

Moreover we have that

$$f^{abc} = -\frac{i}{\sqrt{2}}\Big(Tr[T^aT^bT^c] - Tr[T^aT^cT^b]\Big) \tag{1.69}$$

which allows us to rewrite $f^{abe}f^{cde}$ in the four-gluon vertex as a product of traces. Another useful identity is

$$(T^a)^{i_1}_{j_1}(T^a)^{i_2}_{j_2} = \delta^{i_2}_{j_1}\delta^{i_1}_{j_2} - \frac{1}{N_c}\delta^{i_1}_{j_1}\delta^{i_2}_{j_2} \tag{1.70}$$

Using eq. (1.69) and eq. (1.70) we can act directly on the Feynman diagrams to get the color structure. [7]

For an n gluon tree-level amplitude for example we get

$$\mathcal{M}^{tree}_n(\{k_i,\lambda_i,a_i\}) = g^{n-2}\sum_{\sigma\in S_n/Z_n} Tr[T^{a_{\sigma(1)}}\cdots T^{a_{\sigma(n)}}]\mathbf{M}^{tree}_n(\sigma(1^{\lambda_1}),\cdots,\sigma(n^{\lambda_n})) \tag{1.71}$$

Here g is the gauge coupling and k_i, λ_i, a_i are the gluon momenta, helicities and colors respectively. $\mathbf{M}^{tree}_n(1^{\lambda_1},\cdots,n^{\lambda_n})$ are the partial amplitudes which contain all the kinematic information and are color ordered, meaning that they receive contributions only from diagrams with a particular cyclic ordering of the gluons. The sum runs over all the non-cyclic permutations of the n colors. The amplitudes we are going to compute are the independent partial amplitudes.

The partial amplitudes satisfy a number of properties and relationships, some of which are:

- $\mathbf{M}^{tree}_n(1^{\lambda_1},\cdots,n^{\lambda_n})$ is gauge invariant
- cyclic invariance:
$$\mathbf{M}^{tree}_n(1,2,\cdots,n) = \mathbf{M}^{tree}_n(2,\cdots,n,1) \tag{1.72}$$
- reflection invariance:
$$\mathbf{M}^{tree}_n(n,n-1\cdots,2,1) = (-1)^n\mathbf{M}^{tree}_n(1,2,\cdots,n-1,n) \tag{1.73}$$
- the so called photon decoupling identity:
$$\mathbf{M}^{tree}_n(1,2,3,\cdots,n) + \mathbf{M}^{tree}_n(2,1,3,\cdots,n) + \cdots + \mathbf{M}^{tree}_n(2,3,\cdots,1,n) = 0 \tag{1.74}$$

The fact that the partial amplitude is still gauge invariant is very important because it allows us to use all the tools available for the computation of the complete amplitude on the partial amplitudes as well. In particular the Ward identity holds for gauge invariant amplitudes and ensures, among other things, that if we use the spinor-helicity formalism the value of the amplitude is independent from the reference vectors chosen to represent the polarization vectors of the gluons through eq. (1.44).

Finally one can rederive the Feynman rules for the partial amplitudes, which will have a simpler form because of the missing color structure. For the three-point and four point gluon self-interaction we have

$$V_3^{\mu\nu\rho}(P,Q) = \frac{i}{\sqrt{2}}(\eta^{\nu\rho}(P-Q)^\mu + 2\eta^{\rho\mu}Q^\nu - 2\eta^{\mu\nu}P^\rho) \tag{1.75}$$

$$V_4^{\mu\nu\rho\sigma} = \frac{i}{2}(2\eta^{\mu\rho}\eta^{\nu\sigma} - \eta^{\mu\nu}\eta^{\rho\sigma} - \eta^{\mu\sigma}\eta^{\nu\rho}) \tag{1.76}$$

Notice that in the three-point vertex momentum conservation has been used to get rid of the dependence on one of the three gluon momenta, leaving only the dependence on the two independent ones.

[7] For further details see for example [13].

Maximally Helicity Violating amplitudes

Maximally helicity violating (MHV) amplitudes were first discovered in 1986 by Parke and Taylor [24]. They are tree-level amplitudes in certain specific helicity configurations having as external particles either all gluons or one quark, one antiquark and all the rest gluons.

Taking an all gluon color-ordered amplitude one has that:

$$\mathbf{M}(1^+, 2^+, \cdots, n-1^+, n^+) = 0 \tag{1.77a}$$

$$\mathbf{M}(1^-, 2^-, \cdots, n-1^-, n^-) = 0 \tag{1.77b}$$

$$\mathbf{M}(1^+, \cdots, i^-, \cdots, n^+) = 0 \tag{1.78a}$$

$$\mathbf{M}(1^-, \cdots, i^+, \cdots, n^-) = 0 \tag{1.78b}$$

$$\mathbf{M}(1^+, \cdots, i^-, \cdots, j^-, \cdots, n^+) = \frac{\langle i|j\rangle^4}{\langle 1|2\rangle\langle 2|3\rangle \cdots \langle n-1|n\rangle\langle n|1\rangle} \tag{1.79a}$$

$$\mathbf{M}(1^-, \cdots, i^+, \cdots, j^+, \cdots, n^-) = (-1)^n \frac{[i|j]^4}{[1|2][2|3] \cdots [n-1|n][n|1]} \tag{1.79b}$$

Equation (1.77) can be easily proved reasoning on the Lorentz structure of the amplitude: in each three-gluon vertex we have a linear dependence in the momentum meaning that each internal three-gluon vertex adds a Lorentz index to the amplitude. These indexes need to be saturated with the external polarization vectors, but since in an n-gluon amplitude there are at most $n-2$ of these internal vertexes we are always left with at least two external polarizations which saturate among themselves. In other words any tree-level n-gluon amplitude must contain at least one scalar product of two polarization vectors $\epsilon_i \cdot \epsilon_j$, independently of the helicities. However if all of gluons have positive (negative) helicity than this scalar product will be proportional to $\langle r_i|r_j\rangle$ ($[r_i|r_j]$) for every i, j and were r_i,r_j are the reference vectors of the i-th and j-th gluon respectively. Being the reference vectors arbitrary we can choose all of the r_i for $i = 1, \cdots, n$ to be equal, then due to the antisymmetry of the angle and square products $\epsilon_i \cdot \epsilon_j = 0$ meaning that the whole amplitude must vanish.

Proving the other identities is far less simple and can be done for example by induction using either the Berends-Giele or the BCFW recursion.[8] Similar results can be obtained for the case of an amplitude containing one quark, one antiquark and the rest gluons. Notice that the helicity of the quark and antiquark is opposite due the convention we adopted of choosing all particles as incoming.

$$\mathbf{M}(q(1)^-, g(2)^+, \cdots, g(n-1)^+, \overline{q}(n)^+) = 0 \tag{1.80a}$$

$$\mathbf{M}(q(1)^-, g(2)^-, \cdots, g(n-1)^-, \overline{q}(n)^+) = 0 \tag{1.80b}$$

$$\mathbf{M}(q(1)^-, g(2)^+, \cdots, g(i)^-, \cdots, g(n-1)^+, \overline{q}(n)^+) = \frac{\langle 1|i\rangle^3\langle n|i\rangle}{\langle 1|2\rangle\langle 2|3\rangle \cdots \langle n-1|n\rangle\langle n|1\rangle} \tag{1.81a}$$

[8]For a proof of the MHV formulas see for example [28].

$$\mathbf{M}(q(1)^-, g(2)^-, \cdots, g(i)^+, \cdots, g(n-1)^-, \overline{q}(n)^+) = (-1)^{n-1} \frac{[1|i][n|i]^3}{[1|2][2|3]\cdots[n-1|n][n|1]} \tag{1.81b}$$

The corresponding relations with the helicities of the quark and antiquark interchanged can be obtained by charge conjugation.

1.3 Parametrizations of an n-point function

Considering any given amplitude $\mathbf{M}_n(1, \cdots, n)$, we have that $|\mathbf{M}_n|^2$ can be described in terms of $3n - 10$ independent variables: given n particles we have $4n$ parameters (the momentum components), n are fixed by the on-shellness conditions, 4 are fixed by overall momentum conservation, and finally other 6 by Lorentz invariance. Let us call these variables x_i, $i \in \{1, \cdots, 3n - 10\}$.

The amplitude itself however may depend on all of the $4n$ variables, so this means that it can be factored as follows

$$\mathbf{M}_n(1, \cdots, n) = \Phi_n \mathbf{M}_n(x_1, \cdots, x_{3n-10}) \tag{1.82}$$

where Φ_n is a phase depending on all the dependent and independent $4n$ variables and $\mathbf{M}_n(x_1, \cdots, x_{3n-10})$ is the part depending only on the $3n-10$ independent variables which contributes to the cross section.

1.3.1 Little group transformations and scaling of the amplitude

The little group is the subgroup of the Lorentz group which leaves a given on-shell momentum invariant. Recalling eq. (1.26) one can easily see that

$$|p\rangle \mapsto t|p\rangle\ , \qquad |p] \mapsto \frac{1}{t}|p] \tag{1.83}$$

is a little group transformation. For real momenta t has to be a complex phase in order for eq. (1.20) to be preserved.

Since amplitudes that contain massless particles can always be written in terms of spinor products we need to scale external lines under the rule eq. (1.83) as well (internal lines are made of invariant momenta). As can be seen from the expressions of the momentum and the polarization vectors in terms of angle and square brackets we have that:

- scalar particles do not scale under the transformation
- angle and square spinors for fermions scale as t^{-2h} for $h = \pm\frac{1}{2}$, which is simply a restatement of eq. (1.83)
- polarization vectors for spin-1 bosons scale as t^{-2h} for $h = \pm 1$. They do not transform under scaling of the reference momentum

Thus we have that transforming under the i-th particle's little group

$$\mathbf{M}_n\bigg(\{|1\rangle, |1], h_1\}, \cdots, \{t_i|i\rangle, \frac{1}{t_i}|i], h_i\}, \cdots, \{|n\rangle, |n], h_n\}\bigg) = \\ t_i^{-2h_i}\mathbf{M}_n\bigg(\{|1\rangle, |1], h_1\}, \cdots, \{|i\rangle, |i], h_i\}, \cdots, \{|n\rangle, |n], h_n\}\bigg) \tag{1.84}$$

The amplitude can now be factorized into a phase transforming under the little group and a part invariant under this scaling. Notice that the phase depends only on the helicity of the external particles.

Factoring out the phase

It will be convenient to extract the Φ_n factor. There is no unique choice for it, but depending only on the transformation properties of $\mathbf{M}_n$ under the little group it then only depends on the helicities of the external particles and can thus be built without any other knowledge about the amplitude itself. A general expression for Φ_n valid for any number of external legs can be found in [2]

$$\Phi_n(1^{h_1}, \cdots, n^{h_n}) = \left(\frac{\langle 1|3\rangle}{[1|2]\langle 2|3\rangle}\right)^{h_1} \prod_{i=2}^{n} \left(\frac{\langle 1|i\rangle^2 [1|2]\langle 2|3\rangle}{\langle 1|3\rangle}\right)^{-h_i} \tag{1.85}$$

The function which we will be computing through rational reconstruction is the phase-free ratio $\mathbf{M}_n(1, \cdots, n)/\Phi_n$ i.e. the function $\mathbf{M}_n(x_1, \cdots, x_{3n-10})$.

The x_i can be chosen in different ways depending on the task at hand. For example a possible choice are the kinematic invariants $s_{ij\cdots k} = (p_i + p_j + \cdots p_k)^2$. Among these one may then choose to parametrize an n-point amplitude only through double-invariants s_{ij}, or some combination of double, triple, etc. invariants in terms of which all the others can be expressed. The relations among dependent and independent variables is given by overall momentum conservation and (for $n \geq 6$) by geometrical conditions connected with the four-dimensionality of space.[9]

The choice of the parametrization for the amplitude is crucial for what follows. We need the amplitude to be a rational function in order to apply rational reconstruction methods. In general amplitudes are not rational functions of the momenta. However it is a rational function of the angle and square spinors, so what we are looking for is a suitable parametrization for the spinor components. These are the so called momentum-twistor variables.

1.3.2 Momentum-twistor variables

Before dealing with momentum-twistor variables, which provide the parametrization needed for rational reconstruction, we will briefly review some concepts about momentum twistors. A detailed description of these can e found for example in [14] or [32].

Momentum twistors

Momentum conservation can be represented geometrically. The fact that p_i^μ sum to zero means that the vectors close into a contour, an example for a six particle scattering is represented in figure 1.1.

So instead of defining this contour by its edges (momenta) we could define it by its cusps, points of the dual space which we will be calling y_i^μ. They can be related to the momenta by

$$p_i^\mu = (y_{i+1} - y_i)^\mu \tag{1.86}$$

The dual coordinates are not space-time coordinates and have mass dimension 1. The momentum conservation in dual space corresponds to the periodicity condition $y_{n+1} = y_1$. As can be seen the ordering of the external particles matters in this picture, so we restrict our attention to color-ordered amplitudes where we can define

$$y_{ij}^\mu \equiv (y_i - y_j)^\mu = (p_j + \cdots + p_{i-1})^\mu \tag{1.87}$$

[9]For a detailed treatment see [1].

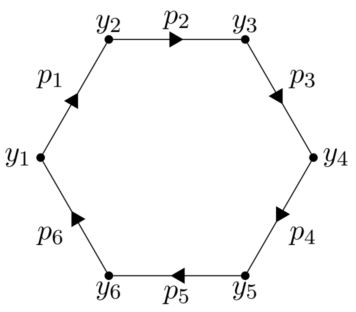


Figure 1.1: Relation between the momenta p_i and the dual coordinates y_i.

We can also provide a definition of the dual coordinates in terms of spinors by taking into account the Dirac equation

$$0 = \not{p}_i|i\rangle = (\not{y}_{i+1} - \not{y}_i)|i\rangle \tag{1.88}$$

This relation, called incidence relation, allows to define a new variable $|\mu_i]$

$$|\mu_i] = \not{y}_i|i\rangle = \not{y}_{i+1}|i\rangle \tag{1.89}$$

With this definition we have that

$$y_i^\mu = \frac{1}{2}\frac{\langle i|\gamma^\mu|\mu_{i-1}] - \langle i-1|\gamma^\mu|\mu_i]}{\langle i|i-1\rangle} \tag{1.90}$$

since, using eq. (1.33) and the Clifford algebra relation $\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu}$

$$\begin{aligned}\langle i|\gamma^\mu|\mu_{i-1}] - \langle i-1|\gamma^\mu|\mu_i] &= \langle i|\gamma^\mu|\not{y}_i|i-1\rangle - \langle i-1|\gamma^\mu|\not{y}_i|i\rangle \\ &= (\langle i|\gamma^\mu\gamma^\nu|i-1\rangle - \langle i-1|\gamma^\mu\gamma^\nu|i\rangle)y_{i,\nu} \\ &= -\langle i-1|(\gamma^\nu\gamma^\mu + \gamma^\mu\gamma^\nu)|i\rangle y_{i\nu} \\ &= -2\langle i-1|i\rangle y_i^\mu\end{aligned} \tag{1.91}$$

In other words we translated the dual coordinates y_i into $Z_i \equiv (|i\rangle, |\mu_i])$. These new four component spinor variables Z_i are called momentum twistors.[10]

Under little group transformation the Z_i undergo a uniform scaling

$$Z_i \mapsto t_i Z_i(|i\rangle, |\mu_i]) = Z_i(t_i|i\rangle, t_i|\mu_i]) \tag{1.92}$$

Furthermore, the relation between y_i and $p_i^\mu = \frac{1}{2}\langle i|\gamma^\mu|i]$ implies that $|i]$ can be expressed in terms of $|i\rangle$ and $|\mu_i]$:

$$[i| = \frac{\langle i+1|i\rangle[\mu_{i-1}| + \langle i|i-1\rangle[\mu_{i+1}| + \langle i-1|i+1\rangle[\mu_i|}{\langle i-1|i\rangle\langle i+1|i\rangle} \tag{1.93}$$

This relation can be easily checked:

$$\begin{aligned}&\langle i+1|i\rangle[\mu_{i-1}| + \langle i|i-1\rangle[\mu_{i+1}| + \langle i-1|i+1\rangle[\mu_i| = \\ &\qquad \not{p}_i\langle i+1|i\rangle\langle i-1| + \not{p}_{i+1}\langle i|i-1\rangle\langle i+1| + \not{p}_i\langle i-1|i+1\rangle\langle i|\end{aligned} \tag{1.94}$$

using the Schouten identity

$$\langle i+1|i\rangle\langle i-1| + \langle i-1|i+1\rangle\langle i| = -\langle i|i-1\rangle\langle i+1| \tag{1.95}$$

[10] See [15].

we have

$$\begin{aligned}
eq.\ (1.94) &= (\not{p}_{i+1} - \not{p}_i)\langle i|i-1\rangle\langle i+1| && (1.96)\\
&= \langle i|i-1\rangle\langle i+1|(\frac{\mathbb{1}+\gamma^5}{2})(\not{p}_{i+1} - \not{p}_i) && (1.97)\\
&= \langle i|i-1\rangle\langle i+1|(|i+1]\,[i+1|-|i]\,[i|) && (1.98)\\
&= -\langle i|i-1\rangle\langle i+1|i]\,[i| && (1.99)
\end{aligned}$$

Starting from n momentum twistors $(Z_i, \cdots, Z_n)$, we see that the momenta p_i satisfy by definition the massless condition and momentum conservation. This means that all momentum twistors can be freely chosen without any constraint. Notice that providing a rational parametrization for the twistor space, also provides one for the momentum space through eq. (1.93).

A rational parametrization of the amplitude

Now we are going to present some possible parametrizations of the amplitude in momentum space. Define the components of angle and square spinors as

$$|i\rangle = \begin{pmatrix} \xi_i \\ \eta_i \end{pmatrix}, \qquad |i] = \begin{pmatrix} \overline{\xi}_i \\ \overline{\eta}_i \end{pmatrix} \tag{1.100}$$

Through the two dimensional ϵ tensor we can map these to the dual space getting

$$\langle i| = \begin{pmatrix} \eta_i \\ -\xi_i \end{pmatrix}, \qquad [i| = \begin{pmatrix} -\overline{\eta}_i \\ \overline{\xi}_i \end{pmatrix} \tag{1.101}$$

The anti-symmetric spinor products read

$$\begin{aligned}
\langle i|j\rangle &= -\xi_i\eta_j + \xi_j\eta_i && (1.102)\\
[i|j] &= \overline{\xi}_i\overline{\eta}_j - \overline{\xi}_j\overline{\eta}_i && (1.103)
\end{aligned}$$

Now imposing the momentum conservation in each of the space-time components leads to a system of four equations in terms of $\{\xi_i, \eta_i, \overline{\xi}_i, \overline{\eta}_i\}$

$$\sum_i p_i^\mu = \frac{1}{2}\sum_i \langle i|\sigma^\mu|i] = 0 \tag{1.104}$$

$$\begin{cases}
\sum_i (\eta_i\overline{\xi}_i - \xi_i\overline{\eta}_i) = 0\\
\sum_i (\eta_i\overline{\eta}_i - \xi_i\overline{\xi}_i) = 0\\
\sum_i (\eta_i\overline{\eta}_i + \xi_i\overline{\xi}_i) = 0\\
\sum_i (\eta_i\overline{\xi}_i + \xi_i\overline{\eta}_i) = 0
\end{cases}$$

which is equivalent to

$$\sum_i \eta_i\overline{\xi}_i = \sum_i \xi_i\overline{\eta}_i = \sum_i \eta_i\overline{\eta}_i = \sum_i \xi_i\overline{\xi}_i = 0 \tag{1.105}$$

At this point one needs to find $3n-10$ variable for the $\{\xi_i, \eta_i, \overline{\xi}_i, \overline{\eta}_i\}$ to depend on, such that these equations are satisfied. Clearly since we want the amplitude to be a rational function of the x_i, the spinor components need to be as well rational functions of them. Notice that, in general, because we are free to choose the normalization and the phase of the spinor components, there is no unique way to rewrite them in terms of the x_i.

One simple case is that of $n=4$, we have only two independent variables and choose them as:

$$x_1 = s_{12} \qquad x_2 = \frac{s_{23}}{s_{12}} \tag{1.106}$$

The spinor components can be chosen to be

$$\begin{aligned} \xi_1 = 1\,, \quad \xi_2 = 0\,, \quad \xi_1 = \frac{1}{x_1}\,, \quad \eta_1 = 0\,, \quad \eta_2 = \eta_3 = \eta_4 = 1 \\ \overline{\xi}_1 = -1\,, \quad \overline{\xi}_4 = 0\,, \quad \overline{\eta}_2 = 0 \end{aligned} \tag{1.107}$$

The remaining variables $\xi_4, \overline{\xi}_2, \overline{\xi}_3, \overline{\eta}_1, \overline{\eta}_3, \overline{\eta}_4$ are then fixed from eq. (1.105) and eq. (1.106).

For $n \geq 5$ we will present two explicit choices.

A possible explicit parametrization

This first parametrization we are reporting is drawn from Simon Badger et al. (cfr. e.g. [2, 3]).

The first set of equations presented are the relations between the x_i and the angle and square spinors (recall that $s_{ij} = \langle i|j\rangle[j|i]$), the second set relates the spinor components to the x_i.

$$x_1 = s_{12} \tag{1.108}$$

$$x_i = -\frac{\langle i|i+1\rangle\langle i+2|1\rangle}{\langle 1|i\rangle\langle i+1|i+2\rangle} \qquad \text{for } i = 2, \cdots, n-2 \tag{1.109}$$

$$x_{n-1} = \frac{s_{23}}{s_{12}} \tag{1.110}$$

$$x_i = \sum_{j=2}^{i-n+4} \frac{\langle i-n+5|j|2]}{[1|2]\langle 1|i-n+5\rangle} \qquad \text{for } i = 2, \cdots, 2n-6 \tag{1.111}$$

$$x_i = \sum_{j=2}^{i-2n+9} \frac{\langle 1|p_2+p_3|j|i-2n+10\rangle}{s_{23}\langle 1|i-2n+10\rangle} \qquad \text{for } i = 2n-5, \cdots, 3n-11 \tag{1.112}$$

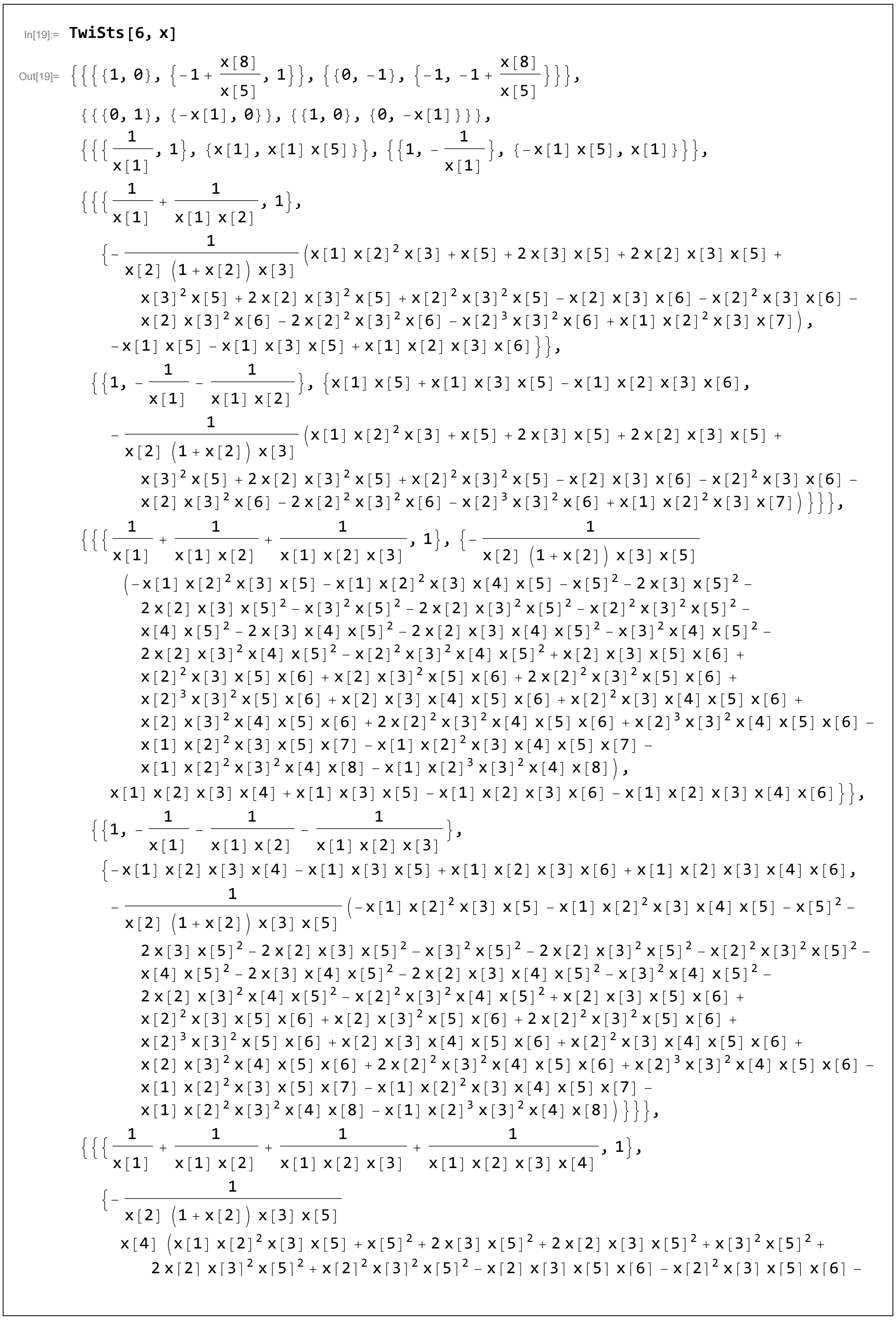
$$x_{3n-10} = \frac{s_{123}}{s_{12}} \tag{1.113}$$

and the spinor components in terms of the x_i

$$\begin{aligned} \xi_1 &= 1\,, & \xi_2 &= 0\,, & \xi_i &= \textstyle\sum_{j=1}^{i-2} \frac{1}{\prod_{k=1}^{j} x_k} \quad \text{for } i = 3, \cdots, n \\ \eta_1 &= 0\,, & \eta_i &= 1 & &\text{for } i = 2, \cdots, n \\ \overline{\xi_1} &= -1 + \tfrac{x_{3n-10}}{x_{n-1}}\,, & \overline{\xi}_2 &= -x_1\,, & \overline{\xi}_3 &= x_1 \\ \overline{\eta}_1 &= 1\,, & \overline{\eta}_2 &= 0\,, & \overline{\eta}_3 &= x_1 x_{n-1} \end{aligned} \tag{1.114}$$

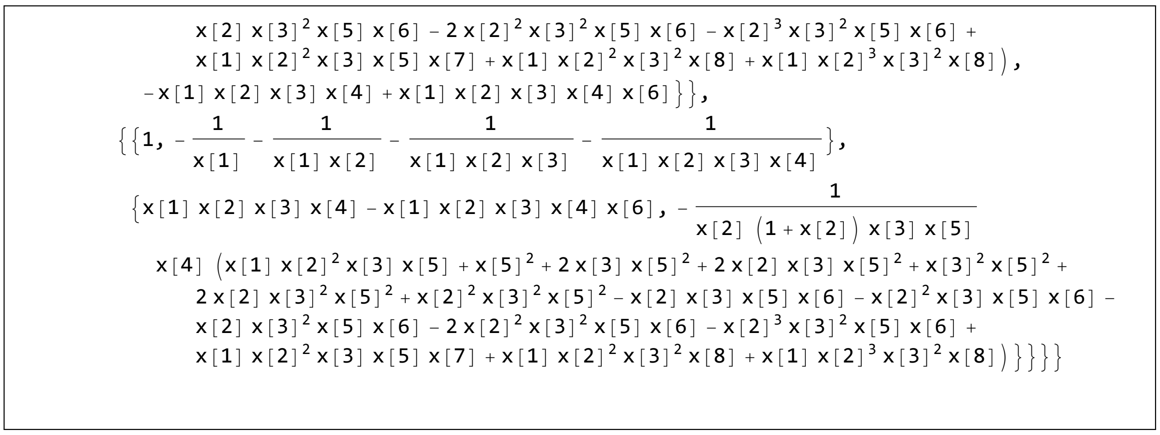
This parametrization satisfies eq. (1.108),eq. (1.109),eq. (1.110) and eq. (1.113) automatically. The remaining $2n-6$ spinor components yet unspecified, namely $\overline{\xi}_i$ and $\overline{\eta}_i$ for $i = 4, \cdots, n$, are completely determined by the $2n-6$ constrains of eq. (1.111), eq. (1.112) and eq. (1.105).

In figure 1.2 and figure 1.3 we provide an example of this parametrization for a 6-point amplitude generated through our Mathematica code. Because of the length of the output we split it into two figures.



```
In[19]:= TwiSts[6, x]

Out[19]= {{{{1, 0}, {-1 + x[8]/x[5], 1}}, {{0, -1}, {-1, -1 + x[8]/x[5]}}},
  {{{0, 1}, {-x[1], 0}}, {{1, 0}, {0, -x[1]}}},
  {{{1/x[1], 1}, {x[1], x[1] x[5]}}, {{1, -1/x[1]}, {-x[1] x[5], x[1]}}},
  {{{1/x[1] + 1/(x[1] x[2]), 1},
    {-1/(x[2] (1 + x[2]) x[3]) (x[1] x[2]^2 x[3] + x[5] + 2 x[3] x[5] + 2 x[2] x[3] x[5] +
        x[3]^2 x[5] + 2 x[2] x[3]^2 x[5] + x[2]^2 x[3]^2 x[5] - x[2] x[3] x[6] - x[2]^2 x[3] x[6] -
        x[2] x[3]^2 x[6] - 2 x[2]^2 x[3]^2 x[6] - x[2]^3 x[3]^2 x[6] + x[1] x[2]^2 x[3] x[7]),
      -x[1] x[5] - x[1] x[3] x[5] + x[1] x[2] x[3] x[6]}},
   {{1, -1/x[1] - 1/(x[1] x[2])}, {x[1] x[5] + x[1] x[3] x[5] - x[1] x[2] x[3] x[6],
     -1/(x[2] (1 + x[2]) x[3]) (x[1] x[2]^2 x[3] + x[5] + 2 x[3] x[5] + 2 x[2] x[3] x[5] +
        x[3]^2 x[5] + 2 x[2] x[3]^2 x[5] + x[2]^2 x[3]^2 x[5] - x[2] x[3] x[6] - x[2]^2 x[3] x[6] -
        x[2] x[3]^2 x[6] - 2 x[2]^2 x[3]^2 x[6] - x[2]^3 x[3]^2 x[6] + x[1] x[2]^2 x[3] x[7])}}},
  {{{1/x[1] + 1/(x[1] x[2]) + 1/(x[1] x[2] x[3]), 1}, {-1/(x[2] (1 + x[2]) x[3] x[5])
      (-x[1] x[2]^2 x[3] x[5] - x[1] x[2]^2 x[3] x[4] x[5] - x[5]^2 - 2 x[3] x[5]^2 -
        2 x[2] x[3] x[5]^2 - x[3]^2 x[5]^2 - 2 x[2] x[3]^2 x[5]^2 - x[2]^2 x[3]^2 x[5]^2 -
        x[4] x[5]^2 - 2 x[3] x[4] x[5]^2 - 2 x[2] x[3] x[4] x[5]^2 - x[3]^2 x[4] x[5]^2 -
        2 x[2] x[3]^2 x[4] x[5]^2 - x[2]^2 x[3]^2 x[4] x[5]^2 + x[2] x[3] x[5] x[6] +
        x[2]^2 x[3] x[5] x[6] + x[2] x[3]^2 x[5] x[6] + 2 x[2]^2 x[3]^2 x[5] x[6] +
        x[2]^3 x[3]^2 x[5] x[6] + x[2] x[3] x[4] x[5] x[6] + x[2]^2 x[3] x[4] x[5] x[6] +
        x[2] x[3]^2 x[4] x[5] x[6] + 2 x[2]^2 x[3]^2 x[4] x[5] x[6] + x[2]^3 x[3]^2 x[4] x[5] x[6] -
        x[1] x[2]^2 x[3] x[5] x[7] - x[1] x[2]^2 x[3] x[4] x[5] x[7] -
        x[1] x[2]^2 x[3]^2 x[4] x[8] - x[1] x[2]^3 x[3]^2 x[4] x[8]),
     x[1] x[2] x[3] x[4] + x[1] x[3] x[5] - x[1] x[2] x[3] x[6] - x[1] x[2] x[3] x[4] x[6]}},
   {{1, -1/x[1] - 1/(x[1] x[2]) - 1/(x[1] x[2] x[3])},
    {-x[1] x[2] x[3] x[4] - x[1] x[3] x[5] + x[1] x[2] x[3] x[6] + x[1] x[2] x[3] x[4] x[6],
     -1/(x[2] (1 + x[2]) x[3] x[5]) (-x[1] x[2]^2 x[3] x[5] - x[1] x[2]^2 x[3] x[4] x[5] - x[5]^2 -
        2 x[3] x[5]^2 - 2 x[2] x[3] x[5]^2 - x[3]^2 x[5]^2 - 2 x[2] x[3]^2 x[5]^2 - x[2]^2 x[3]^2 x[5]^2 -
        x[4] x[5]^2 - 2 x[3] x[4] x[5]^2 - 2 x[2] x[3] x[4] x[5]^2 - x[3]^2 x[4] x[5]^2 -
        2 x[2] x[3]^2 x[4] x[5]^2 - x[2]^2 x[3]^2 x[4] x[5]^2 + x[2] x[3] x[5] x[6] +
        x[2]^2 x[3] x[5] x[6] + x[2] x[3]^2 x[5] x[6] + 2 x[2]^2 x[3]^2 x[5] x[6] +
        x[2]^3 x[3]^2 x[5] x[6] + x[2] x[3] x[4] x[5] x[6] + x[2]^2 x[3] x[4] x[5] x[6] +
        x[2] x[3]^2 x[4] x[5] x[6] + 2 x[2]^2 x[3]^2 x[4] x[5] x[6] + x[2]^3 x[3]^2 x[4] x[5] x[6] -
        x[1] x[2]^2 x[3] x[5] x[7] - x[1] x[2]^2 x[3] x[4] x[5] x[7] -
        x[1] x[2]^2 x[3]^2 x[4] x[8] - x[1] x[2]^3 x[3]^2 x[4] x[8])}}},
  {{{1/x[1] + 1/(x[1] x[2]) + 1/(x[1] x[2] x[3]) + 1/(x[1] x[2] x[3] x[4]), 1},
    {-1/(x[2] (1 + x[2]) x[3] x[5])
      x[4] (x[1] x[2]^2 x[3] x[5] + x[5]^2 + 2 x[3] x[5]^2 + 2 x[2] x[3] x[5]^2 + x[3]^2 x[5]^2 +
          2 x[2] x[3]^2 x[5]^2 + x[2]^2 x[3]^2 x[5]^2 - x[2] x[3] x[5] x[6] - x[2]^2 x[3] x[5] x[6] -
```

Figure 1.2: Example of momentum-twistor parametrization for a six-point function



```
      x[2] x[3]^2 x[5] x[6] - 2 x[2]^2 x[3]^2 x[5] x[6] - x[2]^3 x[3]^2 x[5] x[6] +
      x[1] x[2]^2 x[3] x[5] x[7] + x[1] x[2]^2 x[3]^2 x[8] + x[1] x[2]^3 x[3]^2 x[8]),
   -x[1] x[2] x[3] x[4] + x[1] x[2] x[3] x[4] x[6]}},
          1          1              1                    1
{{1, - ----- - ----------- - ---------------- - ---------------------},
        x[1]   x[1] x[2]    x[1] x[2] x[3]    x[1] x[2] x[3] x[4]
                                                            1
 {x[1] x[2] x[3] x[4] - x[1] x[2] x[3] x[4] x[6], - ----------------------------
                                                    x[2] (1 + x[2]) x[3] x[5]
   x[4] (x[1] x[2]^2 x[3] x[5] + x[5]^2 + 2 x[3] x[5]^2 + 2 x[2] x[3] x[5]^2 + x[3]^2 x[5]^2 +
      2 x[2] x[3]^2 x[5]^2 + x[2]^2 x[3]^2 x[5]^2 - x[2] x[3] x[5] x[6] - x[2]^2 x[3] x[5] x[6] -
      x[2] x[3]^2 x[5] x[6] - 2 x[2]^2 x[3]^2 x[5] x[6] - x[2]^3 x[3]^2 x[5] x[6] +
      x[1] x[2]^2 x[3] x[5] x[7] + x[1] x[2]^2 x[3]^2 x[8] + x[1] x[2]^3 x[3]^2 x[8])}}}}
```

Figure 1.3: Example of momentum-twistor parametrization for a six-point function

Another equivalent choice of variables

The following is an equivalent choice by Tiziano Peraro drawn from [26].

$$x_1 = s_{12} \tag{1.115}$$

$$x_{i-2} = \frac{\langle 1|3\rangle\langle 2|i\rangle}{\langle 2|3\rangle\langle 1|i\rangle} \quad \text{for } i = 4, \cdots, n \tag{1.116}$$

$$x_{n-1} = \frac{s_{23}}{s_{12}} \tag{1.117}$$

$$x_{n+i-4} = \frac{(s_{12}+s_{13})\langle 1|i|2]}{s_{12}\langle 1|3|2]} - \frac{s_{i1}}{s_{12}} \quad \text{for } i = 4, \cdots, n-2 \tag{1.118}$$

$$x_{2n+i-9} = \frac{\langle 1|i|2|3\rangle}{s_{12}\langle 1|3\rangle} \quad \text{for } i = 4, \cdots, n-2 \tag{1.119}$$

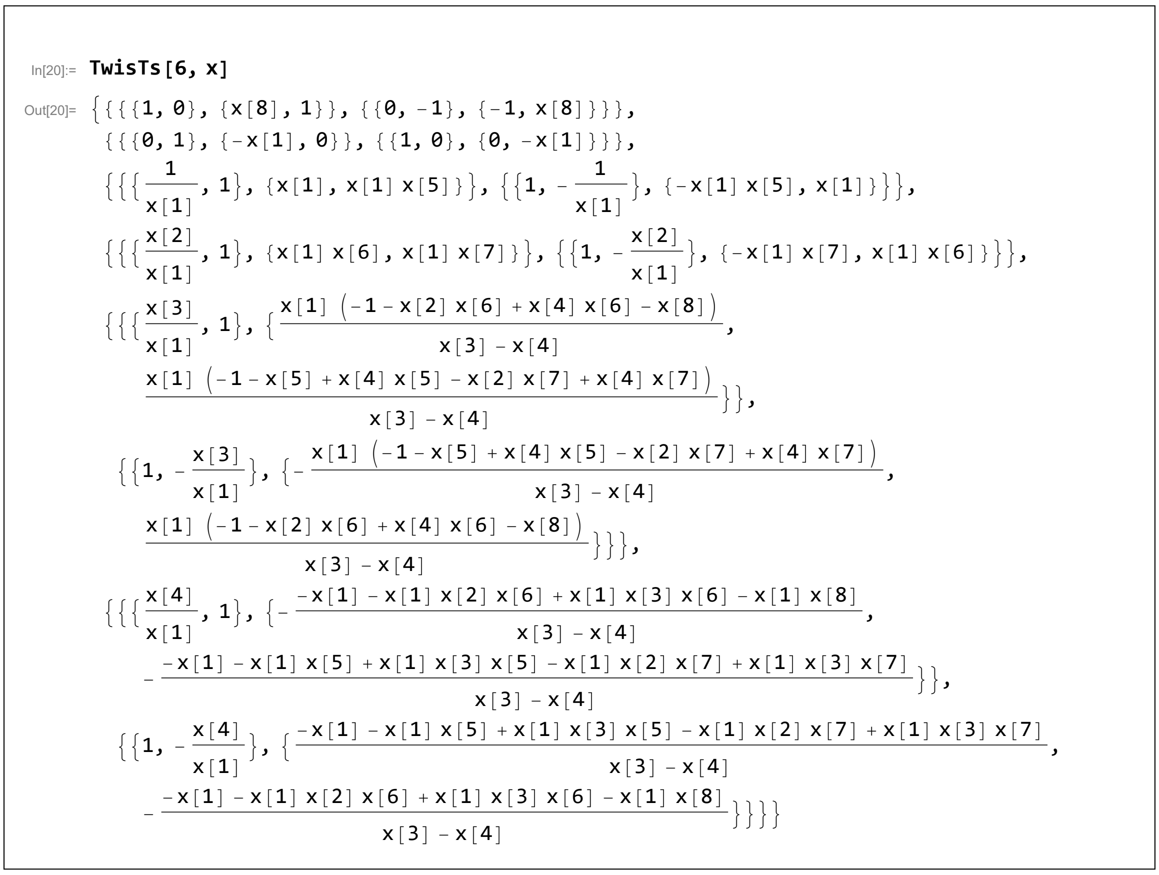
$$x_{3n-10} = \frac{s_{12}+s_{13}}{s_{23}} \tag{1.120}$$

and the spinor components in terms of the x_i:

$$\begin{array}{llll} \xi_1 = 1\,, & \xi_2 = 0\,, & \xi_3 = \frac{1}{x_1}\,, & \xi_i = \frac{x_{i-2}}{x_1} \quad \text{for } i = 4, \cdots, n \\ \eta_1 = 0\,, & \eta_i = 1 & \text{for } i = 2, \cdots, n & \\ \overline{\xi}_1 = x_{3n-10}\,, & \overline{\xi}_2 = -x_1\,, & \overline{\xi}_3 = x_1\,, & \overline{\xi}_i = x_1 x_{n-4+i} \quad \text{for } i = 4, \cdots, n-2 \\ \overline{\eta}_1 = 1\,, & \overline{\eta}_2 = 0\,, & \overline{\eta}_3 = x_1 x_{n-1}, & \overline{\eta}_i = x_1 x_{2n-9+i} \quad \text{for } i = 4, \cdots, n-2 \end{array} \tag{1.121}$$

This fixes all the spinor components but $\overline{\xi}_{n-1}$,$\overline{\xi}_n$,$\overline{\eta}_{n-1}$ and $\overline{\eta}_n$, and satisfies automatically all the definitions of the x_i. The remaining four spinor components are fixed by eq. (1.105).

In figure 1.4 a 6-point amplitude parametrization using Peraro's choice of variables is shown.



```
In[20]:= TwisTs[6, x]

Out[20]= {{{{1, 0}, {x[8], 1}}, {{0, -1}, {-1, x[8]}}},
          {{{0, 1}, {-x[1], 0}}, {{1, 0}, {0, -x[1]}}},
             1                                     1
          {{{────, 1}, {x[1], x[1] x[5]}}, {{1, - ────}, {-x[1] x[5], x[1]}}},
            x[1]                                  x[1]
             x[2]                                          x[2]
          {{{────, 1}, {x[1] x[6], x[1] x[7]}}, {{1, - ────}, {-x[1] x[7], x[1] x[6]}}},
             x[1]                                          x[1]
             x[3]           x[1] (-1 - x[2] x[6] + x[4] x[6] - x[8])
          {{{────, 1}, {────────────────────────────────────────────,
             x[1]                      x[3] - x[4]
             x[1] (-1 - x[5] + x[4] x[5] - x[2] x[7] + x[4] x[7])
             ─────────────────────────────────────────────────────}},
                               x[3] - x[4]
                  x[3]       x[1] (-1 - x[5] + x[4] x[5] - x[2] x[7] + x[4] x[7])
           {{1, - ────}, {- ─────────────────────────────────────────────────────,
                  x[1]                          x[3] - x[4]
             x[1] (-1 - x[2] x[6] + x[4] x[6] - x[8])
             ────────────────────────────────────────}}}},
                          x[3] - x[4]
             x[4]             -x[1] - x[1] x[2] x[6] + x[1] x[3] x[6] - x[1] x[8]
          {{{────, 1}, {- ────────────────────────────────────────────────────,
             x[1]                           x[3] - x[4]
              -x[1] - x[1] x[5] + x[1] x[3] x[5] - x[1] x[2] x[7] + x[1] x[3] x[7]
           - ─────────────────────────────────────────────────────────────────────}},
                                         x[3] - x[4]
                  x[4]       -x[1] - x[1] x[5] + x[1] x[3] x[5] - x[1] x[2] x[7] + x[1] x[3] x[7]
           {{1, - ────}, {─────────────────────────────────────────────────────────────────────,
                  x[1]                                x[3] - x[4]
              -x[1] - x[1] x[2] x[6] + x[1] x[3] x[6] - x[1] x[8]
           - ────────────────────────────────────────────────────}}}}}
                                 x[3] - x[4]
```

Figure 1.4: Another equivalent parametrization for a six-point function

1.4 Berends-Giele recurrence relation

The Berends-Giele recurrence relation [4] allows to generate tree-level amplitudes recursively in the number of legs. Even if one cannot simplify analytically the expressions obtained in this way, the recursive approach lends itself to efficient numerical evaluation.

We will be treating only the case of amplitudes involving solely gluons, but a generalization to theories with fermions is easily accomplished following exactly the same steps here presented.

First of all one has to introduce an auxiliary quantity with one leg off-shell, a current which we will denote $J^\mu(1, \cdots, n)$.

$J^\mu(1, \cdots, n) =$

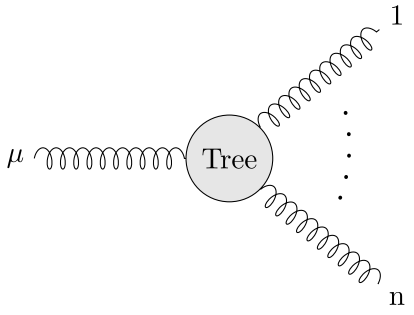


Figure 1.5: Graphical representation of the Berends-Giele current

J^μ is the sum of color-ordered $n+1$-point tree-level Feynman graphs, where legs $1, \cdots, n$ are on-shell gluons and leg μ is off-shell. Finally an appropriate off-shell propagator attached to the uncontracted μ leg is defined to be included in the current. Notice that since J^μ is an off-shell quantity it is gauge dependent, for example it depends on the reference momenta chosen for the polarization vectors, which thus must be kept fixed until an on-shell result has been extracted.

At this point the recursion relation is easily established. Being at tree-level and considering only gluons, if we follow the off-shell line μ back inside the graph there are only two possible scenarios, either we encounter a three-point or a four-point gluon vertex. Attached to these vertexes there will be subgraphs with exactly the same form as the initial J^μ we constructed but with a lower number of on-shell legs, see figure 1.6. Thus the n-point amplitude will be expressible in terms of a sum over all the possible lower-point currents contracted with the three and four-point vertex.

$$\begin{aligned} J^\mu(1, \cdots, n) = \frac{-i}{P^2_{1,n}} \Bigg[& \sum_{i=2}^{n-1} V_3^{\mu\nu\rho} J^\nu(1, \cdots, i) J^\rho(i+1, \cdots, n) \\ & + \sum_{i=2}^{n-2} \sum_{j=i+1}^{n-1} V_4^{\mu\nu\rho\sigma} J^\nu(1, \cdots, i) J^\rho(i+1, \cdots, j) J^\sigma(j+1, \cdots, n) \Bigg] \end{aligned} \tag{1.122}$$

where $P_{l,m} = \sum_{i=l}^{m} p_i$ and the V_i are the color-ordered gluon self-interactions 1.75 and 1.76.

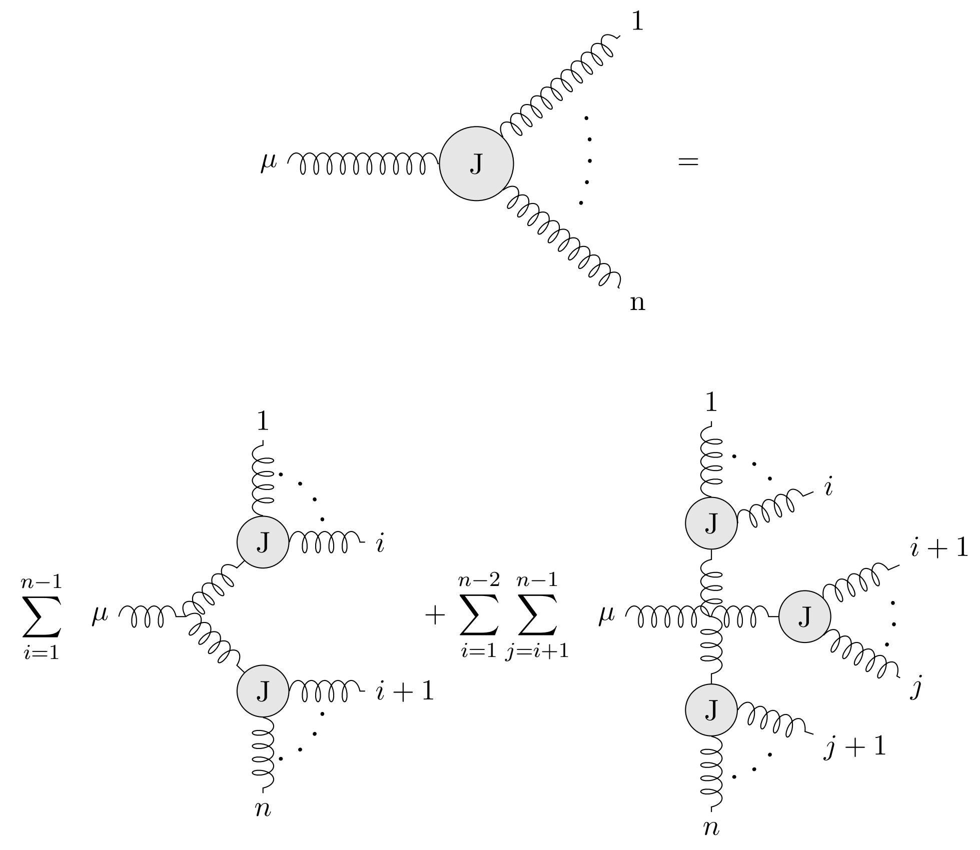


Figure 1.6: Graphical representation of the Berends-Giele recursion relation

The recursion terminates when only currents of the form $J^\mu(i)$, $i \in \{1, \cdots, n\}$ are left. By definition we set

$$J^\mu(i) \equiv \epsilon^\mu(p_i) \tag{1.123}$$

where $\epsilon^\mu(p_i)$ is the polarization vector associated to the i-th gluon.

Finally in order to get the $\mathbf{M}_{n+1}$ partial amplitude associated to J^μ one first amputates the off-shell propagator, then contracts with the appropriate polarization vector and takes the limit $P_{1,n}^2 = p_{n+1}^2 \to 0$.

Some propeties of the current J^μ

The Berends-Giele current satisfies the following identities:

- photon decoupling relation

$$J^\mu(1,2,3,\cdots,n) + J^\mu(2,1,3,\cdots) + \cdots + J^\mu(2,3,\cdots,n,1) = 0 \tag{1.124}$$

- reflection identity

$$J^\mu(1,2,\cdots,n) = (-1)^{n+1} J^\mu(n,\cdots,2,1) \tag{1.125}$$

- current conservation

$$P_{1,n}^\mu J_\mu(1,\cdots,n) = 0 \tag{1.126}$$

Equation 1.124 and 1.125 descend from the fact that gluonic currents and partial amplitudes can be obtained from one another, so they share these properties. [11] In order to prove eq. (1.126) one can proceed by induction. By definition $J(1) \cdot p_1 = 0$, and for a two-gluon current

$$\begin{aligned} P^\mu_{1,2} J_\mu(1,2) &= (p_1+p_2)^\mu \frac{-i}{P^2_{1,2}} V_{3\mu\nu\rho}(p_1,p_2)\epsilon_1^\nu \epsilon_2^\rho \\ &= \frac{1}{\sqrt{2}P^2_{1,2}}[\epsilon_1\cdot\epsilon_2(p_1^2-p_2^2)+2p_1\cdot\epsilon_2 p_2\cdot\epsilon_1 - p_2\cdot\epsilon_1 p_1\cdot\epsilon_2] \\ &= 0 \end{aligned} \tag{1.127}$$

having used the masslessness of gluons. Assuming now its validity for $n < m$ we prove the current conservation for $n = m$. We have that

$$P^\mu_{1,m} J_\mu(1,\cdots,m) = \frac{-i}{P^2_{1,m}}(A_1+A_2) \tag{1.128}$$

with

$$A_1 \equiv \sum_{i=1}^{m-1} \frac{i}{\sqrt{2}}(P^2_{1,i} - P^2_{i+1,m}) J(1,\cdots,i)\cdot J(i+1,\cdots,m)\,, \tag{1.129}$$

$$\begin{aligned} A_2 \equiv \sum_{i=1}^{m-2}\sum_{j=i+1}^{m-1} \frac{i}{2}\Big[&(P_{1,i}+P_{j+1,m})\cdot J(i+1,\cdots,j) J(1,\cdots,i)\cdot J(j+1,\cdots,m) \\ &- P_{i+1,m}\cdot J(1,\cdots,m) J(i+1,\cdots,j)\cdot J(j+1,\cdots,m) \\ &- P_{1,j}\cdot J(j+1,\cdots,m) J(1,\cdots,i)\cdot J(i+1,\cdots,j)\Big] \end{aligned} \tag{1.130}$$

where we already used eq. (1.126) for $n < m$. The term in eq. (1.130) arising from the four point vertex can be expressed in terms of three-point vertexes as

$$\begin{aligned} A_2 =& \\ &\sum_{i=1}^{m-2}\sum_{j=i+1}^{m-1} \frac{1}{\sqrt{2}}\Big[J_\mu(1,\cdots,i) V_3^{\mu\nu\rho}(P_{i+1,j},P_{j+1,m}) J_\nu(i+1,\cdots,j) J_\rho(j+1,\cdots,m) \\ &- J_\mu(j+1,\cdots,m) V_3^{\mu\nu\rho}(P_{1,i},P_{i+1,j}) J_\nu(1,\cdots,i) J_\rho(i+1,\cdots,j)\Big] \end{aligned} \tag{1.131}$$

Now we can see that

$$\begin{aligned} &\sum_{i=1}^{m-2}\sum_{j=i+1}^{m-1} V_3^{\mu\nu\rho}(P_{i+1,j},P_{j+1,m}) J_\nu(i+1,\cdots,j) J_\rho(j+1,\cdots,m) \\ &\qquad = iP^2_{i+1,m} J\mu(i+1,\cdots,m) \\ &\qquad - \sum_{k=i+1}^{m-2}\sum_{l=k+1}^{m-1} V_4^{\mu\nu\rho\sigma} J_\nu(i+1,\cdots,k) J_\rho(k+1,\cdots,l) J_\sigma(l+1,\cdots,m) \end{aligned} \tag{1.132}$$

[11] See eq. (1.73) and eq. (1.74)

and re-expressing the sum $\sum_{i=1}^{m-2}\sum_{j=i+1}^{m-1}$ as $\sum_{i=1}^{j-1}\sum_{j=2}^{m-1}$ in the second term of eq. (1.131) we have

$$\begin{aligned}\sum_{i=1}^{j-1} V_3^{\mu\nu\rho}(P_{1,i},P_{i+1,j})J_\nu(1,\cdots,i)J_\rho(i+1,\cdots,j)&\\ = iP_{i,j}^2 J^\mu(1,\cdots,j)&\\ -\sum_{k=1}^{j-2}\sum_{l=k+1}^{j-1} V_4^{\mu\nu\rho\sigma}J_\nu(1,\cdots,k)J_\rho(k+1,\cdots,l)J_\sigma(l+1,\cdots,j)& \qquad (1.133)\end{aligned}$$

Then substituting everything[12] back in A_2 we have that $A_2 = -A_1$ which completes the proof.

The auxiliary current J_a^μ

As already mentioned, once we computed the current J^μ, in order to get the amplitude we need to amputate the off-shell propagator, then contract with the appropriate polarization vector and take the limit $P_{1,n}^2 = p_{n+1}^2 \to 0$ which takes the leg μ on-shell:

$$\mathbf{M}_{n+1}(1,\cdots,n+1) = \lim_{P_{1,n}\to 0}\epsilon_\mu(n+1)J^\mu(1,\cdots,n)P_{1,n}^2 \qquad (1.134)$$

However, once we agree on the validity of the established recursive relation, in terms of a numerical evaluation there is no need to add the off-shell propagator $\frac{-i}{P_{1,n}^2}$. In fact it would only be a multiplicative factor which cancels when taking the limit to extract the amplitude from the current. Furthermore, that propagator is the only reason why the current needs to be off-shell, otherwise due to carrying momentum $P_{1,n}$ it would be singular. Thus we will omit that propagator, so we can take all momenta to be on-shell and no limit procedure is needed. So we will be considering two different currents, J^μ defined as in eq. (1.122), and another auxiliary current J_a^μ defined as in eq. (1.122) but without the $\frac{-i}{P_{1,n}^2}$ propagator. J^μ will be used through out the entire recursion, except for the last step where it will be substituted by J_a^μ. In other words, if we want to compute the amplitude $\mathbf{M}_{n+1}(1,\cdots,n+1)$, we define:

$$\mathbf{M}_{n+1}(1,\cdots,n+1) = \epsilon_\mu(n+1)J_a^\mu(1,\cdots,n) \qquad (1.135)$$

with

$$\begin{aligned}J_a^\mu(1,\cdots,n) = \Bigg[&\sum_{i=2}^{n-1} V_3^{\mu\nu\rho}J^\nu(1,\cdots,i)J^\rho(i+1,\cdots,n)\\ &+\sum_{i=2}^{n-2}\sum_{j=i+1}^{n-1} V_4^{\mu\nu\rho\sigma}J^\nu(1,\cdots,i)J^\rho(i+1,\cdots,j)J^\sigma(j+1,\cdots,n)\Bigg] \qquad (1.136)\end{aligned}$$

1.5 Britto-Cachazo-Feng-Witten recurrence relation

The Britto-Cachazo-Feng-Witten (BCFW) recursion [9] is a tree-level on-shell recurrence relation which allows the complete factorization of n-point amplitudes into three-point on-shell amplitudes and propagators. This is made possible by the analytic continuation of the

[12]Notice that the double sum term in eq. (1.132) and eq. (1.133) is absent respectively for $i = m-2$ and $j = 2$.

amplitudes to complex momenta: massless onshell three-point amplitudes are identically zero, but through complex kinematics they become nonvanishing. This in turn can be achieved by considering the tree-level color-ordered amplitudes as analytic functions of the angle brackets and square brackets, basing on an original idea of Cachazo, Svrcek and Witten [11]

So first we will focus on the three-point amplitude and complex kinematics, turning then to the recurrence relation itself. We will be concerned with all gluon amplitudes only, but the same arguments with slight modifications leads to a generalization for theories including scalar and spin $\frac{1}{2}$ particles as well.

1.5.1 Three-point amplitudes and complex kinematics

Consider first of all writing an on-shell three-point scattering amplitude in terms of the massless momenta p_1, p_2, p_3, supposing these to be real, as of course they should in order to have physical meaning. Because of momentum conservation $p_1 + p_2 + p_3 = 0$[13] we have:

$$s_{ij} \equiv (p_i + p_j)^2 = 0 \tag{1.137}$$

for any choice of i,j as can be easily seen taking for example $i = 1$ and $j = 2$:

$$s_{12} = (p_1 + p_2)^2 = (-p_3)^2 = 0 \tag{1.138}$$

This is also equivalent to:

$$p_1 \cdot p_2 = \frac{1}{2}(p_1^2 + 2p_1 \cdot p_2 + p_2^2) = \frac{1}{2}s_{12} = 0 \tag{1.139}$$

So apparently there aren't any non-vanishing invariants for the amplitude to depend on, so we expect it to be zero.

Consider then the amplitude to be an analytic function of the angle and square brackets. All possible helicity configurations lead to MHV amplitudes, since only three particles are involved. Take then equations eq. (1.79a) and write:

$$i\mathbf{M}(1^-, 2^-, 3^+) = i\frac{\langle 1|2\rangle^4}{\langle 1|2\rangle\langle 2|3\rangle\langle 3|1\rangle} \tag{1.140}$$

$$i\mathbf{M}(1^+, 2^+, 3^-) = -i\frac{[1|2]^4}{[1|2][2|3][3|1]} \tag{1.141}$$

The on-shell condition implies:

$$0 = s_{ij} = \langle i|j\rangle[j|i] \tag{1.142}$$

So either $\langle i|j\rangle = 0$ or $[j|i] = 0$, but because of:

$$\langle i|j\rangle = [j|i]^* \tag{1.143}$$

both need to be true. Thus as expected eq. (1.140) and eq. (1.141) both vanish.

The key point is now to allow the external momenta to be complex. If we do so eq. (1.143) does not hold any more. This means that if we take $[i|j] = 0$ then $\langle i|j\rangle$ is allowed to be non-zero. In fact now it is possible to choose all three left-handed spinors

[13]We are considering all particles as outgoing.

to be proportional: $|1] = c_1|3]$, $|2] = c_2|3]$ and from momentum conservation one gets $c_1|1\rangle + c_2|2\rangle + |3\rangle = 0$. Then:

$$[1|2] = [2|3] = [3|1] = 0 \tag{1.144}$$

while $\langle 1|2\rangle$, $\langle 2|3\rangle$ and $\langle 3|1\rangle$ are all non-vanishing. This means that now eq. (1.140) is nonsingular even though all the invariants s_{ij} still vanish, which is equivalent to all three particles beeing still on-shell.

There is a class of conjugate momenta for which instead:

$$\langle 1|2\rangle = \langle 2|3\rangle = \langle 3|1\rangle = 0 \tag{1.145}$$

with $[1|2]$, $[2|3]$ and $[3|1]$ all non-vanishing and then of course eq. (1.141) becomes non-singular. If either of the amplitudes of eq. (1.140) or eq. (1.141) appears in the "wrong" kinematics, so respectively when eq. (1.145) or eq. (1.144) apply, then it should be set to zero because more vanishing spinor products appear in the numerator than in the denominator [5]. In the case of on-shell complex momenta, the all-plus as well as the all-minus amplitudes still vanish. Again this can be seen by choosing all reference momenta for the polarization vectors to be equal.

Given these building blocks we can now turn to the BCFW recurrence.

1.5.2 Complex kinematics and factorization

The strategy we are going to present is based on [9].

Consider a color-ordered amplitude $iM(1,\cdots,n)$ and choose two legs i, j. Then take the complex variable z and define the shifted angle and square brackets:[14]

$$\begin{aligned} |\hat{i}\rangle &= |i\rangle & |\hat{i}] &= |i] + z|j] \\ |\hat{j}\rangle &= |j\rangle - z|i\rangle & |\hat{j}] &= |j] \end{aligned} \tag{1.146}$$

From these we get the associated momenta:

$$\begin{aligned} \hat{p}_i^\mu &= p_i^\mu + \frac{1}{2} z\langle i|\sigma^\mu|j] \\ \hat{p}_j^\mu &= p_j^\mu - \frac{1}{2} z\langle i|\sigma^\mu|j] \end{aligned} \tag{1.147}$$

Notice that $\hat{p}_i + \hat{p}_j = p_i + p_j$ so momentum conservation is respected, moreover $p_i^2 = 0 = p_j^2$ so both particles are still on-shell.[15] The amplitude is now a function of this complex variable z, $i\mathbf{M}(z) = i\mathbf{M}(1,\cdots,\hat{i}(z),\cdots,\hat{j}(z),\cdots,n)$. Consider then the integral

$$\frac{1}{2\pi i}\oint_{C_R} \frac{i\mathbf{M}(z)dz}{z} \tag{1.148}$$

taken around a large circle C_R of radius R in the complex z plane. As known from complex analysis as $R = |z| \to \infty$ eq. (1.148) equals to the sum of the residues of all the poles in the complex z plane of the integrand function, including the one in $z = \infty$. Moreover the value of this sum is zero. Stated differently

$$\operatorname*{\mathbf{Res}}_{z=0} \frac{i\mathbf{M}(z)}{z} = -\sum_{poles\ \alpha} \operatorname*{\mathbf{Res}}_{z=z_\alpha} \frac{i\mathbf{M}(z)}{z} - \operatorname*{\mathbf{Res}}_{z=\infty} \frac{i\mathbf{M}(z)}{z} \tag{1.149}$$

[14]From here on we are going to use the hat for shifted quantities which carry a z dependence.

[15]To prove this the Fierz rearrangement eq. (1.41) and antisymmetry of the spinor product is used.

with the sum running over all the poles α at finite values of z excluding the one in $z = 0$ which was explicitly isolated. Notice that the residue in $z = 0$ is none other than the physical amplitude $i\mathbf{M}(z = 0)$.

If now

$$\frac{i\mathbf{M}(z)}{z} \xrightarrow[|z|\to\infty]{} 0 \qquad \Rightarrow \qquad \underset{z=\infty}{\mathbf{Res}} \frac{i\mathbf{M}(z)}{z} = 0 \tag{1.150}$$

and we get the simple relation

$$i\mathbf{M}(z = 0) = -\sum_{poles\ \alpha} \underset{z=z_\alpha}{\mathbf{Res}} \frac{i\mathbf{M}(z)}{z} \tag{1.151}$$

So, provided eq. (1.150) holds, if we are able to locate all of the poles and evaluate their residues we will immediately get the value of the physical amplitude. It can be shown that given a certain helicity configuration it is always possible to choose the particles i and j whose spinor components are to be shifted such that eq. (1.150) is true. For the proof of this fact see section 1.5.3. As long as we are not concerned with this matter we are not going to assign explicitly the helicity of the external particles, as done so far.

Tackling first the issue of where the poles are, the successive evaluation of the residues will be straight forward. Since we are dealing with tree-level amplitudes the only place were poles can come up are the denominators of the propagators. This means that what we are interested in are the values of z for which intermediate states go on-shell and these denominators vanish.

Consider a propagator carrying momentum Q, we can write the amplitude with respect to this propagator as [16]

$$i\mathbf{M}(z) = i\widetilde{\mathbf{M}}^{\alpha}_{L}(b+1, \cdots, \hat{i}(z), \cdots, a-1)\, \frac{-i\eta_{\alpha\beta}}{\hat{Q}(z)^2}\, i\widetilde{\mathbf{M}}^{\beta}_{R}(a, \cdots, \hat{j}(z), \cdots, b) \tag{1.152}$$

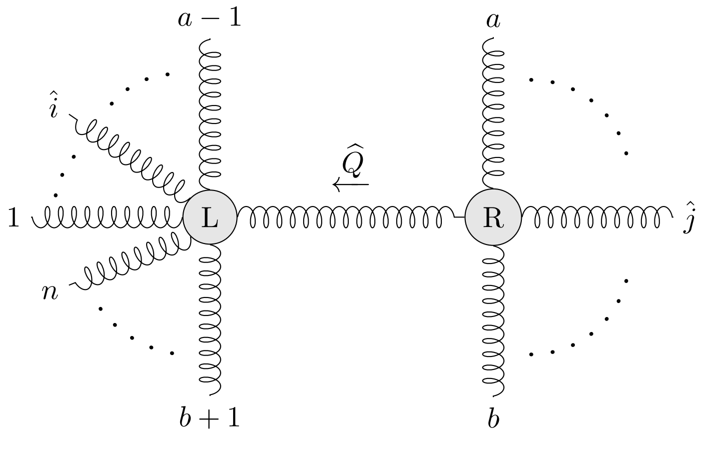


Figure 1.7: Posible factorization of one of the terms in BCFW recursion

as depicted more diagrammatically in figure 1.7. The tilde above $\mathbf{M}^{\alpha}_{L}$ and $\mathbf{M}^{\beta}_{R}$ is there to remind us that these are just some subdiagrams of the complete amplitude. They are not proper amplitudes if considered on their own, or at least not yet.

Lets compute the value z^* for which this propagator leads to a (simple) pole in the amplitude. Defining $Q = -(p_{b+1} + \cdots + p_{a-1})$ we have

[16]We are using the Feynman gauge for the propagator.

$$\begin{aligned}\hat{Q}(z)^2 &= (-p_{b+1}\cdots - \hat{p}_i \cdots - p_{a-1})^2 \\ &= (Q - \frac{1}{2}z\langle i|\sigma|j])^2 \\ &= Q^2 - z\langle i|Q_{1\cdots l}|j] \\ &\overset{!}{=} 0\end{aligned} \tag{1.153}$$

where we used the Fierz identity eq. (1.41) and the antisymmetry of the angle-angle and bracket-bracket product which lead to cancellation of the term proportional to z^2. And so we get

$$z^* = \frac{Q^2}{\langle i|Q|j]} \tag{1.154}$$

The associated residue can now easily be computed using the general expression

$$\underset{z=z^*}{\mathbf{Res}}\, f(z) = \frac{1}{(n-1)!}\lim_{z\to z^*}\left[\frac{d^{n-1}}{dz^{n-1}}(z-z^*)^n f(z)\right] \tag{1.155}$$

where n is the order of the pole in z^*. In our case $n = 1$, so we get

$$\underset{z=z^*}{\mathbf{Res}}\, \frac{i\mathbf{M}(z)}{z} = \lim_{z\to z^*}(z-z^*)\frac{1}{z}i\widetilde{\mathbf{M}}^{\alpha}_L(z)\,\frac{-i\eta_{\alpha\beta}}{\hat{Q}^2(z)}\,i\widetilde{\mathbf{M}}^{\beta}_R(z) \tag{1.156}$$

$$= \lim_{z\to z^*}\left[\frac{z\langle i|Q|j] - Q^2}{\langle i|Q|j]}\right]\frac{1}{z}i\widetilde{\mathbf{M}}^{\alpha}_L(z)\,\frac{-i\eta_{\alpha\beta}}{\hat{Q}^2(z)}\,i\widetilde{\mathbf{M}}^{\beta}_R(z) \tag{1.157}$$

$$= -\frac{1}{z^*}i\widetilde{\mathbf{M}}^{\alpha}_L(z^*)\,\frac{-i\eta_{\alpha\beta}}{\langle i|Q|j]}\,i\widetilde{\mathbf{M}}^{\beta}_R(z^*) \tag{1.158}$$

$$= i\widetilde{\mathbf{M}}^{\alpha}_L(z^*)\,\frac{i\eta_{\alpha\beta}}{Q^2}\,i\widetilde{\mathbf{M}}^{\beta}_R(z^*) \tag{1.159}$$

where in eq. (1.157) and eq. (1.159) we substituted the value of z^*. Using then the completeness relation eq. (1.52) in the form

$$-\eta^{\mu\rho} = \epsilon^{\mu}_{+}(k,q)\epsilon^{\rho}_{-}(k,q) + \epsilon^{\mu}_{-}(k,q)\epsilon^{\rho}_{+}(k,q) - \frac{q^{\rho}k^{\mu} + q^{\mu}k^{\rho}}{q\cdot k} \tag{1.160}$$

where q is some reference momentum, we can substitute the metric tensor with a combination of polarization vectors plus a term proportional to k^{μ}. Since the internal momentum $\hat{Q}(z^*)$ is on-shell we have that

$$\widetilde{\mathbf{M}}^{\alpha}_L(b+1,\cdots,\hat{i}(z^*),\cdots,a-1)\epsilon_{\alpha}(\hat{Q}(z^*),q) = \mathbf{M}(b+1,\cdots,\hat{i}(z^*),\cdots,a-1,\hat{Q}(z^*)) \tag{1.161}$$

$$\epsilon_{\beta}(\hat{Q}(z^*),q)\widetilde{\mathbf{M}}^{\beta}_R(a,\cdots,\hat{j}(z^*),\cdots,b) = \mathbf{M}(\hat{Q}(z^*),a,\cdots,\hat{j}(z^*),\cdots,b) \tag{1.162}$$

are proper on-shell amplitudes. Moreover the contraction of k with both $\widetilde{M}_L$ and $\widetilde{M}_R$ vanishes due to Ward identity

$$k_{\mu}\widetilde{\mathbf{M}}^{\mu}_R(a,\cdots,\hat{j}(z^*),\cdots,b) = 0 \tag{1.163}$$

$$k_{\mu}\widetilde{\mathbf{M}}^{\mu}_L(b+1,\cdots,\hat{i}(z^*),\cdots,a-1) = 0 \tag{1.164}$$

This leads to the final form of the residue in z^*

$$\underset{z=z^*}{\mathbf{Res}} \frac{i\mathbf{M}(z)}{z} = $$

$$i\mathbf{M}(1,\cdots,\hat{i}(z^*),\cdots,l,\hat{Q}^+(z^*))\frac{i}{Q^2}i\mathbf{M}(\hat{Q}^-(z^*),l+1,\cdots,\hat{j}(z^*),\cdots,n)$$
$$+\, i\mathbf{M}(1,\cdots,\hat{i}(z^*),\cdots,l,\hat{Q}^-(z^*))\frac{i}{Q^2}i\mathbf{M}(\hat{Q}^+(z^*),l+1,\cdots,\hat{j}(z^*),\cdots,n) \quad (1.165)$$

As can be see a factorization of the n-point amplitude into lower point amplitudes happened. Equation (1.165) is just one contribution to the complete amplitude. Considering all the poles arising from the denominators and the associated residues we have

$$i\mathbf{M}(z=0) = \sum_{a,b}\Bigg[i\mathbf{M}(b+1,\cdots,a-1,\hat{Q}^+(z_{a,b}))\frac{-i}{Q^2}i\mathbf{M}(\hat{Q}^-(z_{a,b}),a,\cdots,b)$$
$$+\, i\mathbf{M}(b+1,\cdots,a-1,\hat{Q}^-(z_{a,b}))\frac{-i}{Q^2}i\mathbf{M}(\hat{Q}^+(z_{a,b}),a,\cdots,b)\Bigg] \quad (1.166)$$

with $z_{a,b}$ value of the pole associated to a given choice of a and b which identify uniquely a certain propagator. If $\hat{i}$ and $\hat{j}$ are on the same side of the propagator, Q carries no z dependence since no complex momentum "flows" through this propagator and so this channel gives no contribution to the amplitude being its residue zero.

In the end we are left with the amplitude expressed in terms of products of lower point amplitudes, which can now be computed in a completely independent way. It is possible to keep applying the BCFW mechanism to each of this lower point amplitudes, until we end up with a sum of products of three-point amplitudes and propagators. Being complex kinematics involved these three-point amplitudes will be non vanishing and easily computable as discussed in section 1.5.1.

1.5.3 Helicity configurations and border condition

In order to complete the computation and evaluate the three-point amplitudes appearing in the final factorization of the BCFW recursion, we need to reintroduce the helicity labels suppressed so far. These are important also for another reason: eq. (1.150) always holds if the shift of the spinor components is performed in the appropriate way, and this is determined by the helicity configuration of the particles involved in the shift.

We want to study the behaviour of the amplitude in the limit $z \to \infty$. The z dependence of any diagram is relatively simple, if we shift the two external legs (i,j) we can follow the flow of the complex momentum from leg i to leg j through the diagram. Thus only the two polarization vectors $\epsilon(i,r)$ and $\epsilon(j,s)$, with r,s reference vectors, and the propagators and vertexes interested by this momentum flow contribute to the $z \to \infty$ limit, see figure 1.8.

Using the shift of eq. (1.146) the complex part of the momenta is given by $zl^\mu \equiv z\frac{1}{2}\langle i|\sigma^\mu|j]$, as displayed in eq. (1.180). Since $l^2 = 0$, given any generic q momentum representing the real part of the propagator

$$(q+zl)^2 = q^2 + z\langle i|q|j] \quad (1.167)$$

so the propagators will go as $\frac{1}{z}$, and recalling that the three-point vertexes depend linearly on the momentum they will go as z. Notice that there will always be exactly one z

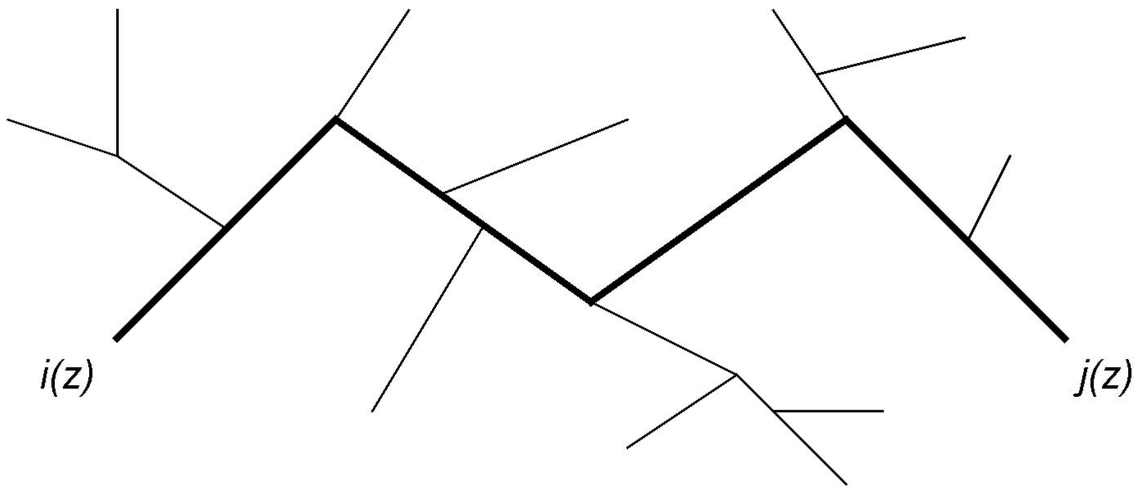


Figure 1.8: A possible graph in the $z \to \infty$ limit, the thicker line represents the complex momentum flow

dependent vertex more than the z dependent propagators,[17] thus all of these together will contribute as $\mathcal{O}(z)$. The dependence of the polarization vectors on z is related to the helicities of the particles. Being the reference vectors arbitrary we may choose them as we please provided the polarization vector does not become singular. Exploiting this fact we take:

$$\begin{aligned} \epsilon_+(\hat{i}, r) &\propto \frac{\langle r|\gamma|\hat{i}]}{\langle r|i\rangle} = \frac{\langle r|\gamma|(|i] + z|j])}{\langle r|i\rangle} \\ &= \frac{\langle j|\gamma|i] + z\langle j|\gamma|j]}{\langle j|i\rangle} \propto \mathcal{O}(z) \end{aligned} \tag{1.168}$$

$$\begin{aligned} \epsilon_-(\hat{i}, r) &\propto \frac{\langle \hat{i}|\gamma|r]}{[\hat{i}|r]} = \frac{\langle i|\gamma|r]}{([i|+z[j|)|r]} \\ &= \frac{\langle i|\gamma|i]}{z[j|i]} \propto \mathcal{O}\left(\frac{1}{z}\right) \end{aligned} \tag{1.169}$$

$$\begin{aligned} \epsilon_+(\hat{j}, s) &\propto \frac{\langle s|\gamma|\hat{j}]}{\langle s|\hat{j}\rangle} = \frac{\langle s|\gamma|j]}{\langle s|(|j\rangle - z|i\rangle)} \\ &= -\frac{\langle j|\gamma|j]}{z\langle j|i\rangle} \propto \mathcal{O}\left(\frac{1}{z}\right) \end{aligned} \tag{1.170}$$

$$\begin{aligned} \epsilon_-(\hat{j}, s) &\propto \frac{\langle \hat{j}|\gamma|s]}{[\hat{j}|s]} = \frac{(\langle j|-z\langle i|)|\gamma|s]}{[j|s]} \\ &= \frac{\langle j|\gamma|i] - z\langle i|\gamma|i]}{[j|i]} \propto \mathcal{O}(z) \end{aligned} \tag{1.171}$$

where we chose $r, s = i$ in eq. (1.169) and eq. (1.171), whereas $r, s = j$ in eq. (1.168) and eq. (1.170).

[17]This can be seen analysing the final completely factorized expression of the amplitude.

As can be seen the helicity configuration $(i,j) = (-,+)$ leads to a z dependence of the whole amplitude of the form $\mathcal{O}(\frac{1}{z})$. Considering the case $(i,j) = (-,-)$ instead we would have $\mathbf{M}$ of order $\mathcal{O}(z)$, however notice that

$$\epsilon_-(\hat{i}) \cdot \epsilon_-(\hat{j}) = 0\,, \qquad l \cdot \epsilon_-(\hat{i}) = l \cdot \epsilon_-(\hat{j}) = 0 \tag{1.172}$$

thus similar terms cannot even arise. The first non-zero contraction is given by $\epsilon_-(\hat{i})$ and $\epsilon_-(\hat{j})$ both dotted into an order $\mathcal{O}(1)$ term, i.e. a real part of the momentum. But these terms are down by a factor of two in z leading again to an order $\mathcal{O}(\frac{1}{z})$ dependence. A similar reasoning applies to the $(+,+)$ configuration.

The only truly problematic term is the $(+,-)$ one, which is irremediably of order $\mathcal{O}(z^3)$. Thus in the case we found such a helicity configuration the shift we chose is not suitable. In such a situation one can simply invert the roles of i and j getting a new shift that cancels the border term as desired.

1.5.4 A possible momentum shift convention

Summing up, referring to figure 1.9, one may choose the following shift convention:

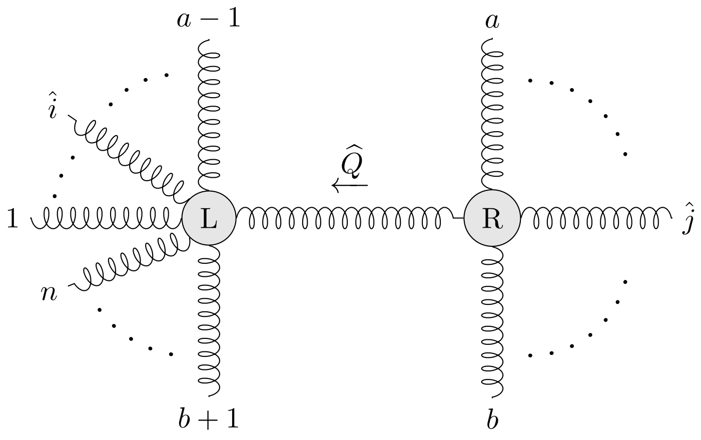


Figure 1.9: Generic term in the BCFW recurrence factorization.

$$Q_{k,m} \equiv -(p_k + p_{k+1} + \cdots + p_{m-1} + p_m) \tag{1.173}$$

$(i,j) = (-,+), (-,-), (+,+)$:

$$\begin{aligned} |\hat{i}\rangle &= |i\rangle & \qquad |\hat{i}] &= |i] + z|j] \\ |\hat{j}\rangle &= |j\rangle - z|i\rangle & \qquad |\hat{j}] &= |j] \end{aligned} \tag{1.174}$$

The pole is then located at

$$z^* = \frac{Q^2_{b+1,a-1}}{\langle i|Q_{b+1,a-1}|j]} \tag{1.175}$$

defining

$$l^\mu(z^*) \equiv \frac{1}{2} z^* \langle i|\sigma^\mu|j] \tag{1.176}$$

For the last possible helicity configuration $(i,j) = (+,-)$ instead

$$\begin{aligned} |\hat{i}\rangle &= |i\rangle - z|j\rangle & \qquad |\hat{i}] &= |i] \\ |\hat{j}\rangle &= |j\rangle & \qquad |\hat{j}] &= |j] + z|i] \end{aligned} \tag{1.177}$$

$$z^* = -\frac{Q^2_{b+1,a-1}}{\langle j|Q_{b+1,a-1}|i]} \tag{1.178}$$

defining

$$l^\mu(z^*) \equiv -\frac{1}{2} z^* \langle j|\sigma^\mu|i] \tag{1.179}$$

The shifted momenta are given by

$$\begin{aligned} \hat{p}^\mu_i &= p^\mu_i + l^\mu(z^*) \\ \hat{p}^\mu_j &= p^\mu_j - l^\mu(z^*) \\ \hat{Q}_{b+1,a-1} &= Q_{b+1,a-1} - l^\mu(z^*) \end{aligned} \tag{1.180}$$

We assigned a specific momentum shift also to the configurations $(+,+)$ and $(-,-)$, but as shown in the preceding section they may be shifted in either way, in none of the two situations border terms arise.

Chapter 2

Functional reconstruction and finite fields

In this and the next chapter we are going to present a Mathematica language implementation of the functional reconstruction over finite fields and its application to tree-level scattering amplitudes.

2.1 Reconstruction of rational functions

Now we are going to talk about the reconstruction of rational functions. This topic can be divided in two relevant cases, the univariate case and the multivariate case. In our work so far we implemented a code only for the former one leaving the latter to a later stage of our studies, since an implementation of multivariate reconstruction is technically more involved and outside the scope of this thesis.

What we are going to say is true for any kind of rational function and does not need any modification in order to be applied to scattering amplitudes.

2.1.1 Univariate polynomials

Before dealing with univariate rational functions we start with the reconstruction of univariate polynomials, since the idea behind the two techniques is pretty much the same.

Consider the unknown polynomial $G(z)$ whose analytic form

$$G(z) = c_0 + c_1 z + \cdots + c_R z^R \tag{2.1}$$

we want to determine. The only information we have is an algorithm that takes as input a numerical value of the variable $z = z^*$ and returns as output the value $G(z^*)$: this is usually called the black-box interpolation problem. [1] If we knew the degree R of the polynomial we could get its analytic form simply by solving a system of $R+1$ equations in the $R+1$ variables $\{a_0, a_1, \cdots, a_R\}$:

$$\begin{cases} c_0 + c_1 z_1 + \cdots + c_R z_1^R = G(z_1) \\ \quad \vdots \\ c_0 + c_1 z_R + \cdots + c_R z_R^R = G(z_R) \\ c_0 + c_1 z_{R+1} + \cdots + c_R z_{R+1}^R = G(z_{R+1}) \end{cases} \tag{2.2}$$

[1]From here on we will often say "evaluate G at z^*", what we actually mean by that is to run the above mentioned algorithm, which in our specific case will be either BCFW or Berends-Giele recursion, for the given value z^*.

However there is no way of knowing beforehand the degree R. Moreover even if there was one, despite being efficient for simple functions depending on one or two variables, this method has a bad scaling behaviour when increasing the number n of variables involved and the total degree. In fact write the individual monomials as strings of $R+n$ terms with n multiplications $\times$ and supply the missing powers of the variables with 1's. For example for $R = 12$, $n = 5$ a possible term is

$$z_1^3 z_2^2 z_3 z_4^2 z_5 = 111 \times z_1 z_1 z_1 \times z_2 z_2 \times z_3 \times z_4 z_4 \times z_5 \tag{2.3}$$

Then it is easy to see that each monomial is uniquely defined by the position of the multiplications $\times$, so the total possible monomials in a polynomial of total degree R and n variables is given by $N = \binom{R+n}{R}$. It is thus evident how the dimension of the system grows fast. Moreover solving an $N \times N$ dense system of linear equations is an $\mathcal{O}(N^3)$ operation, thus this method becomes rather quickly unefficient.

Newton's polynomial representation

A simple reconstruction method for univariate polynomials is based on Newton's polynomial representation. One may rewrite eq. (2.1) as

$$\begin{aligned} G(z) &= \sum_{i=0}^{R} a_i \prod_{j=0}^{i-1} (z - y_j) \\ &= a_0 + (z - y_0)\Big(a_1 + (z - y_1)(a_2 + (z - y_2)(\cdots + (z - y_{R-1})a_R))\Big) \end{aligned} \tag{2.4}$$

Through this differently factorized form we can get the coefficients a_i simply by evaluating the function at the y_i:

$$\begin{aligned} a_0 &= G(y_0) \\ a_1 &= \frac{G(y_1) - a_0}{y_1 - y_0} \\ a_2 &= \left((G(y_2) - a_0)\frac{1}{y_2 - y_0} - a_1\right)\frac{1}{y_2 - y_1} \\ &\vdots \\ a_R &= \left(((G(y_R) - a_0)\frac{1}{y_R - y_0} - a_1)\frac{1}{y_R - y_1} - \cdots - a_{R-1}\right)\frac{1}{y_R - y_{R-1}} \end{aligned} \tag{2.5}$$

The values of the y_i are arbitrary and basing on the choice of these we get different values for the $a_i(y_0, y_1, \cdots, y_{i-1})$ defined through eq. (2.5). Once the sequence of y_i is fixed the Newton form of the polynomial is unique. Notice that each a_i also depends on a_j but only with $j < i$, thus each time a new coefficient is evaluated none of the before computed ones needs to be changed. This clearly differs from the standard representation and is made possible by the fact that the r-th term of the sum in eq. (2.4) contributes to all the powers of z from zero up to r. This is ideal for the case where the total rank of the polynomial is unknown a priori, and may be considered the main reason for preferring this method over other alternatives. Computing the a_i as in eq. (2.5) at some value $i = R$ we will get that the coefficients start evaluating to zero. After getting a sufficiently large set of consecutive vanishing coefficients one may assume that R is the degree of the polynomial and end the computation. Notice that, because of how each term in the summation of Newtons form contributes to the single monomials appearing in the canonical form, eq. (2.1), even if the latter presents several vanishing entries, the former will in general present non-vanishing entries. This fact makes the termination criterion described above robust, and an incorrect determination of G is highly unlikely. Equivalently one may evaluate the reconstructed function $G'(z)$ and the black-box algorithm corresponding to $G(z)$ at some z each time a new coefficient a_i is computed. If the two values match[2] then $i = R$ and we stop the computation. Else we compute a new coefficient and keep going until the test evaluations match each other.

The transposition from Newton's form to the canonical one is simple, one only needs to perform all the multiplications and then sum the like terms. However the analytic form of the c_i expressed in terms of the a_i is not trivial at all.

Example 1

Consider the following function to be reconstructed using Newton's polynomial representation:

$$G(x) = 1 + 2x + 3x^2 + 4x^5 \tag{2.6}$$

We proceed by comparing the test function $ftest(x^*)$ with $G(x^*)$ after computing each of the single a_i coefficients, if they agree (on different x^*) we stop the computation.

First consider the arbitrary y_i as successive numbers starting with 1, then we get:

$$\begin{aligned}
&y_0 = 1\\
&a_0 = 10\\
&ftest(x) = 10\\
&y_1 = 2\\
&a_1 = 135\\
&ftest(x) = 10 + 135(-1 + x)\\
&y_2 = 3\\
&a_2 = 363\\
&ftest(x) = 10 + (135 + 363(-2 + x))(-1 + x)\\
&y_3 = 4\\
&a_3 = 260\\
&ftest(x) = 10 + (135 + (363 + 260(-3 + x))(-2 + x))(-1 + x)\\
&y_4 = 5\\
&a_4 = 60\\
&ftest(x) = 10 + (135 + (363 + (260 + 60(-4 + x))(-3 + x))(-2 + x))(-1 + x)\\
&y_5 = 6\\
&a_5 = 4\\
&ftest(x) = 10 + (135 + (363 + (260 + (60 + 4(-5 + x))(-4 + x))(-3 + x))(-2 + x))\times\\
&\qquad\qquad (-1 + x)
\end{aligned}$$

and performing all the multiplications and summations this yields the canonical form

$$ftest(x) = 1 + 2x + 3x^2 + 4x^5 \tag{2.7}$$

[2]Of course if the evaluation of G and G' matches for the tested value of z one checks the correspondence also on several other values in order to be sure it was not a mere coincidence.

which is the sought-for polynomial.

Example 2

Consider now the same function as before but different sampling points y_i: this will affect the a_i and thus the intermediate and final expressions of $ftest$ in Newton's form, but of course at the end of the computation the canonical form will be just the same. Consider for example taking y_i as multiples of 7:

$$
\begin{aligned}
&y_0 = 7\\
&a_0 = 67390\\
&ftest(x) = 67390\\
&y_1 = 14\\
&a_1 = 297789\\
&ftest(x) = 67390 + 297789(-7+x)\\
&y_2 = 21\\
&a_2 = 123483\\
&ftest(x) = 67390 + (297789 + 123483(-14+x))(-7+x)\\
&y_3 = 28\\
&a_3 = 12740\\
&ftest(x) = 67390 + (297789 + (123483 + 12740(-21+x))(-14+x))(-7+x)\\
&y_4 = 35\\
&a_= 420\\
&ftest(x) = 67390 + (297789 + (123483 + (12740 + 420(-28+x))(-21+x))(-14+x))\times\\
&\qquad\qquad (-7+x)\\
&y_5 = 42\\
&a_5 = 4\\
&ftest(x) = 67390 + (297789 + (123483 + (12740 + (420 + 4(-35+x))(-28+x))(-21+x))\times\\
&\qquad\qquad (-14+x))(-7+x)
\end{aligned}
$$

which in canonical form is again equal to eq. (2.7).

2.1.2 Univariate rational functions

Consider now the black-box interpolation problem for $G(z)$ univariate rational function

$$G(z) = \frac{n_0 + n_1 z + \cdots + n_R z^R}{d_0 + d_1 z + \cdots d_{R'} z^{R'}} \tag{2.8}$$

with the degrees of numerator and denominator, R and R', unknown.

Thiele's interpolation formula

As for the univariate polynomial there is an alternative form for $G(z)$ called Thiele's interpolation formula, which expresses it as a continued fraction:

$$\begin{aligned} G(z) &= a_0 + \cfrac{z-y_0}{a_1 + \cfrac{z-y_1}{a_2 + \cfrac{z-y_2}{\cfrac{\vdots}{a_{N-1} + \cfrac{z-y_{N-1}}{a_N}}}}} \\ &= a_0 + (z-y_0)\left(a_1 + (z-y_1)\left((z-y_2)\left(\cdots + \frac{z-y_{N-1}}{a_N}\right)^{-1}\right)^{-1}\right)^{-1} \end{aligned} \tag{2.9}$$

The y_i are just arbitrary rationals, and again the a_i depend on the coefficients already computed (as well as on the y_j with $j \leq i$):

$$\begin{aligned} a_0 &= G(y_0) \\ a_1 &= (G(y_1) - a_0)^{-1}(y_1 - y_0) \\ &\vdots \\ a_r &= \left(\left((G(y_N) - a_0)^{-1}(y_N - y_0) - a_1\right)^{-1}(y_N - y_1) - \cdots - a_{N-1}\right)^{-1}(y_N - y_{N-1}) \end{aligned} \tag{2.10}$$

As in the univariate polynomial case no previous knowledge about R and R' is needed, however things become a little more involved in this case because of the form of Thiele's formula. Being G expressed as a continued fraction many spurious singularities arise, i.e. singularities which are only present due to the specific representation chosen and do not appear in the canonical form. These singularities are present in eq. (2.10) as well. Thus it may happen that at a certain step j of the computation of the coefficients an y_j is chosen whose value is such that a_j diverges. In such a case we simply discard that problematic value of y and compute a_j through a different y_j, which is perfectly fine because as we said the choice of any single y is arbitrary.

The termination criterion will be given again by the coincidence of the reconstructed function G' and the black-box algorithm when evaluated at several different values of z.

Notice that each time we compute a new coefficient a_j the reconstructed function G' gains a new power of z either in the numerator (if j is odd) or in the denominator (if j is even). So in the Thiele form we will always have that the degree of the numerator R coincides with that of the denominator R' or that $R = R' + 1$. However their actual degree may be whatever, this means that some, or potentially many, of the higher degree coefficients n_j and d_j are vanishing. In other words part of our efforts were devoted to reconstruct zeros. This is why if the degrees R and R' were known a priori, using the system solving approach may be preferred since it would involve a lower number of evaluations of the function G.

Example

Consider the following function:

$$G(x) = \frac{1 + 2x + 3x^2}{9 - x^2} \tag{2.11}$$

using again consecutive numbers for the y_i we have:

$$
\begin{aligned}
y_0 &= 1\\
a_0 &= \frac{3}{4}\\
ftest(x) &= \frac{3}{4}\\
y_1 &= 2\\
a_1 &= \frac{20}{53}\\
ftest(x) &= \frac{3}{4} + \frac{53}{20}(-1+x)\\
y_2 &= 4\\
a_2 &= -\frac{4399}{1572}\\
ftest(x) &= \frac{3}{4} + \cfrac{-1+x}{\frac{20}{53} - \frac{1572(-2+x)}{4399}}\\
y_3 &= 5\\
a_3 &= -\frac{87639}{9911}\\
ftest(x) &= \frac{3}{4} + \cfrac{-1+x}{\frac{20}{53} + \cfrac{-2+x}{-\frac{4399}{1572} - \frac{9911(-4+x)}{87639}}}\\
y_4 &= 6\\
a_4 &= -\frac{374}{393}\\
ftest(x) &= \frac{3}{4} + \cfrac{-1+x}{\frac{20}{53} + \cfrac{-2+x}{-\frac{4399}{1572} + \cfrac{-4+x}{-\frac{87639}{9911} - \frac{393}{374}(-5+x)}}}
\end{aligned}
$$

which in canonical form is

$$ftest(x) = \frac{1+2x+3x^2}{9-x^2} \tag{2.12}$$

The evaluation point $y = 3$ was skipped because of the singularity which G presents at this value.

Notice that even though the coefficients of the canonical form are all order 10, there are coefficients in the Thiele form even of order 10^5. The size of the coefficients becomes relevant when performing the reconstruction over finite fields, for further details see section 3.4.

2.1.3 Multivariate polynomials and rational functions

In this section we will be using multi-index notation. Define the vector of n variables $\mathbf{z} = (z_1, z_2 \cdots, z_n)$, and an n-dimensional multi-index $\alpha = (\alpha_1, \cdots, \alpha_n)$ with $\alpha_i \in \mathbb{N}$, then any monomial can be written as

$$\mathbf{z}^\alpha = \sum_{i=1}^{n} z_i^{\alpha_i} \tag{2.13}$$

Its total degree is given by

$$|\alpha| = \sum_i \alpha_i \tag{2.14}$$

A multi-variate polynomial can then be written as

$$G(\mathbf{z}) = \sum_\alpha c_\alpha \mathbf{z}^\alpha \tag{2.15}$$

and a multi-variate rational function as a ratio of two polynomials:

$$G(\mathbf{z}) = \frac{\sum_\alpha n_\alpha \mathbf{z}^\alpha}{\sum_\beta d_\beta \mathbf{z}^\beta} \tag{2.16}$$

Multivariate polynomials

Consider a polynomial $G(\mathbf{z})$ as in eq. (2.15). The idea behind the reconstruction of such a function is to reduce the problem again to an univariate reconstruction which we already know how to perform. Thus first of all fix $\{z_2, \cdots, z_n\}$ then $G(\mathbf{z})$ reduces to a function of z_1:

$$G(z_1, z_2, \cdots, z_n) = G(z_1) = \sum_{i=0}^{R} a_i(z_2, \cdots, z_n) \prod_{j=0}^{i-1} (z_1 - y_j) \tag{2.17}$$

where the $a_i(z^{,} \cdots, z_n)$ are numerical coefficients determined through eq. (2.5) with the substitutions

$$G(y_i) \to G(y_i, z_2, \cdots, z_n) \,, \qquad a_i \to a_i(z_2, \cdots, z_n) \tag{2.18}$$

However the a_i are actually polynomial functions of the remaining $n-1$ variables, whose form we need to determine so to get the dependence of G on all the n variables. Consider then for example $a_1(z_2, \cdots, z_n)$: after the first reconstruction we got one numerical value for it for assigned values of $\{z_2, \cdots, z_n\}$. If we now perform the reconstruction of G in z_1 multiple times changing only the value assigned to z_2, we will eventually end up with enough numerical values of a_1 in order to reconstruct its dependence on z_2, over which we kept sampling:

$$a_1(z_2, z_3, \cdots, z_n) = a_1(z_2) = \sum_{i=0}^{R'} b_i(z_3, \cdots, z_n) \prod_{j=0}^{i-1} (z_2 - y_j) \tag{2.19}$$

where the coefficients b_i are given by eq. (2.5) with the substitutions

$$G(y_i) \to a_1(y_i, z_3, \cdots, z_n) \,, \qquad a_j \to b_j(z_3, \cdots, z_n) \tag{2.20}$$

And similarly for all the other $a_2, \cdots, a_R$.

Again the b_i are actually polynomial functions of the $n-2$ variables $\{z_3, \cdots, z_n\}$. Now sampling on z_3 we can reconstruct the dependence of each b_i in some other coefficients, say c_i, polynomial functions in $n-3$ variables. We can thus go on like this until we end up with the coefficients being univariate polynomials in z_n, and after this last reconstruction, putting all together we get $G(\mathbf{z})$.

Notice that in order to reconstruct the z_2 dependence of $a_i(z_2, z_3, \cdots, z_n)$ one needs many numerical values of each of these a_i, which are obtained performing each time a new reconstruction of $G(z_1, z_2, z_3, \cdots, z_n)$ in z_1. Then to reconstruct $b_i(z_3, z_4, \cdots, z_n)$ in z_3 one needs many numerical values of b_i for fixed z_3, which are extrapolated from many different reconstructions of a_i in z_2 which again are extracted from many reconstructions of G in z_1. And similarly for all the successive coefficients. Thus reducing the multivariate case of $\mathbf{z}$ variables to an univariate reconstruction implies a great proliferation of evaluations of the black-box algorithm, since the reconstruction of each of the single coefficients does entail further reconstructions, each of which needs itself many evaluations of the whole G. It follows that all the evaluations performed for the computation of each coefficient should be reused as often as possible for the computation of the other coefficients, and need thus to be appropriately stored.

Multivariate rational functions

The reconstruction of multivariate rational functions is closely related to that of multivariate polynomials, in fact it can be reduced to a sequential application of it, just as this is a nested application of the univariate polynomial reconstruction. There are some subtleties to take care of, for a clear treatise of which we refer to [25].
Given a multivariate rational function $G(\mathbf{z})$ as in eq. (2.16), the trick consists of introducing a new variable, call it s, and define

$$H(s, \mathbf{z}) \equiv G(s\mathbf{z}) = G(sz_1, \cdots, sz_n) \tag{2.21}$$

Clearly H can be written in terms of s in canonical form as:

$$H(s, \mathbf{z}) = \frac{\sum_{i=0}^{R} p_i(\mathbf{z}) s^i}{\sum_{i=0}^{R'} q_i(\mathbf{z}) s^i} \tag{2.22}$$

where

$$p_i(\mathbf{z}) = \sum_{|\alpha|=i} n_\alpha \mathbf{z}^\alpha \,, \qquad q_i(\mathbf{z}) = \sum_{|\beta|=i}^{R'} d_\beta \mathbf{z}^\beta \tag{2.23}$$

with α and β multi-indexes. In other words p_i and q_i are multivariate polynomials of degree i in $\mathbf{z}$. Thus we can first reconstruct the dependence of H in s fixing all the $z_1, \cdots, z_n$, then through multivariate polynomial reconstruction get the dependence of the polynomial coefficients p_i, q_i in $\mathbf{z}$ and finally set $s = 1$ obtaining $G(\mathbf{z}) = H(1, \mathbf{z})$. In order to perform the reconstruction of each multivariate polynomial p_i and q_i we need numerical values for these functions at given values $\mathbf{z}^*$ of the coefficient, these are extrapolated from different reconstructions of $H(t, \mathbf{z}^*)$ in t. So multivariate rational reconstruction is given by several multivariate polynomial reconstructions to which a previous step given by an univariate rational reconstruction is added. It is possible to reduce by one the number of variables on which $p_i(\mathbf{z})$ and $q_i(\mathbf{z})$ depend on, somehow making up for the introduction of s. In fact being the degree of these polynomials known one could drop the dependence on one of the given variables, setting say $z_n = 1$, reconstruct the polynomials in $z_1, \cdots, z_{n-1}$ and then restore the z_n dependence simply by homogenizing the result.

Normalization of rational functions

An issue we did not address yet is the normalization of the rational functions. Whereas the coefficients of polynomials are uniquely determined, those of a rational function can be determined only up to a normalization. It can be shown that through Thiele's interpolation formula the obtained result is minimal with respect to the degrees of the numerator and denominator, hence no greatest common divisor simplification is needed when converting the result in canonical form. However an ambiguity over an overall constant factor still remains. In our specific case Mathematica itself provided the normalisation when converting to canonical form, since the output is given by default such that the coefficients are all integers. In general however one may deal with the ambiguity by requiring for example the lowest order term in the denominator of the canonical form to be equal to 1. If the constant term is non-vanishing eq. (2.16) will take the form

$$G(\mathbf{z}) = \frac{\sum_\alpha n_\alpha \mathbf{z}^\alpha}{1 + \sum\limits_{|\beta|>0} d_\beta \mathbf{z}^\beta} \tag{2.24}$$

Identifying this term before the reconstruction is in general not possible, but if the lowest-order non-vanishing term of the denominator is the constant one then this issue can be easily solved. In cases where this is not the constant term, one can always identify a suitable shift $h = (h_1, \cdots, h_n)$ in the arguments $\mathbf{z}$ such that the vanishing constant becomes non-vanishing. For further explanations about this see [25] or [12].

2.2 Finite fields

So far we focused on how a rational function can be reconstructed knowing its numerical values for some (many) given evaluation points.Now we are going to deal with a major problem arising through the usage of this reconstructional approach: intermediate expressions in the computation usually contain rationals whose numerator and denominator are huge numbers which may exceed the machine-size integers. Thus arbitrary-precision arithmetic is needed and this slows down considerably the calculation.

However, as explained e.g. in [25], one can circumvent this problem using finite fields which we will denote with $\mathbb{Z}_p$, i.e. fields comprised of only a finite number p of integers. One can map $\mathbb{Q}$ over one of these fields, making sure that p and thus all the elements of $\mathbb{Z}_p$ are machine-size integers, perform the computation and then map the result back to $\mathbb{Q}$.

Seemingly a new problem has appeared: clearly the mapping from $\mathbb{Q}$ to any given finite field cannot be invertible, since we are mapping an infinite set over a finite one. However it can be shown that if the prime p defining the field $\mathbb{Z}_p$ is big enough with respect to the numerator and denominator of the rational $z \in \mathbb{Q}$, then z can be identified uniquely starting from its image in $\mathbb{Z}_p$. In other words if p is big enough the mapping is invertible for rationals which are small enough, but this takes us back to the initial problem of not exceeding machine-size integers.

The final solution lies in the so called Chinese remainder theorem.

2.2.1 Mapping from $\mathbb{Q}$ to $\mathbb{Z}_p$

The field $\mathbb{Z}_p$

First of all what is $\mathbb{Z}_p$? The fields we are interested in are the so called Galois fields of order p, also known as finite fields because they contain only a finite number of elements. Giving a mathematically rigorous definition of the general structure we are taking advantage of is far beyond the scope of this thesis,[3] we will thus only briefly discuss the idea and the practical aspects we need.

[3]$\mathbb{Z}_p$ is a particular case of field defined by residue classes of a ring R induced by an ideal of R, see [16].

One can define a finite field of order p as the set

$$\mathbb{Z}_p = \{0, 1, \cdots p-1\} \tag{2.25}$$

endowed with suitable addition and multiplication operations.

In a more rigorous way, starting from $\mathbb{Z}$ the field $\mathbb{Z}_p$ can be constructed by defining the following equivalence classes over $\mathbb{Z}$: for a given natural number $p \in \mathbb{N}$, we say that a is equivalent to b modulo p if $a - b = np$ for some $n \in \mathbb{Z}$.

$$a \sim_p b \Leftrightarrow \exists\, n \in \mathbb{Z} \mid a - b = np \tag{2.26}$$

we say then that a, b belong to the same equivalence class, call it $[a]_p$, or simply $b = a \mod p$. Taking the set of equivalence classes $S \equiv \{[0]_p, [1]_p, \cdots, [p-1]_p\}$ we see that the group structure (under addition) of $\mathbb{Z}$ induces a group structure on S as well, in other words the map

$$\begin{aligned} F : \mathbb{Z} &\to S \\ a &\mapsto [a]_p \end{aligned} \tag{2.27}$$

is a group homomorphism. S can be seen to be the cyclic group of order p which is usually indicated as $\mathbb{Z}_p$. In eq. (2.25), with a slight abuse of notation, we used representatives of the equivalence classes instead of the classes themselves as elements of $\mathbb{Z}_p$. Now one needs to show that $\mathbb{Z}_p$ can be endowed with a multiplication operation under which it becomes a field. This can be done provided that p is a prime number. The main idea is the following: $\mathbb{Z}$ itself admits a multiplication under which however it only becomes a ring, since all requirements to be a field are satisfied a part from the inverse elements of this multiplication being defined inside $\mathbb{Z}$. We thus may use the multiplication induced by $\mathbb{Z}$ on $\mathbb{Z}_p$ to endow the latter with a ring structure. Now one needs to define the inverse elements for this operation.

Multiplicative inverse

Given $a \in \mathbb{Z}_p$ its multiplicative inverse is of the form

$$a^{-1} \in \mathbb{Z}_p \;\;\mid\;\; aa^{-1} = a^{-1}a = 1 \mod p \tag{2.28}$$

In order to prove the existence of such an inverse one can make use of the so called Bezout's identity, which states that given any two numbers $a, b \in \mathbb{Z}$ and their greatest common divisor g, there are two integers m, n such that

$$an + bm = g \tag{2.29}$$

Suppose now to take $b = p$ such that p and a are co-primes, which means that their greatest common divisor is 1. We then have

$$an + pm = 1 \quad \Rightarrow \quad an = 1 \mod p \tag{2.30}$$

Thus n is exactly the inverse we were looking for. However we had to assume that a and p were co-primes in order for a to admit a multiplicative inverse, so as to fulfil this requirement for any $a \in \mathbb{Z}_p$ we will have to take p to be a prime number. Notice that this reasoning may seem to have a flaw, since multiples of p itself[4] would admit no inverse. However all of these numbers are mapped to the identity element of the addition in $\mathbb{Z}_p$:

$$a \equiv np = 0 \mod p \tag{2.31}$$

but the zero element, i.e. the identity element for the addition, is the only element of a field which admits no inverse under multiplication.[5]

[4]These are the only possible numbers that are not co-prime with p.

[5]See definition of field A.3.

Further remarks on the multiplicative inverse in $\mathbb{Z}_p$

The fact that p needs to be a prime in order for $\mathbb{Z}_p$ to be a field can also be shown without neither the use of Bezout's identity, nor the introduction of more involved mathematical structures like ideals as it is done in [16]. All we need is the following

Proposition 2.2.1. *Given a generic $p \in \mathbb{N}$ defining the ring $\mathbb{Z}_p$ of integers modulo p, and an element $a \in \mathbb{Z}_p$ such that a and p are coprimes, we have that the product $\cdot$ maps a over the entire ring. In other words for any $c \in \mathbb{Z}_p$ there exists $b \in \mathbb{Z}_p$ such that $a \cdot b = c \mod p$, furthermore b is unique.*

Proof. First prove the uniqueness, i.e. given

$$ab = c \mod p, \qquad ab' = c' \mod p$$

if $b \neq b'$ then $c \neq c'$. Suppose we had $c = c' \mod p$ then

$$c - c' = 0 \mod p \quad \Rightarrow \quad a(b - b') = 0 \mod p$$

meaning that $a(b - b')$ is a multiple of p. Being a and p coprimes their least common multiple is ap, however clearly $(b - b') < p$, which would lead to a contradiction. Thus $a(b - b')$ can't be a multiple of p so $c \neq c'$.

It follows that the image of the map $P_a : \mathbb{Z}_p \to \mathbb{Z}_p$ defined by $P_a(b) = ab$ must be the whole $\mathbb{Z}_p$ in order for no product to yield the same result. □

Due to proposition 2.2.1 if $a \in \mathbb{Z}_p$ and p are coprimes, there exists a unique $b \in \mathbb{Z}_p$ such that $ab = 1 \mod p$, i.e. a admits a uniquely defined inverse under multiplication. In order for $\mathbb{Z}_p$ to be a field such an inverse must be defined for every element of $\mathbb{Z}_p$ so p must be coprime with $\{0, 1, \cdots, p-1\}$ meaning that p must be a prime.

The different behaviour of multiplication in $\mathbb{Z}_p$ for p generic and p prime can be seen using the Cayley table[6] for the product operation. Take as an example the ring $\mathbb{Z}_8$ and the field $\mathbb{Z}_7$:

$\cdot$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Table 2.1: Cayley table for the product in $\mathbb{Z}_8$

[6]A Cayley table for a given operation is a table reporting the outcome of that operation when applied to any possible pair of elements of the group, ring or field. Notice that we introduce Cayley tables for demonstration purpose only, they are not used to perform any calculation here after.

$\cdot$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Table 2.2: Cayley table for the product in $\mathbb{Z}_7$

As can be seen $2, 4, 6 \in \mathbb{Z}_8$, which are not coprime with 8, are not mapped over the entire ring, moreover they are mapped several times over 0: each of these zeros is a common multiple of the given element $a \in \mathbb{Z}_8$ and 8 smaller than $a \cdot 8$.

This second proof may seem redundant but it is important to stress the fact that even for p non prime there are many elements in $\mathbb{Z}_p$ which admit a unique inverse under multiplication. This fact will prove crucial when introducing the Chinese remainder theorem.

The mapping from $\mathbb{Q}$ to $\mathbb{Z}_p$

At this point, for p prime, we can define a mapping

$$\begin{aligned} F : \mathbb{Q} &\to \mathbb{Z}_p \\ \frac{a}{b} &\mapsto (a \cdot (b^{-1} \mod p) \mod p) \end{aligned} \tag{2.32}$$

This map is not well defined for all elements of $\mathbb{Q}$, in fact if b is a multiple of p then its multiplicative inverse does not exist because $b = 0 \mod p$. Thus also the image of z is not defined. There is no analytic solution to this. However recall that our goal is to perform the reconstruction of a rational function over finite fields, meaning that we need to evaluate the black-box algorithm $G(z)$ at several different values of the variable z. If during the evaluation at one of these values the above scenario presented itself, i.e. F failed to provide a proper image, our algorithm would simply skip that value of z, take a new one and keep running. A more detailed discussion of this is provided in chapter 4.

2.2.2 Map from $\mathbb{Z}_p$ to $\mathbb{Q}$ and Euclidean algorithm

On the uniqueness of the inverse element

The mapping $F : \mathbb{Q} \to \mathbb{Z}_p$ is clearly not invertible, in the sense that it is not possible to properly define an inverse function F^{-1}. This is due to the fact that the map F is not injective, starting from an element $c \in \mathbb{Z}_p$ there are infinite possible rationals $x = \frac{a}{b} \in \mathbb{Q}$ that are mapped to that c. Notice however that there are only finitely many such that $a, b < p$ simply because p is finite. The crucial fact[7] is that there is only one such that $a^2, b^2 < \frac{p}{2}$:

Proposition 2.2.2. *Given an element $c \in \mathbb{Z}_p$, there is only one possible $x = \frac{a}{b} \in \mathbb{Q}$ such that $a^2, b^2 < \frac{p}{2}$ and $x = c \mod p$*

[7]See for example [30].

Proof. Suppose that there were another pair of values a', b' such that $\frac{a'}{b'} = c \mod p$ and $(a')^2, (b')^2 < \frac{p}{2}$, then we would have

$$\frac{a}{b} = \frac{a'}{b'} \mod p \quad \Rightarrow \quad ab' = ba' \mod p$$

meaning that there is an integer n such that $ab' - ba' = np$. However $|ab'| < \frac{p}{2}$ and same goes for $|ba'| < \frac{p}{2}$, thus

$$-p < np < p$$

which fixes $n = 0$ and so $ab' = ba'$, then $\frac{a}{b} = \frac{a'}{b'}$ and x is thus uniquely determined. □

Given a certain rational $z = \frac{a}{b}$ there is then an infinite set of growing primes $\{p_1, p_2, \cdots\}$, which we will call I_z, such that $c = F(z)$ and c is not the image of any other rational $z' = \frac{a'}{b'}$ with $(a')^2, (b')^2 < \frac{p}{2}$. Suppose then to have chosen $p \in I_z$, how to determine the uniquely defined z starting from c? Again the answer can be found in Bezout's identity, in particular in the extended Euclidean algorithm which allows to compute the coefficients m, n in eq. (2.29).

Euclidean algorithm

The Euclidean algorithm is a recursive technique which allows to determine the greatest common divisor (g.c.d.) of two given numbers.

Suppose we were looking for $\gcd(a, b)$ with $a > b$, that is the biggest number g such that $a = c_a g$ and $b = c_b g$ for some positive integers c_a and c_b. Clearly, since $a - b = (c_a - c_b)g$, g will also divide $a - b$ and on the same footing it will also divide $r_0 = a - q_0 b$ where q_0 is the quotient of a and b and r_0 the remainder. So consider now the pair (b, r_0), since g divides both of them it also divides $r_1 = b - q_1 r_0$ with q_1 quotient of b and r_0. We can then repeat the argument for (r, r_1) and then (r_1, r_2) and so on until we get to a pair (r_{N-1}, r_N) whose remainder is zero:

$$\begin{aligned} a &= q_0 b + r_0 \\ b &= q_1 r_0 + r_1 \\ &\vdots \\ r_{N-3} &= q_{N-1} r_{N-2} + r_{N-1} \\ r_{N-2} &= q_N r_{N-1} + 0 \end{aligned} \tag{2.33}$$

Notice that the recursion terminates after a number of steps N that must be finite because there are only finitely many integers in the interval $[0, a]$ and $0 \le r_i < |r_{i-1}|$. So we found that g divides all the non-zero r_i and in particular it divides r_{N-1}, thus we must have that $r_{N-1} \ge g$. However starting from the bottom of eq. (2.33) and substituting back the remainders we see that

$$\begin{aligned} r_{N-2} &= q_N r_{N-1} \\ r_{N-3} &= q_{N-1} r_{N-2} + r_{N-1} = (q_{N-1} q_N + 1) r_{N-1} \\ &\vdots \\ b &= (\cdots) r_{N-1} \\ a &= (\cdots) r_{N-1} \end{aligned} \tag{2.34}$$

where the $(\cdots)$ stands for some combination of products of q_i. So r_{N-1} divides both a and b, then we must have $r_{N-1} \le g$. So ultimately $r_{N-1} = g$.

The Euclidean algorithm defines a succession $\{r_i\}$ whose last non-zero term is the greatest common divisor we were looking for. In order to incorporate also a and b in this succession and define the algorithm in a completely recursive way we can relabel the remainders and set:

$$\begin{aligned}
r_0 &\equiv a \\
r_1 &\equiv b \\
r_2 &= r_0 - q_1 r_1 \\
&\vdots \\
r_{i+1} &= r_{i-1} - q_i r_i \\
&\vdots \\
r_{N-2} &= q_N r_{N-1}
\end{aligned} \tag{2.35}$$

where $0 \leq r_{i+1} < |r_i|$.

Extended Euclidean algorithm

One can easily extend the algorithm in order to compute not only the g.c.d but also the coefficients m, n of Bezout's identity

$$am + bn = \gcd(a, b) \tag{2.36}$$

Introduce other two successions $\{s_i\}$ and $\{t_i\}$ defined by:

$$\begin{array}{cc}
s_0 \equiv 1 & t_0 \equiv 0 \\
s_1 \equiv 0 & t_1 \equiv 1 \\
\vdots & \vdots \\
s_{i+1} = s_{i-1} - q_i s_i & t_{i+1} = t_{i-1} - q_i t_i \\
\vdots & \vdots
\end{array} \tag{2.37}$$

these satisfy

$$as_i + bt_i = r_i \tag{2.38}$$

which can be shown by induction: for $i = 0, 1$ it is true by construction, assuming it to be true for $i = k$ we have

$$\begin{aligned}
r_{k+1} &= r_{k-1} - r_k q_k \\
&= (as_{k-1} + bt_{k-1}) - (as_k + bt_k) q_k \\
&= (as_{k-1} - as_k q_k) + (bt_{k-1} - bt_k q_k) \\
&= as_{k+1} + bt_{k+1}
\end{aligned} \tag{2.39}$$

and this proves the identity. When the recursion terminates at step N having $r_N = 0$, we get $\gcd(a, b) = r_{N-1}$ and $n = s_{N-1}$, $m = t_{N-1}$. If a and b are positive and $\gcd(a, b) \neq min(a, b)$, it can be shown that

$$|s_i| < \frac{b}{\gcd(a, b)} \qquad |t_i| < \frac{a}{\gcd(a, b)} \tag{2.40}$$

The inverse mapping

The extended Euclidean algorithm can be used to invert the map of eq. (2.32). If we take eq. (2.38) and set $b = p$ we have

$$as_i = r_i \mod p \quad \Rightarrow \quad \frac{r_i}{s_i} = a \mod p \tag{2.41}$$

Thus given an element $c \in \mathbb{Z}_p$, by running the extended Euclidean algorithm for (c, p) we get at each step of the recursion a possible rational $x_i = \frac{r_i}{s_i}$ whose image in $\mathbb{Z}_p$ is c, so each step of the recursion provides us with a possible candidate for the rational we are seeking. Among these candidates there will be at most one such that $r_i^2, s_i^2 < \frac{p}{2}$, because due to proposition 2.2.2 only one such rational exists. As a result, if the prime p defining $\mathbb{Z}_p$ is big enough with respect to the numerator and denominator of the rational z whose image is c, i.e. $p \in I_z$, we will be able to determine z without ambiguity through the extended Euclidean algorithm. The fact that, supposing a $z = \frac{n}{d}$ such that $n^2, d^2 < \frac{p}{2}$ exists i.e. p is large enough, this z can be found among the candidates provided by the extended Euclidean algorithm is proven in [31].

Notice however that in the choice of p we are bound to the machine-size numbers, otherwise we will fall back into the problem we tried to solve by introducing the mapping over $\mathbb{Z}_p$. In order to be able to invert the mapping from $\mathbb{Q}$ to $\mathbb{Z}_p$ uniquely for rationals of any given size we need somehow to access "larger" finite fields but still working with machine-size integers. This can be done through the Chinese remainder theorem.

2.2.3 Chinese remainder theorem

The Chinese remainder theorem states that given the image of a number $X \in \mathbb{Q}$ over the fields $\mathbb{Z}_{p_1}, \cdots, \mathbb{Z}_{p_n}$ defined by the primes $p_1, \cdots, p_n$, one can always compute the image of X over the ring defined by $P = p_1 \cdots p_n$. Actually one can prove a more general statement:

Theorem 2.2.3 (Chinese remainder)**.** *Let $n_1, \cdots, n_m$ be integers greater than 1 and pairwise coprime, let $a_1, \cdots, a_k$ be integers such that $0 \leq a_i < n_i$ for every i, then there exists only one $0 \leq X < N$ with $N = \prod_i n_i$ such that*

$$\begin{cases} X &= a_1 \mod n_1 \\ & \vdots \\ X &= a_k \mod n_k \end{cases} \tag{2.42}$$

Proof. **Uniqueness:** In order to prove the uniqueness of the solution suppose that two numbers x and y both solve all the congruences. When divided by n_i they give the same remainder a_i, so their difference $x - y$ is a multiple of each of the n_i. As the n_i are pairwise coprime $x - y$ must also be a multiple of N, say $x - y = cN$. But $x < N$ and $y < N$ so we must have $c = 0$ and $x = y$.

Existence: The existence can be proven constructing a solution. Consider the first two equations of eq. (2.42), these are equivalent to

$$\begin{cases} X &= a_1 \mod n_1 = a_1 + l_1 n_1 \\ X &= a_2 \mod n_2 = a_2 + l_2 n_2 \end{cases} \tag{2.43}$$

for some $l_1, l_2 \in \mathbb{Z}$. Thus

$$a_1 + l_1 n_1 = a_2 + l_2 n_2 \tag{2.44}$$

Since n_1 and n_2 are co-prime, one can write Bezout's identity eq. (2.29) for them as[8]

$$m_1 n_1 + m_2 n_2 = 1 \tag{2.45}$$

using this equation we can solve eq. (2.44) with respect to l_1:

$$l_1 n_1 = a_2 - a_1 + l_2 n_2 \tag{2.46}$$

multiply both sides by m_1

$$\begin{aligned} l_1(1 - m_2 n_2) &= m_1(a_2 - a_1) + l_2 m_1 n_1 \\ &\Downarrow \\ l_1 &= m_1(a_2 - a_1) + \underbrace{(m_1 l_2 + m_2 l_1)}_{\equiv L \in \mathbb{Z}} n_2 \end{aligned} \tag{2.47}$$

Then X can be written as

$$X = a_1 + l_1 n_1 = \underbrace{a_1 + m_1 n_1 (a_2 - a_1)}_{\equiv A_1} + L n_1 n_2 \tag{2.48}$$

in other words

$$X = A_1 \mod n_1 n_2 \tag{2.49}$$

in a similar fashion one could have solved eq. (2.44) in l_2 obtaining:

$$l_2 = m_2(a_1 - a_2) + \underbrace{(m_1 l_2 + m_2 l_1)}_{L} n_1 \tag{2.50}$$

$$\begin{aligned} X &= a_2 + l_2 n_2 \\ &= a_2 + m_2 n_2 (a_1 - a_2) + L n_1 n_2 \\ &= a_2 + (1 - m_1 n_1)(a_1 - a_2) + L n_1 n_2 \\ &= a_1 + m_1 n_1 (a_2 - a_1) + L n_1 n_2 \\ &= A_1 \mod n_1 n_2 \end{aligned} \tag{2.51}$$

Again using $m_1 n_+ m_2 n_2 = 1$ we can write A_1 in a more symmetric form:

$$A_1 = a_1 m_2 n_2 + a_2 m_1 n_1 \tag{2.52}$$

The fact that this A_1 solves the system eq. (2.43) can be seen immediately from eq. (2.48) and eq. (2.51):

$$\begin{aligned} X &= a_1 + m_1 n_1 (a_2 - a_1) + L n_1 n_2 \\ &= a_1 + [m_1(a_2 - a_1) + L n_2] n_1 \\ &= a_1 \mod n_1 \end{aligned} \tag{2.53}$$

$$\begin{aligned} X &= a_2 + m_2 n_2 (a_1 - a_2) + L n_1 n_2 \\ &= a_2 + [m_2(a_1 - a_2) + L n_1] n_2 \\ &= a_2 \mod n_2 \end{aligned} \tag{2.54}$$

where the terms between square brackets are integers.

[8] The coefficients m_1 and m_2 may be computed for example using the extended Euclidean algorithm.

At this point we have reduced eq. (2.42) from being a system of k equations to the system of $k-1$ equations

$$\begin{cases} X &= A_1 \mod n_1 n_2 \\ & \vdots \\ X &= a_k \mod n_k \end{cases} \tag{2.55}$$

Take now the first two equations of this new system

$$\begin{cases} X &= A_1 \mod n_1 n_2 \\ X &= a_3 \mod n_3 \end{cases} \tag{2.56}$$

Defining:

$$N = \prod_j n_j \tag{2.57}$$

$$N_i = \frac{N}{n_i} \tag{2.58}$$

since N_i and n_i are co-primes Bezout's identity reads

$$M_i N_i + m_i n_i = 1 \tag{2.59}$$

for some integers M_i, m_i. Following the same steps as above we find

$$\begin{aligned} X &= A_2 \mod n_1 n_2 n_3 \\ &= x_1 N_1 M_1 + x_2 N_2 M_2 + x_3 N_3 M_3 \mod n_1 n_2 n_3 \end{aligned} \tag{2.60}$$

which leads to the system of $k-2$ equations

$$\begin{cases} X &= A_2 \mod n_1 n_2 n_3 \\ & \vdots \\ X &= a_k \mod n_k \end{cases} \tag{2.61}$$

Repeating the procedure other $k-3$ times one ends up with

$$\begin{aligned} X &= A_{k-1} \mod n_1 \cdots n_k \\ &= \sum_{i=1}^{k} a_i M_i N_i \mod N \end{aligned} \tag{2.62}$$

It can be easily seen that A_{k-1} defined as above solves all the congruences in eq. (2.42). In fact notice that N_j is a multiple of n_i for $i \neq j$ in particular

$$N_j = \frac{N_i}{n_j} n_i$$

then isolating the i-th term and using $M_i N_i = 1 - m_i n_i$

$$\begin{aligned} X &= \sum_{j=1}^{m} a_j M_j N_j \\ &= a_i(1 - m_i n_i) + \sum_{j \neq i} a_j M_j \frac{N_i}{n_j} n_i \\ &= a_i + \left(\sum_{j \neq i} a_j M_j \frac{N_i}{n_j} - a_i m_i \right) n_i \\ &= a_i \mod n_i \end{aligned}$$

being the term in parenthesis an integer. This is true for every i and completes the proof. $\square$

This theorem can also be stated saying that given $\{n_1, \cdots, n_m\}$ pairwise coprime there is a ring isomorphism between the ring of integers modulo $N = \prod_i n_i$ and the direct product of the rings of integers modulo n_i:

$$\begin{aligned} C: \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_m} &\to \mathbb{Z}_N \\ (X \mod n_1, \cdots, X \mod n_m) &\mapsto X \mod N \end{aligned} \tag{2.63}$$

This means that one may perform independently a same computation over each of the n_i getting then through the Chinese remainder the value of the same computation if it were performed over $\mathbb{Z}_N$. As can be seen the Chinese remainder theorem exactly fits our needs. Consider $z = \frac{a}{b} \in \mathbb{Q}$ which is the rational result of a computation we are actually performing over finite fields, and suppose a^2, b^2 are beyond machine size. We then still consider p of machine size, but perform the calculation over several of these fields $\{\mathbb{Z}_{p_1}, \cdots, \mathbb{Z}_{p_m}\}$. Then we use the map of eq. (2.63) to obtain the image of z in $\mathbb{Z}_P$ with $P = p_1 \cdots p_m$, if $P \in I_z$ we can compute z uniquely. Else compute z over a new field defined by p_{m+1}, then use the Chinese remainder to map to the ring defined by $P' = p_1 \cdots p_m p_{m+1}$. If $P' \in I_z$ find the unique z, else keep going until the product of all the p_i is in I_z.

Notice that the isomorphism C defined by the Chinese remainder theorem is a *ring* isomorphism which may be extended using the mapping over finite fields to a ring homomorphism

$$\mathbb{Z} \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_m} \to \mathbb{Z}_P \tag{2.64}$$

However it cannot be extended to a *field* homomorphism

$$\mathbb{Q} \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_m} \to \mathbb{Z}_P \tag{2.65}$$

since $P = p_1 \cdots p_m$ is clearly not a prime number and thus $\mathbb{Z}_P$ cannot be endowed with a suitable product operation because not all the elements in $\mathbb{Z}_P$ admit a multiplicative inverse. The problematic elements are those that are not coprime with P, but since $p_1, \cdots, p_m$ are all primes, only multiples of the p_i do not admit an inverse.[9] In other words the problem may arise only if one of the a_i in eq. (2.42) was zero, but this can be easily recognized by the code, the associated p_i excluded from computation and then inversion becomes defined for the value computed.

[9]See proposition 2.2.1 and subsequent reasoning.

Chapter 3

Rational functions from finite fields

3.1 Polynomial reconstruction

As mentioned in section 2.1.1 we will use the Newton's form for univariate polynomials for the reconstruction process.

The first input the routine needs is the black-box algorithm, call it `Fin`, which returns the numerical value of the polynomial to be reconstructed given a value of the variable, call it x.

$$\mathtt{x} = \text{rational} \quad \xrightarrow{input for} \quad \mathtt{Fin[x]} \quad \xrightarrow{returns} \quad \text{rational} \tag{3.1}$$

As a second input one can also specify an integer which will be used for the checking procedure of the reconstructed function. We will come back into this later in more detail. Finally the output will be the analytic form of the reconstructed function.[1]

Before starting the reconstruction itself we need to define some functions which the routine will be making use of:

- a local function called `F` which stores the values of `Fin` computed along the process
- a function called `h` generating the sampling points
- a local function `f` which is the function we are reconstructing. Once the computation is terminated correctly `f` will be given as output.
- a recursive function called `l` which allows to generate the analytic form of `f` in terms of the coefficients we called a_i in section 2.1.1
- a recursive function called `g` which generates the analytic form of the coefficients a_i in terms of a_j with $j < i$ and of the sampling points `h[i]`
- a function called `a[i]` storing the numerical value computed for each of the coefficients a_i. These are the actual coefficients used by `l` to construct the function `f`.

Storing the values of `Fin`

The most time-expensive part of the reconstruction is the evaluation of `Fin`. It is thus useful to store the values one computed, in order not waste time in repeating a same

[1]The variable name of the output may be specified through a third optional input to the routine.

calculation more than once. This strategy comes at handy mainly in the result checking procedure. To accomplish this we define a local variable F as:

```
F[x_] := F[x] = Fin[x];
```

Sampling points h[i]

Throughout the computation the black-box algorithm needs to be evaluated several times over different sampling points generated by the function h[i]. These points are completely arbitrary provided they do not repeat themselves. If the i-th sampling point were chosen to be the same as a preceding one, a singularity would inevitably appear in the computation of the coefficient a[i]. This can be clearly seen from the coefficients reported in figure 3.3, consider for example choosing h[3]=h[2] and computing a[3]. Since in the polynomial reconstruction no infinities may arise at any step of the computation, there is no built-in recovery mechanism from such a scenario. Thus an artificial infinite of this type would blow up the entire computation and must be avoided.

h has been defined in order to be as simple as possible. We chose to take consecutive integers starting from 1:

```
h[0] = 1;
h[i_] := h[i] = h[i - 1] + 1;
```

Notice that h is defined in order to store its values once computed.

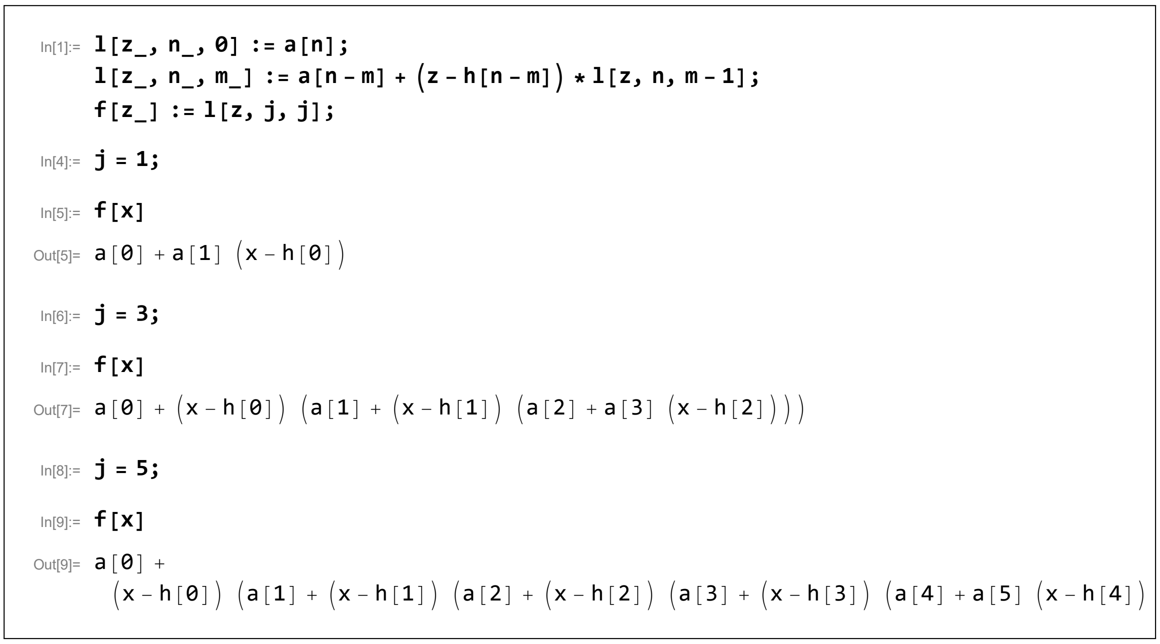
The test function f

The polynomial f is the test function one builds during the reconstruction process. It is built step by step computing successive coefficients a[i] until a given number of test evaluations of f and Fin match. f will be computed in Newton form, thus recursively with respect to the a[i]. To generate it an auxiliary function l is used. In figure 3.1, alongside the definitions, three examples of f[x] are reported. They show f at the step j=1,3,5 of the computation, i.e. after coefficients up to a[1],a[3] and a[5] have been determined. If the computation was actually performed the a[i] would all be replaced by the appropriate rational numbers.

The coefficients a and the function g

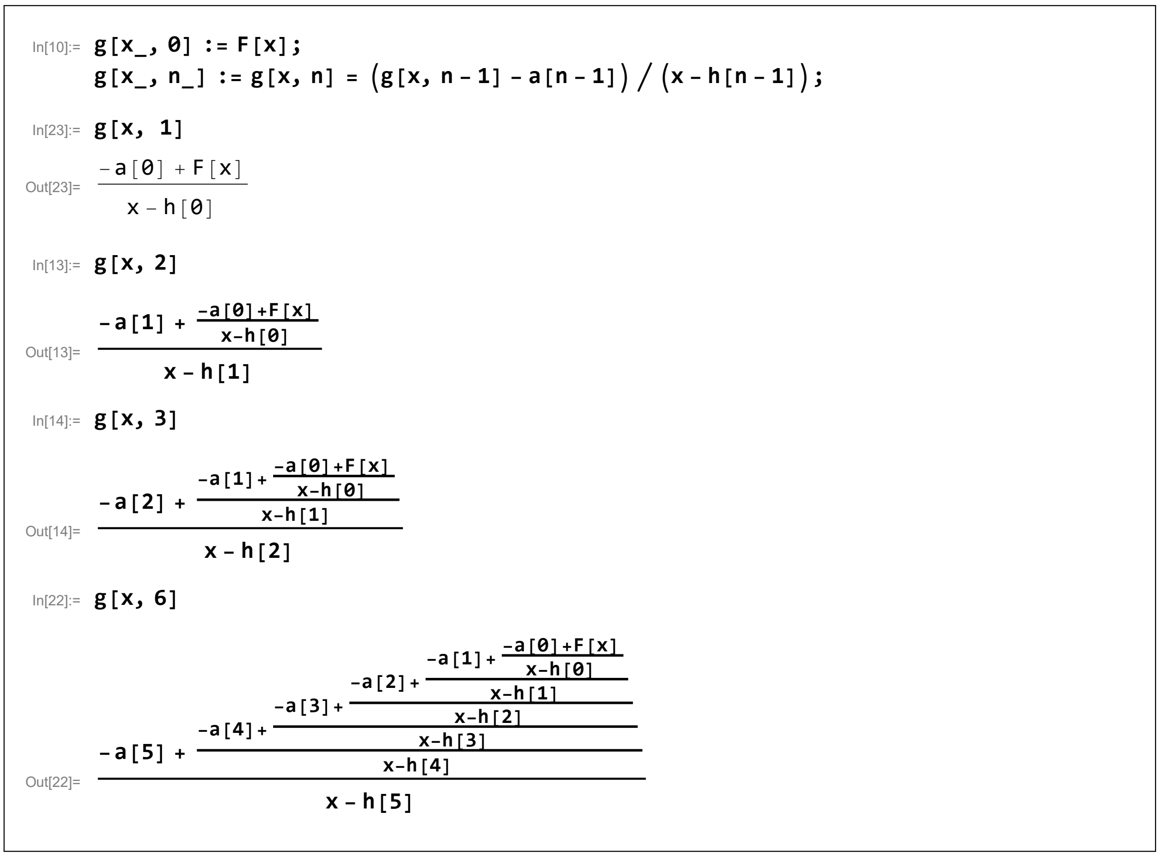
Similarly to f, the coefficients a_i are computed recursively. The recursion itself is performed by the auxiliary function g. The definition of g with some examples is shown in figure 3.2. The $g[x,n]$ is the a_n coefficient computed at x expressed in terms of all the a_i with $i < n$. The printed example outputs correspond to a_1,a_2,a_3 and a_6. The expressions shown still display their analytical dependence on F and h, because these have not been specified when printing the examples in order for the latter to be as general as possible.

The numerical value of the n-th coefficient computed at the n-th sampling point h[n] is stored in a[n], defined as in figure 3.3. The examples printed correspond to a_1,a_2,a_3 previously shown in the examples of g. We did not display the value for $i = 6$ because of



```
In[1]:= l[z_, n_, 0] := a[n];
        l[z_, n_, m_] := a[n - m] + (z - h[n - m]) * l[z, n, m - 1];
        f[z_] := l[z, j, j];

In[4]:= j = 1;

In[5]:= f[x]

Out[5]= a[0] + a[1] (x - h[0])

In[6]:= j = 3;

In[7]:= f[x]

Out[7]= a[0] + (x - h[0]) (a[1] + (x - h[1]) (a[2] + a[3] (x - h[2])))

In[8]:= j = 5;

In[9]:= f[x]

Out[9]= a[0] +
         (x - h[0]) (a[1] + (x - h[1]) (a[2] + (x - h[2]) (a[3] + (x - h[3]) (a[4] + a[5] (x - h[4])))))
```

Figure 3.1: Definition of the test function `f` in terms of the auxiliary function `l`.



```
In[10]:= g[x_, 0] := F[x];
         g[x_, n_] := g[x, n] = (g[x, n - 1] - a[n - 1]) / (x - h[n - 1]);

In[23]:= g[x, 1]
```

$$\text{Out[23]=}\quad \frac{-a[0] + F[x]}{x - h[0]}$$

```
In[13]:= g[x, 2]
```

$$\text{Out[13]=}\quad \frac{-a[1] + \frac{-a[0]+F[x]}{x-h[0]}}{x - h[1]}$$

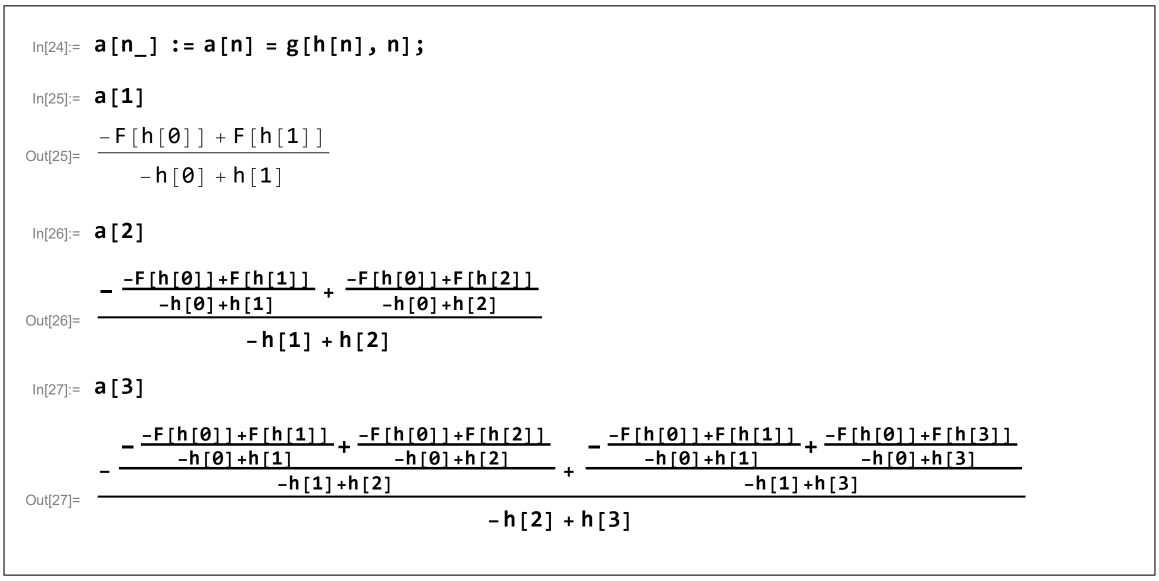
```
In[14]:= g[x, 3]
```

$$\text{Out[14]=}\quad \frac{-a[2] + \frac{-a[1]+\frac{-a[0]+F[x]}{x-h[0]}}{x-h[1]}}{x - h[2]}$$

```
In[22]:= g[x, 6]
```

$$\text{Out[22]=}\quad \frac{-a[5] + \frac{-a[4]+\frac{-a[3]+\frac{-a[2]+\frac{-a[1]+\frac{-a[0]+F[x]}{x-h[0]}}{x-h[1]}}{x-h[2]}}{x-h[3]}}{x-h[4]}}{x - h[5]}$$

Figure 3.2: Auxiliary function `g`, definition and examples.



```
In[24]:= a[n_] := a[n] = g[h[n], n];

In[25]:= a[1]
```

Out[25]=
$$\frac{-F[h[0]] + F[h[1]]}{-h[0] + h[1]}$$

```
In[26]:= a[2]
```

Out[26]=
$$\frac{-\frac{-F[h[0]]+F[h[1]]}{-h[0]+h[1]} + \frac{-F[h[0]]+F[h[2]]}{-h[0]+h[2]}}{-h[1] + h[2]}$$

```
In[27]:= a[3]
```

Out[27]=
$$\frac{-\frac{-\frac{-F[h[0]]+F[h[1]]}{-h[0]+h[1]}+\frac{-F[h[0]]+F[h[2]]}{-h[0]+h[2]}}{-h[1]+h[2]} + \frac{-\frac{-F[h[0]]+F[h[1]]}{-h[0]+h[1]}+\frac{-F[h[0]]+F[h[3]]}{-h[0]+h[3]}}{-h[1]+h[3]}}{-h[2] + h[3]}$$

Figure 3.3: Definition of the numerical coefficients `a` in terms of the auxiliary function `g` and some examples.

the large size of the output. Again all the values are analytical since nor the function `F` nor `h` have been specified.

In the denominators of each of the fractions defining the coefficients a_i there are always only differences among evaluation points. It is thus impossible that any `a[i]` diverges unless we chose two coinciding sampling points. But we require `h` to be a sequence of non-repeating numbers and thus no infinity can appear anywhere through out the computation.

The reconstruction algorithm

Once all these definitions have been given the routine proceeds as shown in algorithm 1 and the associated flowchart figure 3.4. The input parameter `n` is an integer which defines for how many different values of z the test function and the black-box algorithm are required to coincide before declaring the reconstruction to be successful and exit the routine. This last condition is tested by the `check` algorithm which is shown in algorithm 2 and figure 3.5. Its inputs are the two functions `f` and `F` whose coincidence is to be checked and the integer `n`,[2] which is again the number of points on which the two functions are required to coincide before terminating the check.

On the following pages we displayed some examples of polynomial reconstruction. In Mathematica `x // f` means applying the function `f` to `x`, and the symbol % stands for the last output.

```
input  : Fin=black-box algorithm, n=integer
output: f=analytic form of reconstructed function
F[x]=Fin[x];
j=0;
while True do
    compute and store F[h[j]];
    if F[h[j]]=∞ then
        h[j]=h[j+1];
        clear stored value h[j+1];
    else
        compute and store a[j];
        if a[j]=∞ then
            h[j]=h[j+1];
            clear stored value h[j+1];
            clear stored value a[j];
        else
            compute analytic form of f with the a[j] so far evaluated;
            if check[F,f,n]=True then
                Break;
            else
                j=j+1;
            end
        end
    end
end
```

Algorithm 1: Univariate polynomial reconstruction algorithm `polyrec`. The functions `g,a,l,f,h` were defined in the preceding subsections.

[2]As an optional argument the user can also define the name of the output variable. We did so for all the examples in the following pages.

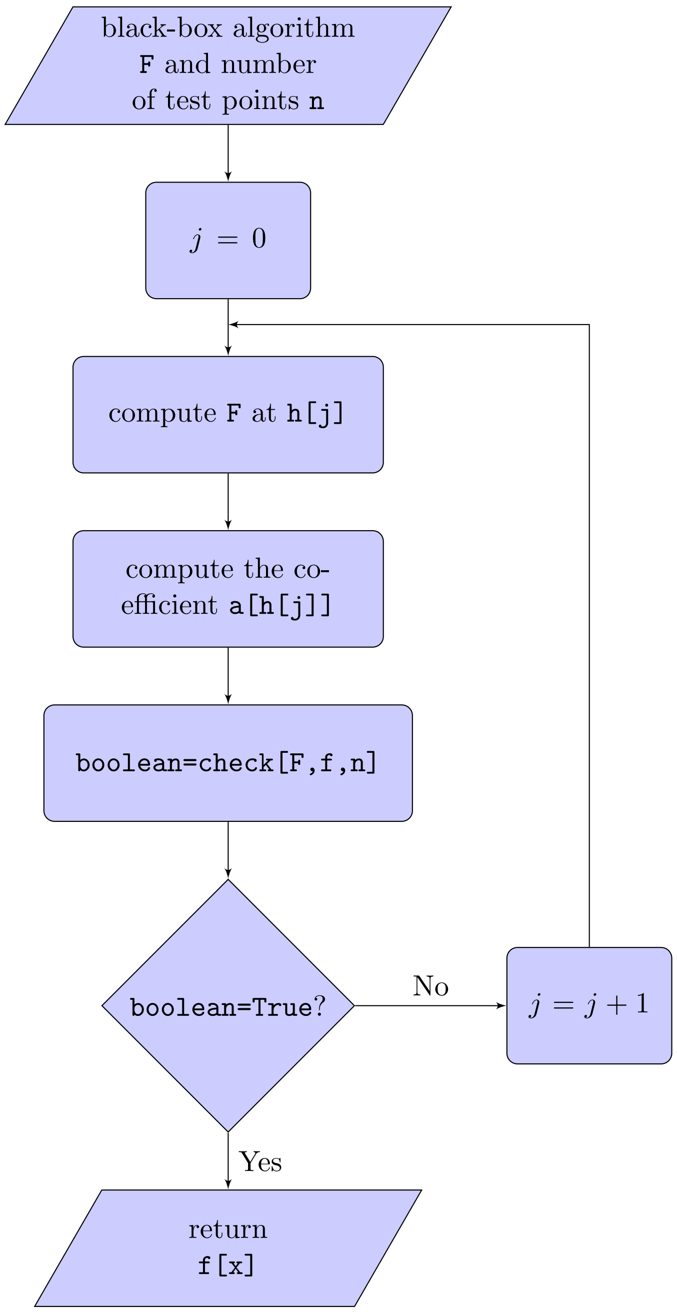


Figure 3.4: Flowchart of the polynomial reconstruction algorithm. The functions `a,f,h` were defined in the preceding subsections. The flowchart of the algorithm `check` which checks the coincidence of the original and the reconstructed function is given in 3.5.

```
input  : F,f functions to be compared, n=integer
output: out=Boolean variable
track=h[j];
h[j]=h[j+1];
clear stored value of h[j+1];
count=0;
while $count < n$ do
    compute F[h[j]];
    compute f[h[j]];
    if F[h[j]]=f[h[j]] then
        count=count+1;
        h[j]=h[j+1];
        clear stored value of h[j+1];
    else
        Break;
    end
end
if $count=n$ then
    out=True;
else
    out=False;
end
h[j]=track;
```

Algorithm 2: Coincidence check function, `check`, for univariate polynomial reconstruction algorithm.

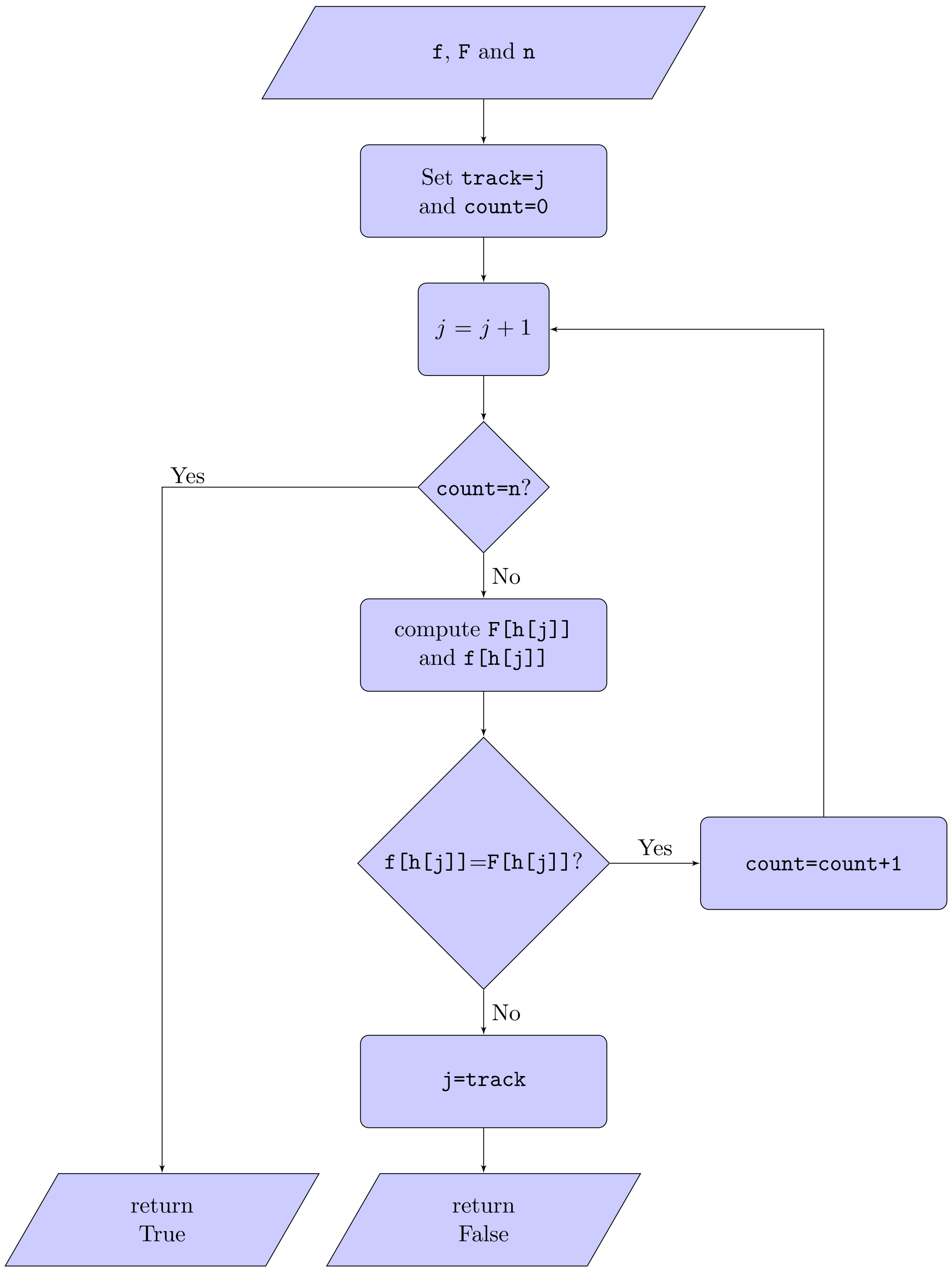
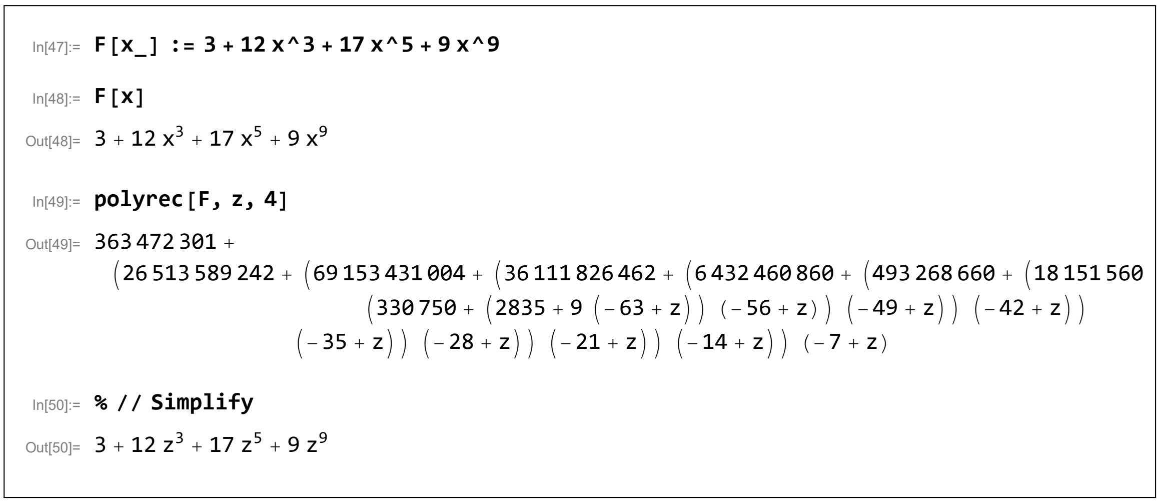


Figure 3.5: Flowchart of the algorithm which checks the coincidence of the reconstructed polynomial and the results given by the black-box algorithm.



```
In[47]:= F[x_] := 3 + 12 x^3 + 17 x^5 + 9 x^9

In[48]:= F[x]
```

Out[48]= $3+12x^3+17x^5+9x^9$

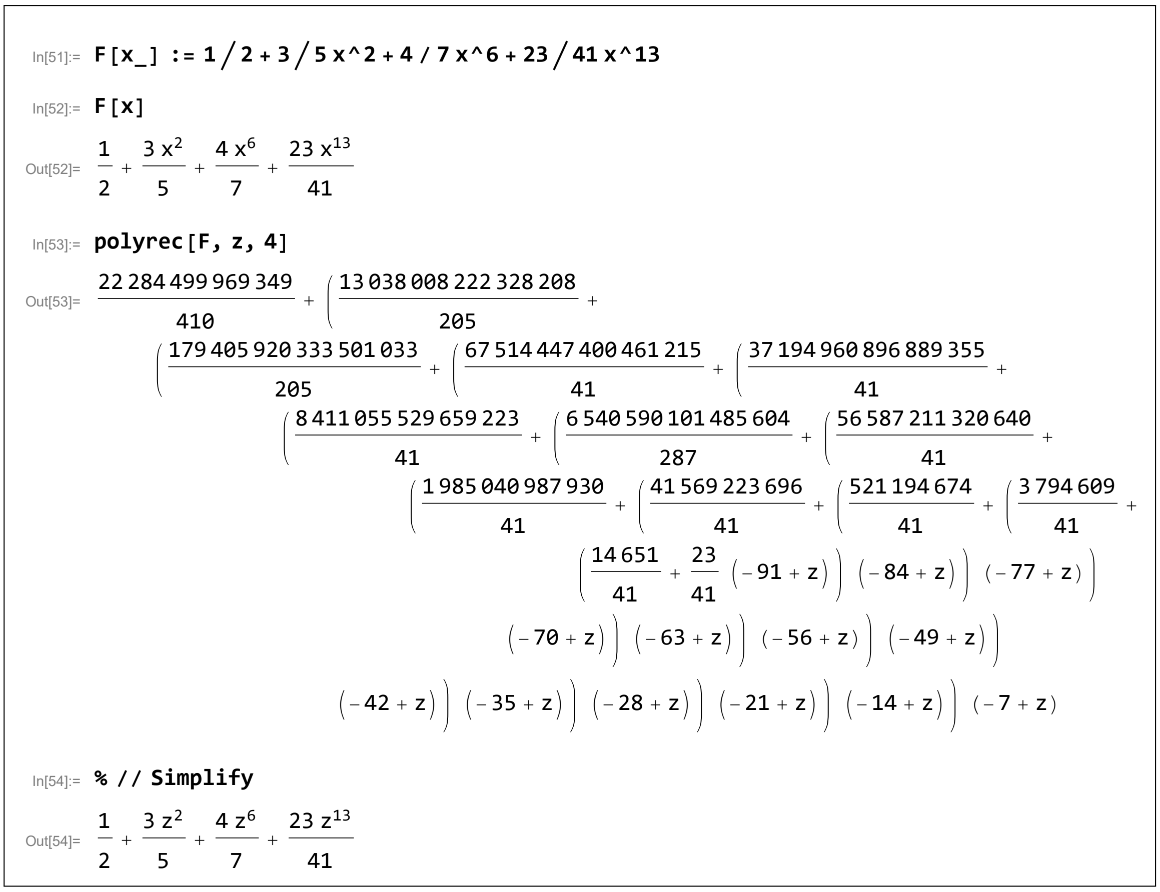
```
In[49]:= polyrec[F, z, 4]
```

Out[49]=
$$363\,472\,301+\Big(26\,513\,589\,242+\Big(69\,153\,431\,004+\Big(36\,111\,826\,462+\Big(6\,432\,460\,860+\Big(493\,268\,660+\Big(18\,151\,560+\big(330\,750+\big(2835+9\,(-63+z)\big)\,(-56+z)\big)\,(-49+z)\Big)\,(-42+z)\Big)\,(-35+z)\Big)\,(-28+z)\Big)\,(-21+z)\Big)\,(-14+z)\Big)\,(-7+z)$$

```
In[50]:= % // Simplify
```

Out[50]= $3+12z^3+17z^5+9z^9$

Figure 3.6: Example of polynomial reconstruction. A simple function.



```
In[51]:= F[x_] := 1/2 + 3/5 x^2 + 4 / 7 x^6 + 23/41 x^13

In[52]:= F[x]
```

Out[52]= $\frac{1}{2}+\frac{3x^2}{5}+\frac{4x^6}{7}+\frac{23x^{13}}{41}$

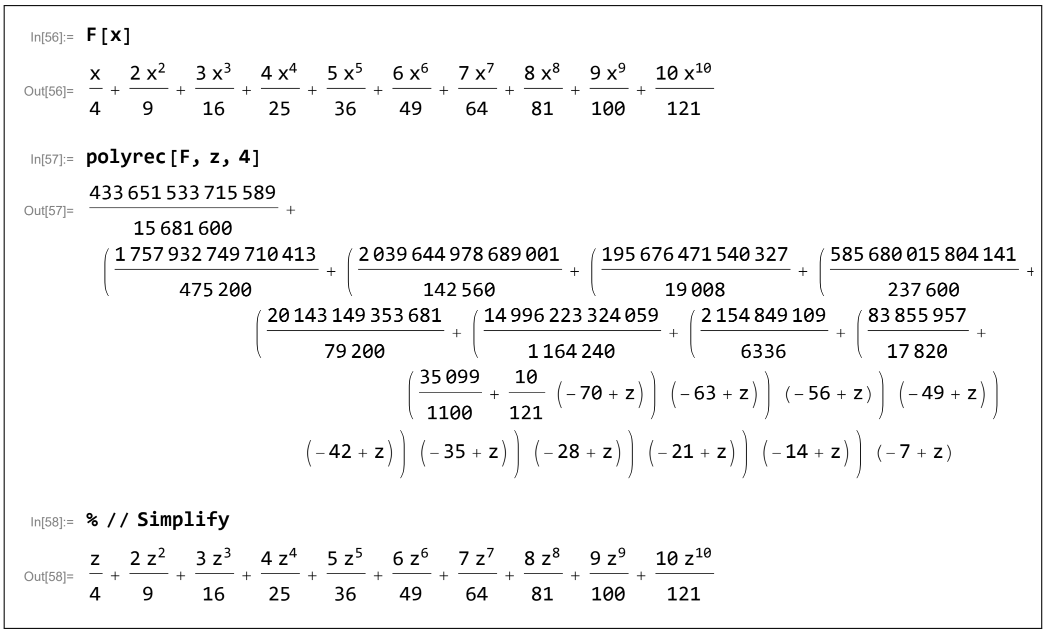
```
In[53]:= polyrec[F, z, 4]
```

Out[53]=
$$\frac{22\,284\,499\,969\,349}{410}+\Bigg(\frac{13\,038\,008\,222\,328\,208}{205}+\Bigg(\frac{179\,405\,920\,333\,501\,033}{205}+\Bigg(\frac{67\,514\,447\,400\,461\,215}{41}+\Bigg(\frac{37\,194\,960\,896\,889\,355}{41}+\Bigg(\frac{8\,411\,055\,529\,659\,223}{41}+\Bigg(\frac{6\,540\,590\,101\,485\,604}{287}+\Bigg(\frac{56\,587\,211\,320\,640}{41}+\Bigg(\frac{1\,985\,040\,987\,930}{41}+\Bigg(\frac{41\,569\,223\,696}{41}+\Bigg(\frac{521\,194\,674}{41}+\Bigg(\frac{3\,794\,609}{41}+\Bigg(\frac{14\,651}{41}+\frac{23}{41}(-91+z)\Bigg)(-84+z)\Bigg)(-77+z)\Bigg)(-70+z)\Bigg)(-63+z)\Bigg)(-56+z)\Bigg)(-49+z)\Bigg)(-42+z)\Bigg)(-35+z)\Bigg)(-28+z)\Bigg)(-21+z)\Bigg)(-14+z)\Bigg)(-7+z)$$

```
In[54]:= % // Simplify
```

Out[54]= $\frac{1}{2}+\frac{3z^2}{5}+\frac{4z^6}{7}+\frac{23z^{13}}{41}$

Figure 3.7: Example of polynomial reconstruction. A sparse polynomial, i.e. with many vanishing coefficients.


In[56]:= `F[x]`

Out[56]=
$$\frac{x}{4}+\frac{2x^2}{9}+\frac{3x^3}{16}+\frac{4x^4}{25}+\frac{5x^5}{36}+\frac{6x^6}{49}+\frac{7x^7}{64}+\frac{8x^8}{81}+\frac{9x^9}{100}+\frac{10x^{10}}{121}$$

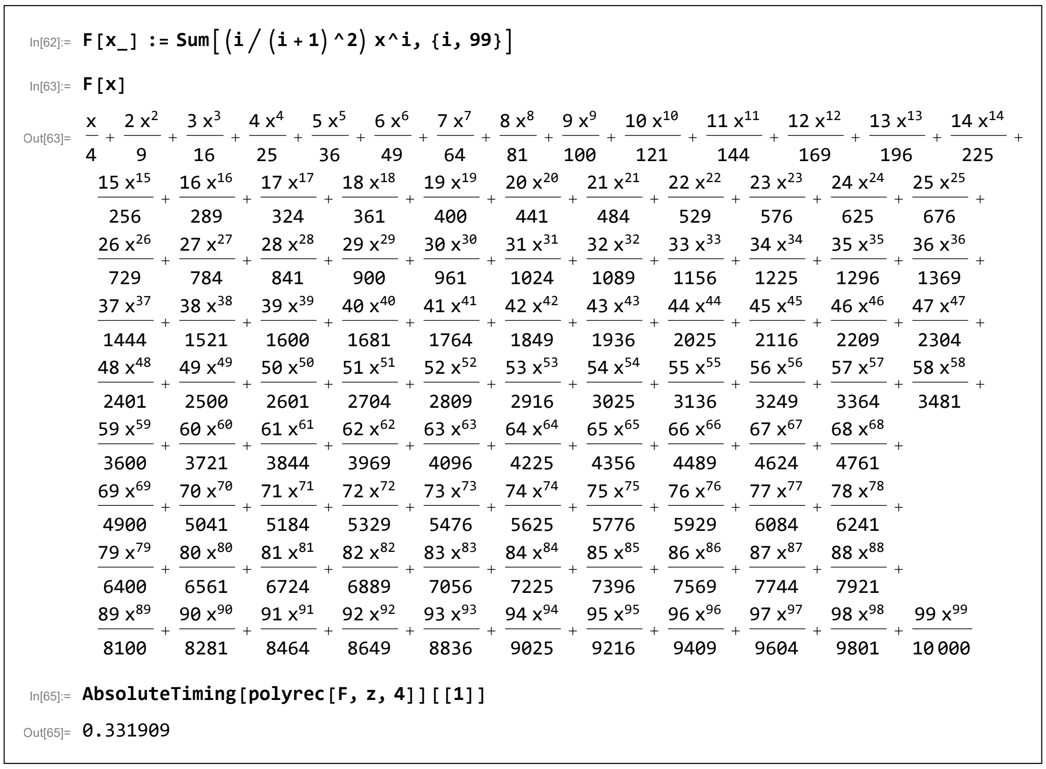
In[57]:= `polyrec[F, z, 4]`

Out[57]=
$$\frac{433\,651\,533\,715\,589}{15\,681\,600}+\left(\frac{1\,757\,932\,749\,710\,413}{475\,200}+\left(\frac{2\,039\,644\,978\,689\,001}{142\,560}+\left(\frac{195\,676\,471\,540\,327}{19\,008}+\left(\frac{585\,680\,015\,804\,141}{237\,600}+\left(\frac{20\,143\,149\,353\,681}{79\,200}+\left(\frac{14\,996\,223\,324\,059}{1\,164\,240}+\left(\frac{2\,154\,849\,109}{6336}+\left(\frac{83\,855\,957}{17\,820}+\left(\frac{35\,099}{1100}+\frac{10}{121}(-70+z)\right)(-63+z)\right)(-56+z)\right)(-49+z)\right)(-42+z)\right)(-35+z)\right)(-28+z)\right)(-21+z)\right)(-14+z)\right)(-7+z)$$

In[58]:= `% // Simplify`

Out[58]=
$$\frac{z}{4}+\frac{2z^2}{9}+\frac{3z^3}{16}+\frac{4z^4}{25}+\frac{5z^5}{36}+\frac{6z^6}{49}+\frac{7z^7}{64}+\frac{8z^8}{81}+\frac{9z^9}{100}+\frac{10z^{10}}{121}$$

Figure 3.8: Example of polynomial reconstruction. A dense polynomial with all non-vanishing coefficients up to the power 10 of the variable.


In[62]:= `F[x_] := Sum[(i/(i+1)^2) x^i, {i, 99}]`

In[63]:= `F[x]`

Out[63]=
$$\frac{x}{4}+\frac{2x^2}{9}+\frac{3x^3}{16}+\frac{4x^4}{25}+\frac{5x^5}{36}+\frac{6x^6}{49}+\frac{7x^7}{64}+\frac{8x^8}{81}+\frac{9x^9}{100}+\frac{10x^{10}}{121}+\frac{11x^{11}}{144}+\frac{12x^{12}}{169}+\frac{13x^{13}}{196}+\frac{14x^{14}}{225}+$$
$$\frac{15x^{15}}{256}+\frac{16x^{16}}{289}+\frac{17x^{17}}{324}+\frac{18x^{18}}{361}+\frac{19x^{19}}{400}+\frac{20x^{20}}{441}+\frac{21x^{21}}{484}+\frac{22x^{22}}{529}+\frac{23x^{23}}{576}+\frac{24x^{24}}{625}+\frac{25x^{25}}{676}+$$
$$\frac{26x^{26}}{729}+\frac{27x^{27}}{784}+\frac{28x^{28}}{841}+\frac{29x^{29}}{900}+\frac{30x^{30}}{961}+\frac{31x^{31}}{1024}+\frac{32x^{32}}{1089}+\frac{33x^{33}}{1156}+\frac{34x^{34}}{1225}+\frac{35x^{35}}{1296}+\frac{36x^{36}}{1369}+$$
$$\frac{37x^{37}}{1444}+\frac{38x^{38}}{1521}+\frac{39x^{39}}{1600}+\frac{40x^{40}}{1681}+\frac{41x^{41}}{1764}+\frac{42x^{42}}{1849}+\frac{43x^{43}}{1936}+\frac{44x^{44}}{2025}+\frac{45x^{45}}{2116}+\frac{46x^{46}}{2209}+\frac{47x^{47}}{2304}+$$
$$\frac{48x^{48}}{2401}+\frac{49x^{49}}{2500}+\frac{50x^{50}}{2601}+\frac{51x^{51}}{2704}+\frac{52x^{52}}{2809}+\frac{53x^{53}}{2916}+\frac{54x^{54}}{3025}+\frac{55x^{55}}{3136}+\frac{56x^{56}}{3249}+\frac{57x^{57}}{3364}+\frac{58x^{58}}{3481}+$$
$$\frac{59x^{59}}{3600}+\frac{60x^{60}}{3721}+\frac{61x^{61}}{3844}+\frac{62x^{62}}{3969}+\frac{63x^{63}}{4096}+\frac{64x^{64}}{4225}+\frac{65x^{65}}{4356}+\frac{66x^{66}}{4489}+\frac{67x^{67}}{4624}+\frac{68x^{68}}{4761}+$$
$$\frac{69x^{69}}{4900}+\frac{70x^{70}}{5041}+\frac{71x^{71}}{5184}+\frac{72x^{72}}{5329}+\frac{73x^{73}}{5476}+\frac{74x^{74}}{5625}+\frac{75x^{75}}{5776}+\frac{76x^{76}}{5929}+\frac{77x^{77}}{6084}+\frac{78x^{78}}{6241}+$$
$$\frac{79x^{79}}{6400}+\frac{80x^{80}}{6561}+\frac{81x^{81}}{6724}+\frac{82x^{82}}{6889}+\frac{83x^{83}}{7056}+\frac{84x^{84}}{7225}+\frac{85x^{85}}{7396}+\frac{86x^{86}}{7569}+\frac{87x^{87}}{7744}+\frac{88x^{88}}{7921}+$$
$$\frac{89x^{89}}{8100}+\frac{90x^{90}}{8281}+\frac{91x^{91}}{8464}+\frac{92x^{92}}{8649}+\frac{93x^{93}}{8836}+\frac{94x^{94}}{9025}+\frac{95x^{95}}{9216}+\frac{96x^{96}}{9409}+\frac{97x^{97}}{9604}+\frac{98x^{98}}{9801}+\frac{99x^{99}}{10\,000}$$

In[65]:= `AbsoluteTiming[polyrec[F, z, 4]][[1]]`

Out[65]= `0.331909`

Figure 3.9: Example of polynomial reconstruction. A dense polynomial with all non-vanishing coefficients up to the power 99 of the variable. The reconstruction output could not be displayed since in Newton's form it would cover several pages. Instead we displayed the time it took Mathematica to execute the reconstruction.

3.2 Univariate rational functions

The reconstruction process follows along the same line as that for univariate polynomials using the Thiele interpolation formula instead of Newton's form for polynomials. For rational functions however particular care is needed at each evaluation of any quantity, since either real or spurious singularities may appear. These need to be recognized and the associated evaluation point be discarded in order for the computation to yield the correct result.

The notation used in this section is the same as in section 2.1.2.

The ingredients needed to perform the reconstruction are again

- a local function called `F` which stores the values of `Fin` computed along the process
- a function `h` generating the sampling points
- a local function `f` which is the function we are reconstructing
- an auxiliary function `l` which allows to generate the analytic form of `f`
- an auxiliary function `g` which generates recursively the analytic form of the coefficients a_i
- a function `a[i]` storing the numerical value computed for each of the coefficients a_i

There is nothing new to add about the function `F`, the definition used is the same as in section 3.1.

Sampling points

Also in the case of rational function reconstruction, due to how Thiele's interpolation formula is constructed, the sampling points are arbitrary and are thus generated in the simplest way possible, i.e. as successive integers

```
h[0] = 1;
h[i_] := h[i] = h[i - 1] + 1;
```

For polynomials we pointed out that the sampling points could not be repeated, else a singularity would appear in some coefficient a_i. This was a problem due to the fact that in the polynomial reconstruction algorithm we do not expect infinities to appear, so in the code no means of dealing with a singularity is included.

Considering the analytic expressions of the coefficients defining a rational function in Thiele form, we see that repeated sampling points this time lead to indeterminate forms $\frac{0}{\infty}$.Consider for example figure 3.11, taking `h[1]=h[0]` leads to `F[h[1]]=F[h[0]]` and `a[1]` $= \frac{0}{\infty}$. Thus also for Thiele's interpolation formula repeated sampling points must be avoided.

The issues related to the repeating `h[i]` may have been inferred a priori, and this is most clear if we consider reconstructing any given function using a system-solving approach. Suppose to be a given an unknown univariate polynomial of degree 5, then one would need to solve a system of 6 *independent* equations of the form

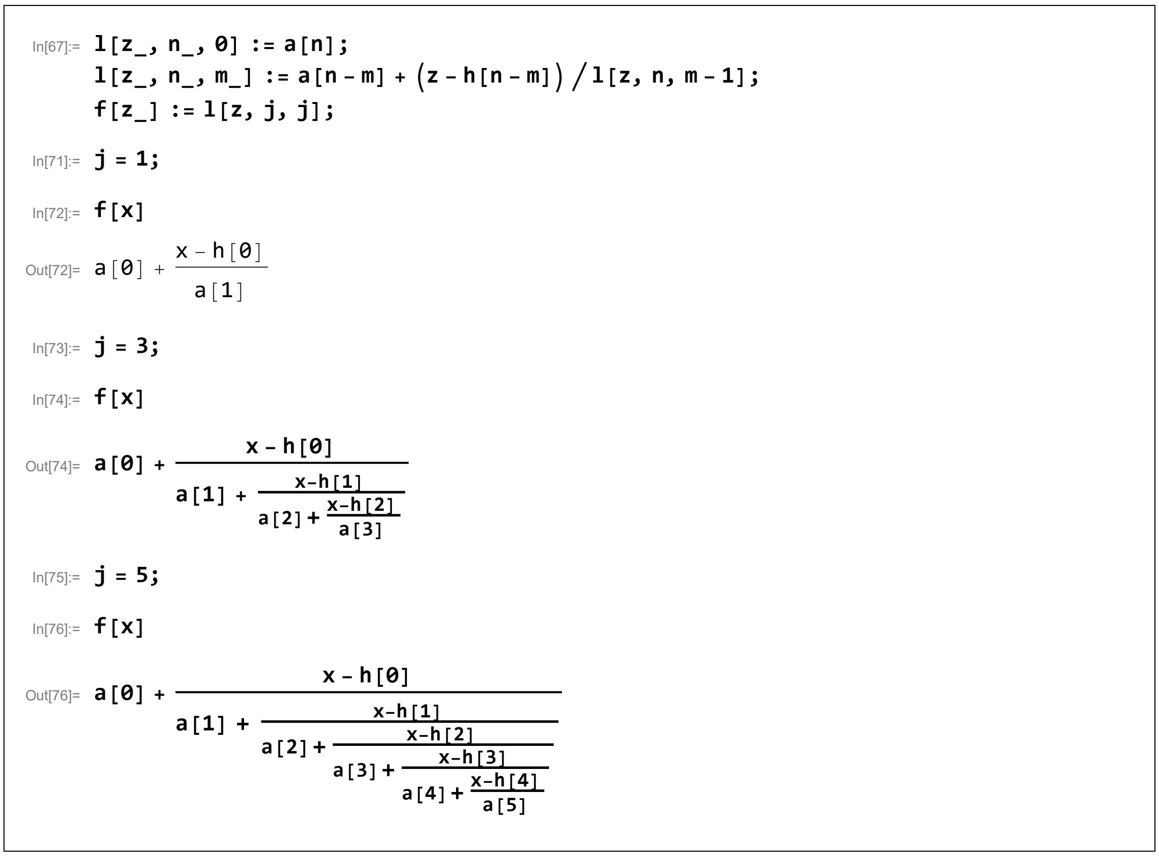
$$\begin{cases} a_0 + a_1(\mathtt{h[1]})^1 + a_2(\mathtt{h[1]})^2 + a_3(\mathtt{h[1]})^3 + a_4(\mathtt{h[1]})^4 + a_5(\mathtt{h[1]})^5 = \mathtt{F[h[1]]} \\ \qquad\qquad\vdots \\ a_0 + a_1(\mathtt{h[6]})^1 + a_2(\mathtt{h[6]})^2 + a_3(\mathtt{h[6]})^3 + a_4(\mathtt{h[6]})^4 + a_5(\mathtt{h[6]})^5 = \mathtt{F[h[6]]} \end{cases} \tag{3.2}$$

Considering two coinciding sampling points means that two of these equations coincide and the system cannot be solved. Redundant sampling points means redundant information, and this does not change for any reconstruction method used.

Despite being easily avoidable on $\mathbb{Q}$, the repetition of sampling points may be a problem impossible to overcome on a finite field $\mathbb{Z}_p$. Consider for example a function which has ten non-vanishing coefficients, then we need at least ten sampling points. If we tried to perform the reconstruction on $\mathbb{Z}_7$ we had only seven sampling points at our disposal, since the entire field is composed of only seven elements. So repetition could not be avoided and the reconstruction would be impossible. This means that given a certain function, and so a certain number of coefficients to be computed, there is a lower bound on the size of the field we can use to perform reconstruction. However in this work we usually consider fields of order 2^{53} and so there are no redundancy issues.

The test function `f`

To generate the analytic expression of the function `f` in Thiele form we use the auxiliary function `l`. This is defined recursively and depends on the coefficients `a[i]` and sampling points `h[i]`:



```
In[67]:= l[z_, n_, 0] := a[n];
         l[z_, n_, m_] := a[n - m] + (z - h[n - m]) / l[z, n, m - 1];
         f[z_] := l[z, j, j];

In[71]:= j = 1;

In[72]:= f[x]
```

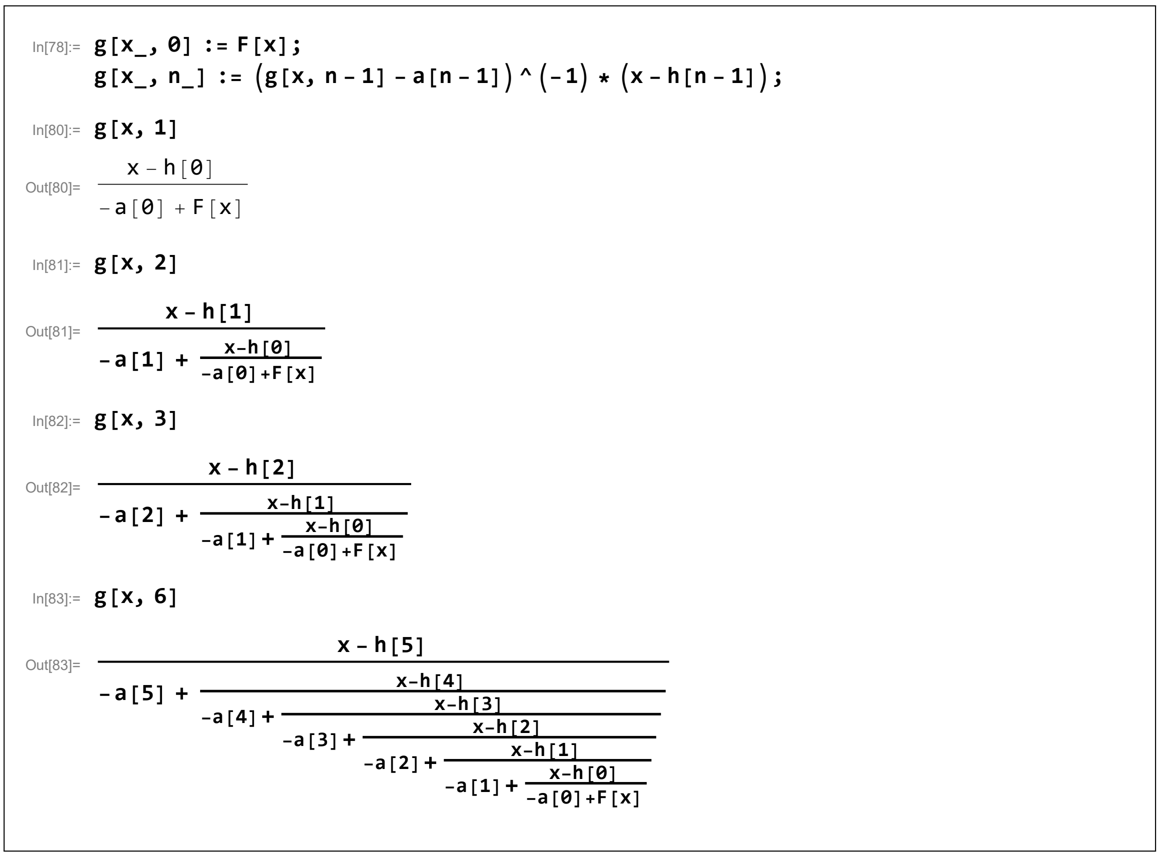
Out[72]=
$$a[0] + \frac{x - h[0]}{a[1]}$$

```
In[73]:= j = 3;

In[74]:= f[x]
```

Out[74]=
$$a[0] + \cfrac{x - h[0]}{a[1] + \cfrac{x-h[1]}{a[2] + \cfrac{x-h[2]}{a[3]}}}$$

```
In[75]:= j = 5;

In[76]:= f[x]
```

Out[76]=
$$a[0] + \cfrac{x - h[0]}{a[1] + \cfrac{x-h[1]}{a[2] + \cfrac{x-h[2]}{a[3] + \cfrac{x-h[3]}{a[4] + \cfrac{x-h[4]}{a[5]}}}}}$$



```
In[78]:= g[x_, 0] := F[x];
         g[x_, n_] := (g[x, n - 1] - a[n - 1])^(-1) * (x - h[n - 1]);

In[80]:= g[x, 1]
```

$$\text{Out[80]=}\ \frac{x-h[0]}{-a[0]+F[x]}$$

```
In[81]:= g[x, 2]
```

$$\text{Out[81]=}\ \frac{x-h[1]}{-a[1]+\frac{x-h[0]}{-a[0]+F[x]}}$$

```
In[82]:= g[x, 3]
```

$$\text{Out[82]=}\ \frac{x-h[2]}{-a[2]+\frac{x-h[1]}{-a[1]+\frac{x-h[0]}{-a[0]+F[x]}}}$$

```
In[83]:= g[x, 6]
```

$$\text{Out[83]=}\ \frac{x-h[5]}{-a[5]+\frac{x-h[4]}{-a[4]+\frac{x-h[3]}{-a[3]+\frac{x-h[2]}{-a[2]+\frac{x-h[1]}{-a[1]+\frac{x-h[0]}{-a[0]+F[x]}}}}}}$$

Figure 3.10: Definition of the auxiliary function `g` and some examples.

Alongside the definitions three examples of `f[x]` are reported. They show `f` at the step `j=1,3,5` of the computation, i.e. after coefficients up to `a[1]`,`a[3]` and `a[5]` have been determined. If the computation was actually performed the `a[i]` would all be replaced by the appropriate rational numbers.

The coefficients `a` and the function `g`

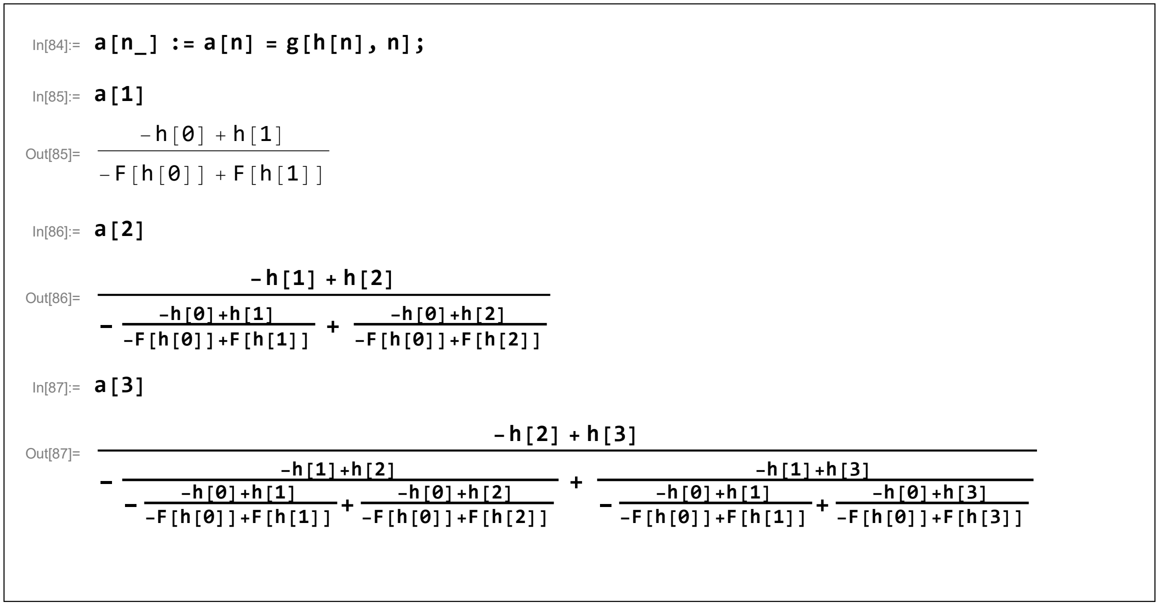
The coefficients a_i define the function `f`. Their analytic form is computed recursively using the auxiliary function `g` as shown in figure 3.10. The $g[x,n]$ is the a_n coefficient expressed in terms of all the a_i with $i < n$. The printed example outputs correspond to a_1,a_2,a_3 and a_6. The expressions shown still display their analytical dependence on `F` and `h`, because these have not been specified when printing the examples in order for the latter to be as general as possible.

The numerical value of the n-th coefficient computed at the n-th sampling point `h[n]` is stored in `a[n]`, see figure 3.11. The examples printed correspond to a_1,a_2,a_3 previously shown in the examples of `g`. We did not display the value for $i = 6$ because of the large size of the output. Again all the values are analytical since nor the function `F` nor `h` have been specified.

Vanishing denominators

Along the computation singularities due to vanishing denominators may appear.

There are three types of infinities:



```
In[84]:= a[n_] := a[n] = g[h[n], n];

In[85]:= a[1]
```

Out[85]=
$$\frac{-h[0]+h[1]}{-F[h[0]]+F[h[1]]}$$

```
In[86]:= a[2]
```

Out[86]=
$$\frac{-h[1]+h[2]}{-\frac{-h[0]+h[1]}{-F[h[0]]+F[h[1]]}+\frac{-h[0]+h[2]}{-F[h[0]]+F[h[2]]}}$$

```
In[87]:= a[3]
```

Out[87]=
$$\frac{-h[2]+h[3]}{-\frac{-h[1]+h[2]}{-\frac{-h[0]+h[1]}{-F[h[0]]+F[h[1]]}+\frac{-h[0]+h[2]}{-F[h[0]]+F[h[2]]}}+\frac{-h[1]+h[3]}{-\frac{-h[0]+h[1]}{-F[h[0]]+F[h[1]]}+\frac{-h[0]+h[3]}{-F[h[0]]+F[h[3]]}}}$$

Figure 3.11: Definition of `a` in terms of `g` and some examples.

- singularities arising from the evaluation of the input function `Fin` in one of its poles
- singularities arising from the evaluation of the test function `f` in one of its poles
- spurious singularities appearing in the coefficients `a[i]` which are by construction continuous fractions

Divergences are associated to specific sampling points. Since these are completely arbitrary, each time an infinite is encountered the currently selected sampling point is discarded. Then the computation is repeated on `h[i+1]`.

The reconstruction algorithm

The complete reconstruction is performed by algorithm 3 whose flowchart is represented in figure 3.12. The termination condition, i.e. the coincidence of the reconstructed function and the black-box algorithm, is tested by `check`. This function requires three arguments: the two functions `f` and `F` whose coincidence needs to be tested and the number of points `n` on which to check coincidence before declaring the two functions to be the same. The `check` function is given by algorithm 4 and its flowchart in figure 3.13.

```
input  : Fin=black-box algorithm, n=integer
output: f=analytic form of reconstructed function

F[x]=Fin[x];
j=0;
while True do
    compute F[h[j]];
    if F[h[j]]=$\infty$ then
        h[j]=h[j+1];
        clear stored value of h[j+1];
    else
        compute a[j];
        if a[j]=$\infty$ then
            h[j]=h[j+1];
            clear stored value of h[j+1];
            clear stored value of a[j];
        else
            compute analytic form of f with the a[j] so far evaluated;
            if check[F,f,n]=True then
                Break;
            else
                j=j+1;
            end
        end
    end
end
```

Algorithm 3: Univariate rational function reconstruction algorithm `rationalrec`. The functions `a,f,h` were defined in the previous subsections.

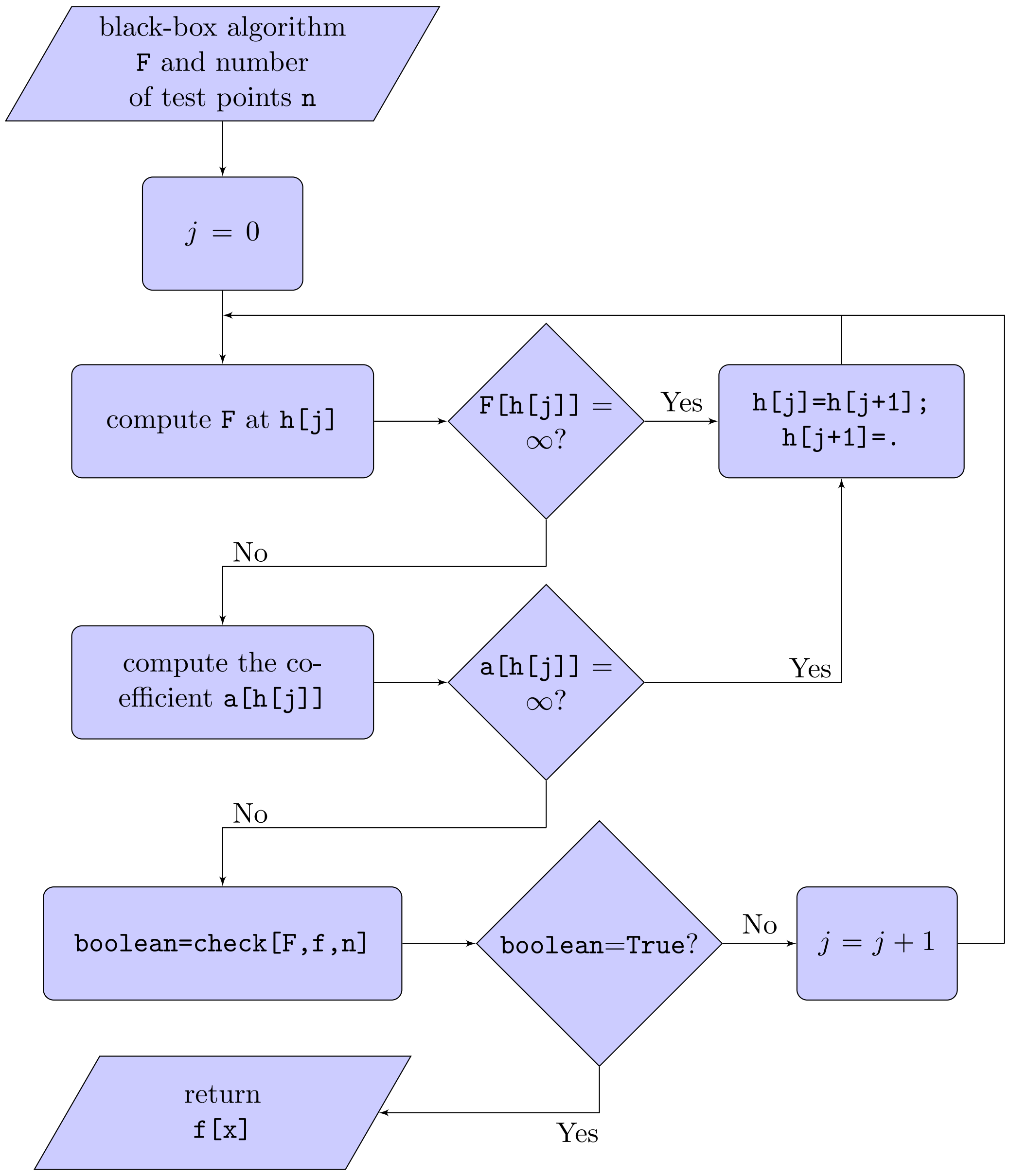


Figure 3.12: Flowchart of the univariate rational reconstruction algorithm, the definitions of `f,h,a` have been given in the previous subsections.

```
input  : F,f functions to be compared, n=integer
output: out=Boolean variable
track=h[j];
h[j]=h[j+1];
clear stored value of h[j+1];
count=0;
while $count < n$ do
    compute F[h[j]];
    compute f[h[j]];
    if F[h[j]]=$\infty$ or f[h[j]]=$\infty$ then
        h[j]=h[j+1];
        clear stored value of h[j+1];
    else
        if F[h[j]]=f[h[j]] then
            count=count+1;
            h[j]=h[j+1];
            clear stored value of h[j+1];
        else
            Break;
        end
    end
end
if $count=n$ then
    out=True;
else
    out=False;
end
h[j]=track;
```
Algorithm 4: Coincidence check function, `check`, for univariate rational function reconstruction algorithm.

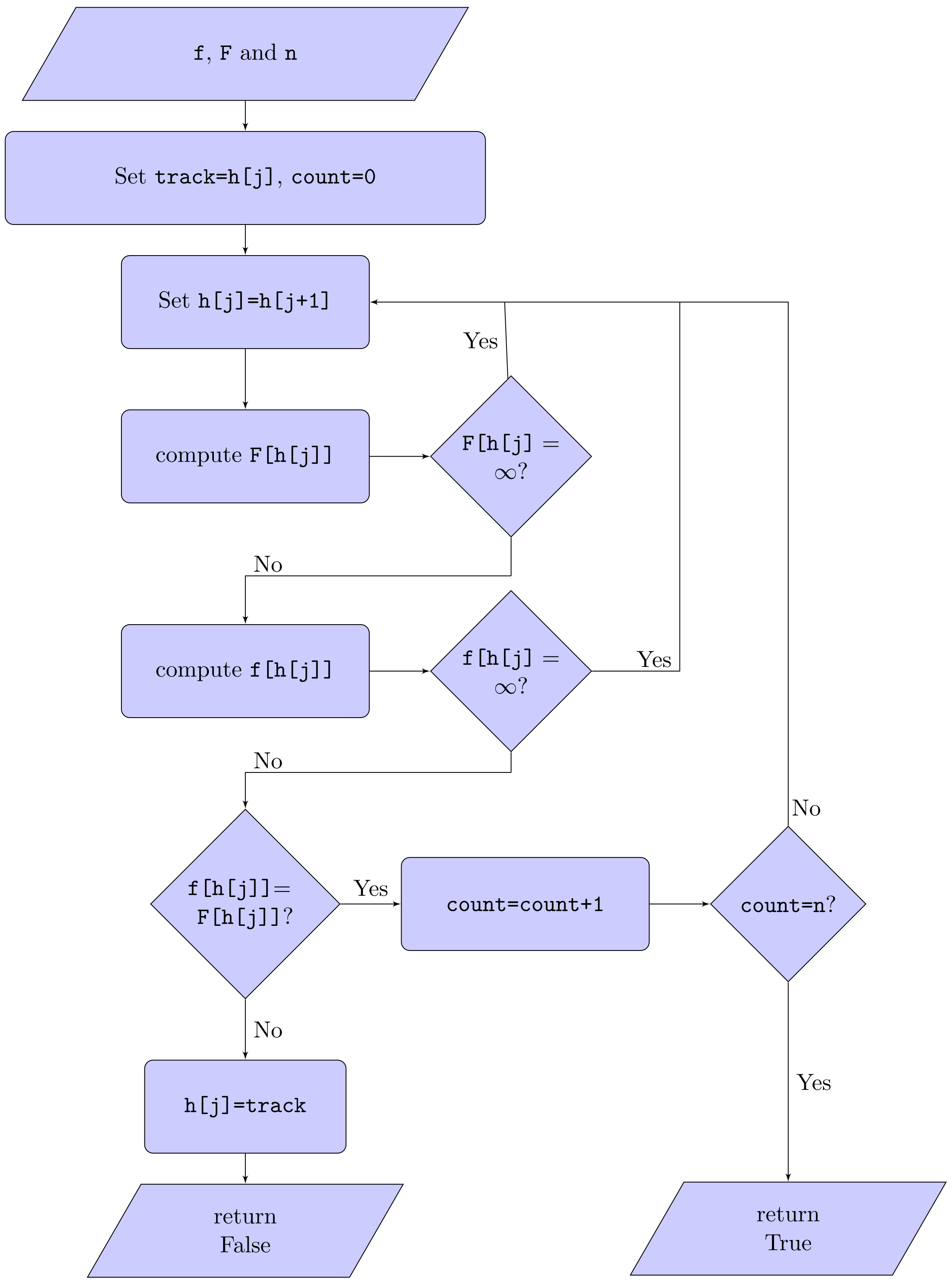
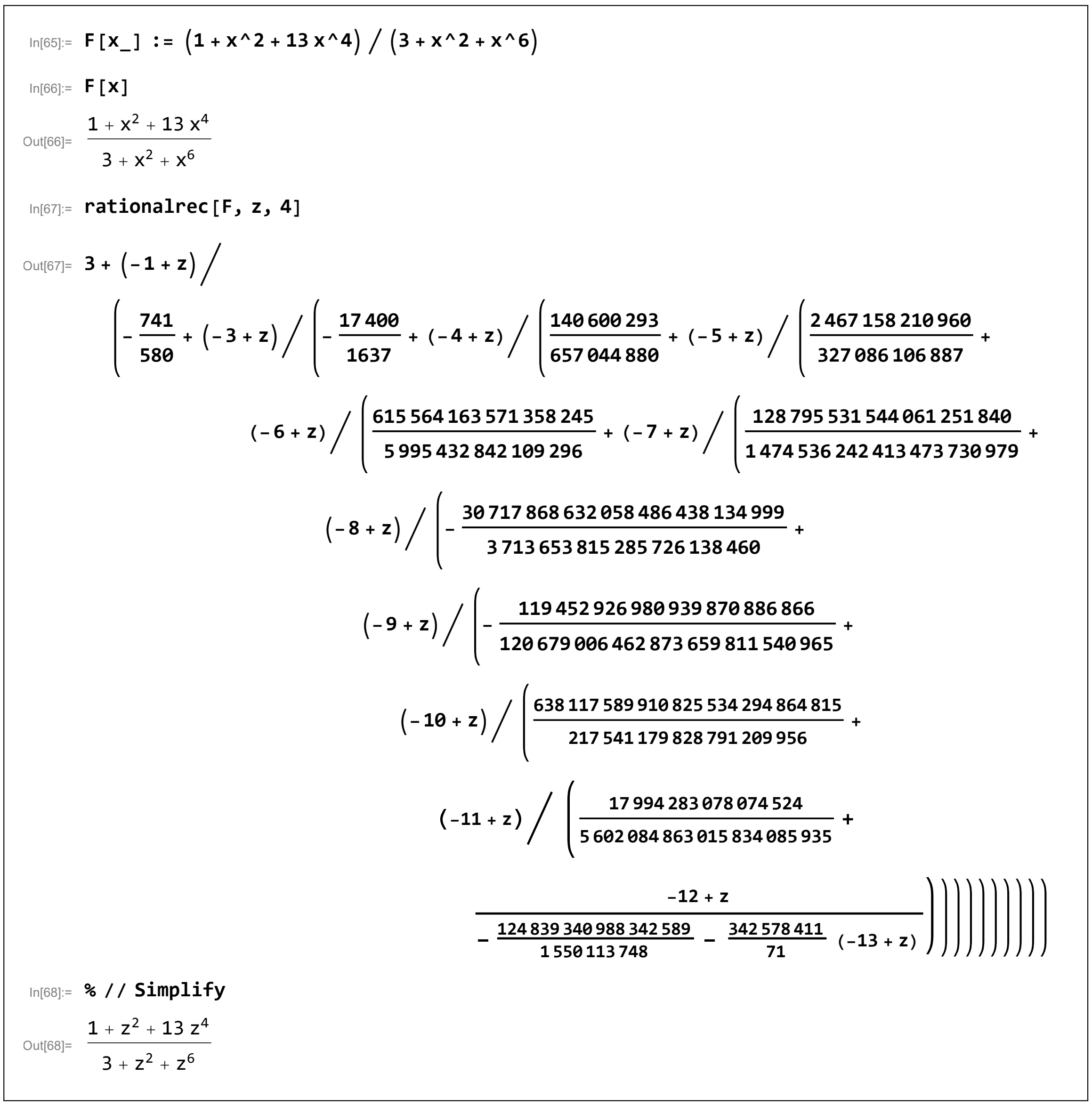


Figure 3.13: Flowchart of the check routine for the univariate rational reconstruction.

In figure 3.14 and figure 3.15 we displayed two examples of univariate rational reconstruction. Notice that the size of the coefficients involved in the Thiele form of the function is huge compared to the size of coefficients in the corresponding canonical form. This will be important when performing the reconstruction over finite fields, since the size



```
In[65]:= F[x_] := (1 + x^2 + 13 x^4) / (3 + x^2 + x^6)

In[66]:= F[x]
```

Out[66]=
$$\frac{1+x^2+13\,x^4}{3+x^2+x^6}$$

```
In[67]:= rationalrec[F, z, 4]
```

Out[67]=
$$3+(-1+z)\Bigg/\left(-\frac{741}{580}+(-3+z)\Bigg/\left(-\frac{17\,400}{1637}+(-4+z)\Bigg/\left(\frac{140\,600\,293}{657\,044\,880}+(-5+z)\Bigg/\left(\frac{2\,467\,158\,210\,960}{327\,086\,106\,887}+(-6+z)\Bigg/\left(\frac{615\,564\,163\,571\,358\,245}{5\,995\,432\,842\,109\,296}+(-7+z)\Bigg/\left(\frac{128\,795\,531\,544\,061\,251\,840}{1\,474\,536\,242\,413\,473\,730\,979}+(-8+z)\Bigg/\left(-\frac{30\,717\,868\,632\,058\,486\,438\,134\,999}{3\,713\,653\,815\,285\,726\,138\,460}+(-9+z)\Bigg/\left(-\frac{119\,452\,926\,980\,939\,870\,886\,866}{120\,679\,006\,462\,873\,659\,811\,540\,965}+(-10+z)\Bigg/\left(\frac{638\,117\,589\,910\,825\,534\,294\,864\,815}{217\,541\,179\,828\,791\,209\,956}+(-11+z)\Bigg/\left(\frac{17\,994\,283\,078\,074\,524}{5\,602\,084\,863\,015\,834\,085\,935}+\frac{-12+z}{-\frac{124\,839\,340\,988\,342\,589}{1\,550\,113\,748}-\frac{342\,578\,411}{71}(-13+z)}\right)\right)\right)\right)\right)\right)\right)\right)\right)\right)$$

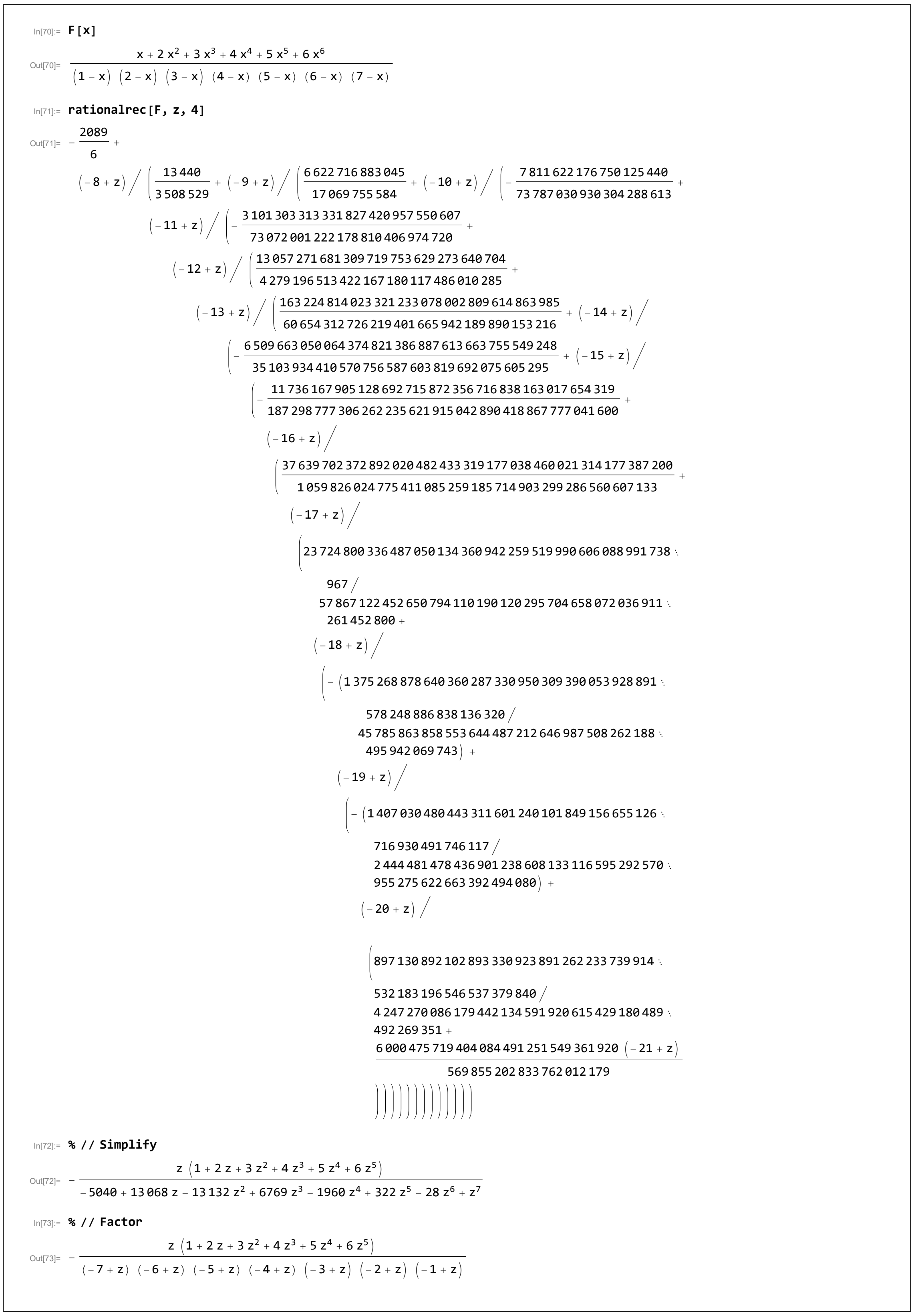
```
In[68]:= % // Simplify
```

Out[68]=
$$\frac{1+z^2+13\,z^4}{3+z^2+z^6}$$

Figure 3.14: Example of rational reconstruction, a function without real poles

of the coefficients determines the size of the field[3] over which one needs to perform the reconstruction in order for this to be successful.

[3]or equivalently the number of fields if using the Chinese remainder theorem.



```
In[70]:= F[x]

           x + 2 x^2 + 3 x^3 + 4 x^4 + 5 x^5 + 6 x^6
Out[70]= ---------------------------------------------------------
         (1 - x) (2 - x) (3 - x) (4 - x) (5 - x) (6 - x) (7 - x)

In[71]:= rationalrec[F, z, 4]

           2089
Out[71]= - ---- +
            6
 (-8 + z) / (13 440 / 3 508 529 + (-9 + z) / (6 622 716 883 045 / 17 069 755 584 + (-10 + z) / (- 7 811 622 176 750 125 440 / 73 787 030 930 304 288 613 +
   (-11 + z) / (- 3 101 303 313 331 827 420 957 550 607 / 73 072 001 222 178 810 406 974 720 +
    (-12 + z) / (13 057 271 681 309 719 753 629 273 640 704 / 4 279 196 513 422 167 180 117 486 010 285 +
     (-13 + z) / (163 224 814 023 321 233 078 002 809 614 863 985 / 60 654 312 726 219 401 665 942 189 890 153 216 + (-14 + z) /
      (- 6 509 663 050 064 374 821 386 887 613 663 755 549 248 / 35 103 934 410 570 756 587 603 819 692 075 605 295 + (-15 + z) /
       (- 11 736 167 905 128 692 715 872 356 716 838 163 017 654 319 / 187 298 777 306 262 235 621 915 042 890 418 867 777 041 600 +
        (-16 + z) /
         (37 639 702 372 892 020 482 433 319 177 038 460 021 314 177 387 200 / 1 059 826 024 775 411 085 259 185 714 903 299 286 560 607 133 +
          (-17 + z) /
           (23 724 800 336 487 050 134 360 942 259 519 990 606 088 991 738 ⋱
              967 /
             57 867 122 452 650 794 110 190 120 295 704 658 072 036 911 ⋱
              261 452 800 +
            (-18 + z) /
             (- (1 375 268 878 640 360 287 330 950 309 390 053 928 891 ⋱
                    578 248 886 838 136 320 /
                   45 785 863 858 553 644 487 212 646 987 508 262 188 ⋱
                    495 942 069 743) +
              (-19 + z) /
               (- (1 407 030 480 443 311 601 240 101 849 156 655 126 ⋱
                     716 930 491 746 117 /
                     2 444 481 478 436 901 238 608 133 116 595 292 570 ⋱
                     955 275 622 663 392 494 080) +
                (-20 + z) /
                 (897 130 892 102 893 330 923 891 262 233 739 914 ⋱
                  532 183 196 546 537 379 840 /
                  4 247 270 086 179 442 134 591 920 615 429 180 489 ⋱
                  492 269 351 +
                  6 000 475 719 404 084 491 251 549 361 920 (-21 + z)
                  ----------------------------------------------------
                               569 855 202 833 762 012 179
                  )))))))))))))

In[72]:= % // Simplify

                      z (1 + 2 z + 3 z^2 + 4 z^3 + 5 z^4 + 6 z^5)
Out[72]= - -------------------------------------------------------------------
           -5040 + 13 068 z - 13 132 z^2 + 6769 z^3 - 1960 z^4 + 322 z^5 - 28 z^6 + z^7

In[73]:= % // Factor

                     z (1 + 2 z + 3 z^2 + 4 z^3 + 5 z^4 + 6 z^5)
Out[73]= - ---------------------------------------------------------------
           (-7 + z) (-6 + z) (-5 + z) (-4 + z) (-3 + z) (-2 + z) (-1 + z)
```

Figure 3.15: Example of rational reconstruction, a function with seven real poles.

3.3 Operations over finite fields

Before talking about operations over finite fields we will deal with the extended Euclidean algorithm, since this will be needed when computing the multiplicative inverse of an element in $\mathbb{Z}_p$ as well as when trying to invert the mapping.

Extended Euclidean algorithm

Recall that the extended Euclidean algorithm allows to compute the greatest common divisor g of two numbers a, b, and at the same time the coefficients m, n such that

$$am + bn = c \tag{3.3}$$

Starting first only with the greatest common divisor, considering $a > b$ we set

$$\begin{cases} \texttt{r[0]=a} \\ \texttt{r[1]=b} \end{cases} \tag{3.4}$$

where the succession `r[i]` of the remainders is defined by

$$\texttt{r[i+1]=r[i-1]-q[i]r[i]} \tag{3.5}$$

and the successive quotients `q[i]` by[4]

$$\texttt{q[i]=}\left\lfloor \frac{\texttt{r[i-1]}}{\texttt{r[i]}} \right\rfloor \tag{3.6}$$

The termination condition is given by

$$\texttt{r[n]=0} \tag{3.7}$$

and the output, which is the greatest common divisor, will be `r[n-1]`.

The integers m and n can be computed by adding other two successions, called `s[i]` and `t[i]` as in eq. (2.37), defined by the initial conditions

$$\begin{cases} \texttt{s[0]=1} \\ \texttt{s[1]=0} \end{cases} \qquad \begin{cases} \texttt{t[0]=0} \\ \texttt{t[1]=1} \end{cases} \tag{3.8}$$

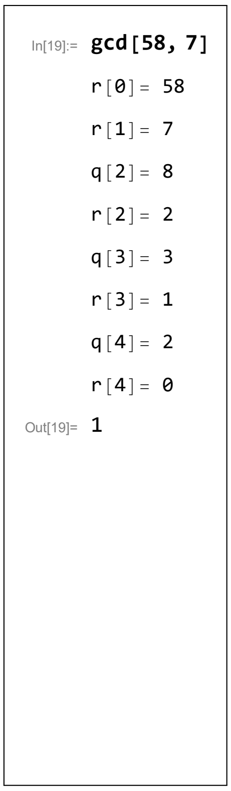
and the recursive relations

$$\texttt{s[i+1]=s[i-1]-q[i]s[i]} \tag{3.9}$$
$$\texttt{t[i+1]=t[i-1]-q[i]t[i]} \tag{3.10}$$

once the termination condition eq. (3.7) is reached the output will be $\gcd(a, b) =$ `r[n-1]` and $m =$ `s[i-1]`, $n =$ `t[i-1]`, where

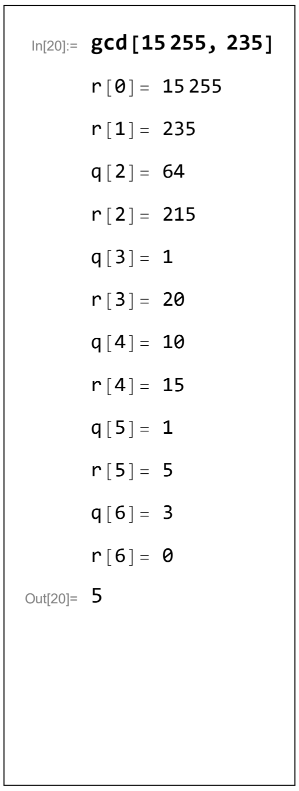
$$am + bn = \gcd(a, b) \tag{3.11}$$

[4] $\lfloor x \rfloor$ is the integer part of x.



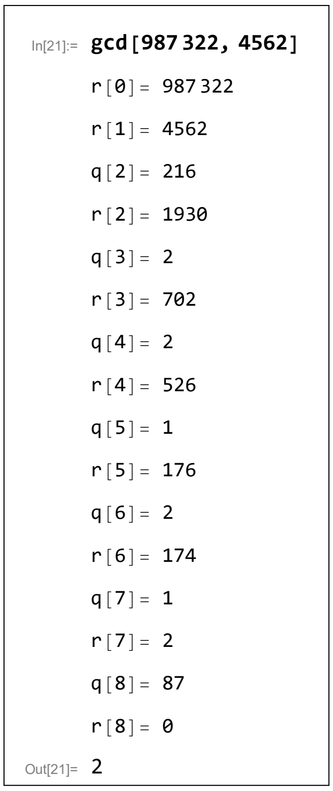
```
In[19]:= gcd[58, 7]
         r[0]= 58
         r[1]= 7
         q[2]= 8
         r[2]= 2
         q[3]= 3
         r[3]= 1
         q[4]= 2
         r[4]= 0
Out[19]= 1
```

(a) g.c.d.(58,7)



```
In[20]:= gcd[15 255, 235]
         r[0]= 15 255
         r[1]= 235
         q[2]= 64
         r[2]= 215
         q[3]= 1
         r[3]= 20
         q[4]= 10
         r[4]= 15
         q[5]= 1
         r[5]= 5
         q[6]= 3
         r[6]= 0
Out[20]= 5
```

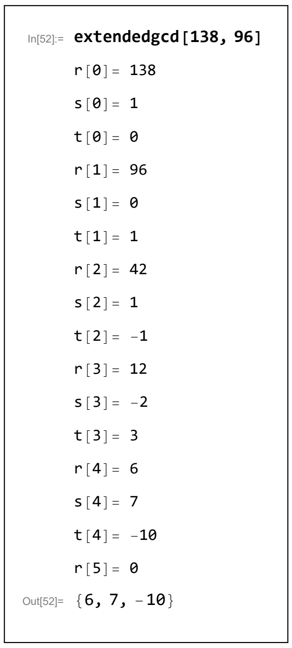
(b) g.c.d.(15255,235)



```
In[21]:= gcd[987 322, 4562]
         r[0]= 987 322
         r[1]= 4562
         q[2]= 216
         r[2]= 1930
         q[3]= 2
         r[3]= 702
         q[4]= 2
         r[4]= 526
         q[5]= 1
         r[5]= 176
         q[6]= 2
         r[6]= 174
         q[7]= 1
         r[7]= 2
         q[8]= 87
         r[8]= 0
Out[21]= 2
```

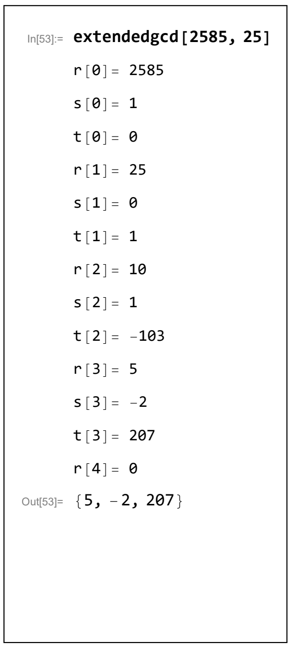
(c) g.c.d.(987322,4562)

Figure 3.16: Some examples of application of the Euclidean algorithm to compute the greatest common divisor.



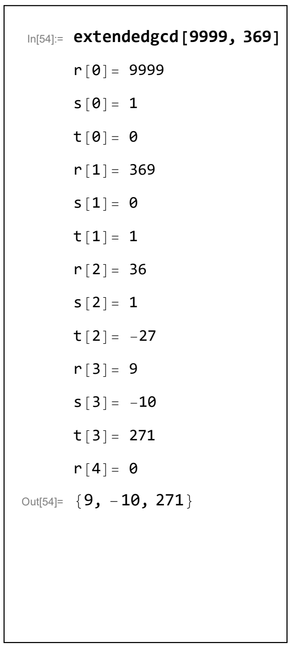
```
In[52]:= extendedgcd[138, 96]
         r[0]= 138
         s[0]= 1
         t[0]= 0
         r[1]= 96
         s[1]= 0
         t[1]= 1
         r[2]= 42
         s[2]= 1
         t[2]= -1
         r[3]= 12
         s[3]= -2
         t[3]= 3
         r[4]= 6
         s[4]= 7
         t[4]= -10
         r[5]= 0
Out[52]= {6, 7, -10}
```

(a)



```
In[53]:= extendedgcd[2585, 25]
         r[0]= 2585
         s[0]= 1
         t[0]= 0
         r[1]= 25
         s[1]= 0
         t[1]= 1
         r[2]= 10
         s[2]= 1
         t[2]= -103
         r[3]= 5
         s[3]= -2
         t[3]= 207
         r[4]= 0
Out[53]= {5, -2, 207}
```

(b)



```
In[54]:= extendedgcd[9999, 369]
         r[0]= 9999
         s[0]= 1
         t[0]= 0
         r[1]= 369
         s[1]= 0
         t[1]= 1
         r[2]= 36
         s[2]= 1
         t[2]= -27
         r[3]= 9
         s[3]= -10
         t[3]= 271
         r[4]= 0
Out[54]= {9, -10, 271}
```

(c)

Figure 3.17: Some examples of application of the extended Euclidean algorithm. The output has the form {`r[n-1]`,`s[n-1]`,`t[n-1]`} where `r[n]=0`.

Map from $\mathbb{Q}$ to $\mathbb{Z}_p$

The first operation which needs to be implemented is the mapping from $\mathbb{Z}$ to $\mathbb{Z}_p$, which will be called $F_{\mathbb{Z}}$.

Recall that $\mathbb{Z}_p = \{1, \cdots, p-1\}$ and each of its elements is an equivalence class defined by the relation

$$a \simeq b \quad \Leftrightarrow \quad a = b + np \quad \exists n \in \mathbb{Z} \tag{3.12}$$

in other words a belongs to the equivalence class $[b]_p \in \mathbb{Z}_p$ if the remainder of the division $\frac{a}{p}$ is b. In eq. (3.13) we reported some simple examples.

p	a	$\lfloor \frac{a}{p} \rfloor$	remainder	$a = b \mod p$
7	15	2	1	$15 = 1 \mod 7$
10	9	0	9	$9 = 9 \mod 10$
17	51	3	0	$51 = 0 \mod 17$
23	10^6	43478	6	$10^6 = 6 \mod 23$
23	10^9	4.3478×10^7	4	$10^9 = 4 \mod 23$
23	132546879841	5762907819	4	$132546879841 = 4 \mod 23$

(3.13)

The associated routine will be called `mod[p_,z_]` and takes as input the order p of the ring $\mathbb{Z}_p$ and an element of $\mathbb{Z}$. So far p needs not to be a prime number.

Now we need to extend this definition to a map from $\mathbb{Q}$ to $\mathbb{Z}_p$. This will be achieved in two steps, consider the rational $z = \frac{n}{d} = n \times d^{-1}$, first one computes the multiplicative inverse of d over $\mathbb{Z}_p$, then we can simply multiply it by n and taking the modulo of this quantity will give us the image of z in $\mathbb{Z}_p$. The multiplicative inverse is defined as

$$c = d^{-1} \quad \Leftrightarrow \quad c \times d = 1 \mod p \tag{3.14}$$

Recall that p needs to be a prime number for the multiplicative inverse to be defined for any element of $\mathbb{Z}_p$, see section 2.2.1. So starting from $\frac{1}{d} \in \mathbb{Q}$ we first map d onto $\mathbb{Z}_p$, then look for the element $c \in \mathbb{Z}_p$ such that eq. (3.14) is true. To compute c we start from the Bezout identity eq. (3.3), take $a = d$, $b = p$ then the greatest common divisor becomes $g = 1$ being p prime. We then get

$$dm + np = 1 \quad \Leftrightarrow \quad dm = 1 - np \quad \Rightarrow \quad d \times m = 1 \mod p \tag{3.15}$$

thus the coefficient m is the multiplicative inverse of d over $\mathbb{Z}_p$. Call the associated routine `multinv[p_,d_]`, it takes as input the integer d and the order of the field p and returns as output the coefficient m after running the extended Euclidean algorithm for d and p.

Now we can build the map from $\mathbb{Q}$ to $\mathbb{Z}_p$

$$\begin{aligned} F_{\mathbb{Q}} : \mathbb{Q} &\to \mathbb{Z}_p \\ \frac{n}{d} &\mapsto (n \cdot (d^{-1} \mod p) \mod p) \end{aligned} \tag{3.16}$$

calling the associated routine `ffmap[p_,z_]` with $z = \frac{n}{d}$ we have:

$$\texttt{ffmap[p\_,z\_]=mod[p,mod[p,n]} \times \texttt{multinv[p,mod[p,d]]]} \tag{3.17}$$

We will define this map symbolically as `ffmap[p,x]` $= [x]_p$.

Let us compute for example the image of $\left[\frac{25}{12}\right]_7$ on $\mathbb{Z}_7$:

$$\begin{aligned}&\texttt{mod[p,d]=mod[7,12]=5;}\\ &\texttt{multinv[p,mod[p,d]]=multinv[7,5]=3;}\\ &\texttt{mod[p,n]=mod[7,25]=4;}\\ &\texttt{ffmap[p,z]=mod[7,4}\times\texttt{3]=5;}\end{aligned} \tag{3.18}$$

Consider as an another example $\left[\frac{96}{275}\right]_{31}$ on $\mathbb{Z}_{31}$:

$$\begin{aligned}&\texttt{mod[p,d]=mod[31,275]=27;}\\ &\texttt{multinv[p,mod[p,d]]=multinv[31,27]=23;}\\ &\texttt{mod[p,n]=mod[31,96]=3;}\\ &\texttt{ffmap[p,z]=mod[31,3}\times\texttt{23]=7;}\end{aligned} \tag{3.19}$$

We remark again that it may happen that the denominator of the rational to be mapped to $\mathbb{Z}_p$ is a multiple of p, thus $d = 0 \mod p$. This is a problem since the identity under addition, i.e. 0, does not admit a multiplicative inverse on any field. So this particular class of rationals cannot be mapped to $\mathbb{Z}_p$. However, as will be explained in section 3.4 this does not represent a problem when performing the reconstruction.

Addition over $\mathbb{Z}_p$

Once the mapping to $\mathbb{Z}_p$ has been performed, we need to implement the two binary operations characterizing the field, addition and multiplication $\mod p$. We want to build them in terms of ordinary addition and multiplication. To do so, one needs to use the fact that the map $F_\mathbb{Q}$ is a ring homomorphism and does preserve these two operations.[5] This means that performing for example the addition of two elements $a, b \in \mathbb{Q}$ and then mapping the outcome to $\mathbb{Z}_p$ yields the same result as directly mapping a, b to $\mathbb{Z}_p$, and then using the addition defined on $\mathbb{Z}_p$ to sum their images. Calling $+$ the addition over $\mathbb{Q}$ and $+_p$ the addition over $\mathbb{Z}_p$:

$$F_\mathbb{Q}(a + b) = F_\mathbb{Q}(a) +_p F_\mathbb{Q}(b) \tag{3.20}$$

Suppose then to be given $x, y \in \mathbb{Z}_p$ we want to compute

$$c = x +_p y \tag{3.21}$$

what we know is that

$$c = F_\mathbb{Q}(a + b) \tag{3.22}$$

where a, b are two arbitrary elements of $\mathbb{Q}$ such that

$$\begin{cases} F_\mathbb{Q}(a) = x \\ F_\mathbb{Q}(b) = y \end{cases} \tag{3.23}$$

In other words a, b are simply two arbitrary representatives of the equivalence classes $x \mod p$ and $y \mod p$. We may, and will, thus choose $x \in \mathbb{Q}$ and $y \in \mathbb{Q}$ as these representatives, perform the addition on $\mathbb{Q}$ and then map the result to $\mathbb{Z}_p$.

For example consider $9, 11 \in \mathbb{Z}_{13}$:

$$c = 9 +_{13} 11 \tag{3.24}$$

[5] See definition A.5.

take then as representatives $9, 11 \in \mathbb{Q}$ and perform the sum in $\mathbb{Q}$

$$9 + 11 = 20 \ \in \ \mathbb{Q} \tag{3.25}$$

and then map the result to $\mathbb{Z}_{13}$:

$$c = F_{\mathbb{Q}}(20) = 7 \ \in \ \mathbb{Z}_{13} \tag{3.26}$$

Or again considering $15, 17 \in \mathbb{Z}_{23}$

$$\underbrace{15 + 17}_{\in \, \mathbb{Z}_{23}} = F_{\mathbb{Q}}(\underbrace{15 + 17}_{\in \, \mathbb{Q}}) = 9 \ \in \ \mathbb{Z}_{23} \tag{3.27}$$

The associated routine will look like

```
ffsum[p_,a_,b_]=mod[p,a+b]
```
(3.28)

This can then be extended to an arbitrary number of addenda $\{a, \cdots, n\}$ in two different ways, either

```
ffsum[p_,a_,⋯,n_] = mod[p,a+⋯+n]
```
(3.29)

or

```
ffsum[p_,a_,⋯,n_]=mod[p,a+mod[p,b+mod[p,⋯+mod[p,(n-1)+n]]⋯]]
```
(3.30)

the difference being that in eq. (3.29) depending on the number and size of addenda huge numbers over $\mathbb{Q}$ may arise, whereas taking a module operation after each single addition keeps the size of involved numbers always below $2p$. Since the whole point of introducing finite fields was to keep the size of the numbers small, in order for the computation not to need arbitrary precision arithmetics, eq. (3.32) is to be preferred.

Multiplication over $\mathbb{Z}_p$

All the same reasoning presented for the addition applies to the product on $\mathbb{Z}_p$ called $\times_p$, which will thus be defined as:

```
ffprod[p_,a_,b_]=mod[p,a×b]
```
(3.31)

and extending to an arbitrary number of factors

```
ffprod[p_,a_,⋯,n_]=mod[p,a×mod[p,b×mod[p,⋯×mod[p,(n-1)×n]]⋯]]
```
(3.32)

Actually we already used implicitly the product over finite fields in `ffmap` which could be rewritten as

```
ffmap[p_,z_]=ffprod[p,mod[p,n],multinv[p,mod[p,d]]]
```
(3.33)

with $z = \frac{n}{d}$.

Map from $\mathbb{Z}_p$ to $\mathbb{Q}$

Once the amplitude has been computed over finite fields, one needs to recover the result in $\mathbb{Q}$. As already mentioned, the extended Euclidean algorithm can provide a number of different possible rationals that are mapped to a given element c in $\mathbb{Z}_p$. The algorithm produces three successions r_i, s_i, t_i which satisfy

$$as_i + bt_i = r_i \quad \forall\, i \tag{3.34}$$

where a, b are the two numbers whose greatest common divisor we are computing. Running the algorithm for $a = c$ and $b = p$ at each step of the iteration we get

$$cs_i + pt_i = r_i \quad \Rightarrow \quad \frac{r_i}{s_i} = c \mod p \tag{3.35}$$

Thus, if it took the algorithm N steps to compute the g.c.d., we will have N possible candidate fractions whose image in $\mathbb{Z}_p$ is c. Among these candidates we want to pick the one such that

$$r_i^2, s_i^2 < \frac{p}{2} \tag{3.36}$$

since this will be the unique fraction whose numerator and denominator satisfy eq. (3.36) and whose image in $\mathbb{Z}_p$ is c.

We can then build the algorithm `invmap[p_,c_]` which computes the inverse of $F_\mathbb{Q}$. It runs the extended Euclidean algorithm with `c,p` as inputs, termination condition given by eq. (3.36) and the output will be $\frac{\texttt{r[i]}}{\texttt{s[i]}}$. As already mentioned in section 2.2.2, the fact that if p is big enough, among the possible candidates provided by the extended Euclidean algorithm there is always one satisfying eq. (3.36) is proven in [31].

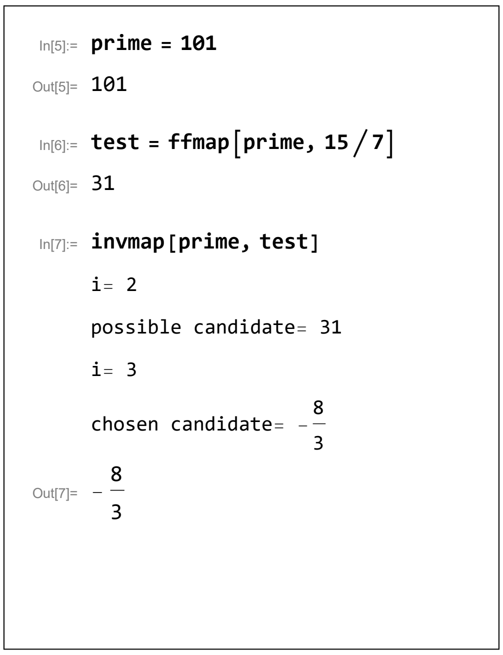
As an example consider $z = \frac{15}{7}$, we displayed the output of the inverse mapping for the fields $\mathbb{Z}_{101}$, $\mathbb{Z}_{503}$, $\mathbb{Z}_{1009}$ and $\mathbb{Z}_{5003}$, see figure 3.18. As can be seen for $p = 503$ the inverse map returns the correct result, this is due to the fact that $503 > 2 \times 15^2$ and of course $503 > 2 \times 7^2$. And similarly for $p = 1009, 5003$. Since 101 is far smaller than 2×15^2 the chances of obtaining the correct result are very low, in fact we get the wrong one.

Considering the examples shown in figure 3.19, one can see that even if eq. (3.36) is not satisfied, there is none the less a chance to get the correct result. This is due to the fact that given $z = \frac{n}{d}$ such that $F_\mathbb{Q}(z) = c \in \mathbb{Z}_p$, the more the order p of the field grows, the less rationals there are which are mapped to c and whose numerator and denominator have comparable size to n, d. As soon as eq. (3.36) is satisfied there is only one left. The chances of picking the correct inverse among the possible candidates thus grows as the order p grows, and it seems to become relevant when

$$r_i^2, s_i^2 < p \tag{3.37}$$

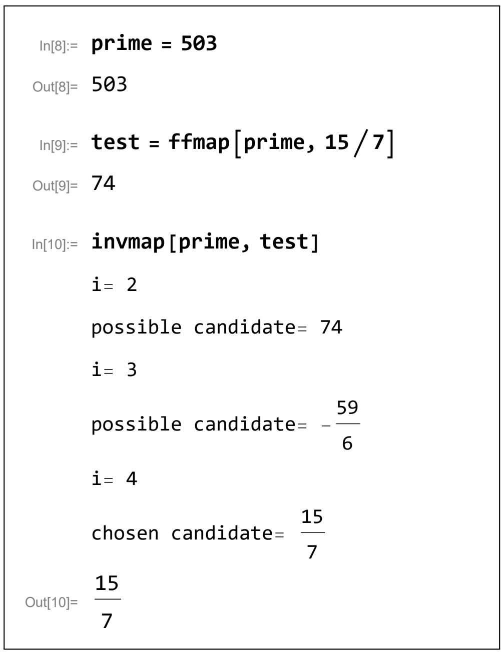
Thus we will be considering this relaxed condition eq. (3.37) instead of eq. (3.36).

As mentioned in section 2.2.3, the finite fields reconstruction is performed over multiple fields whose order is of machine size. Then the results are combined through the Chinese remainder theorem in order to get the image over the global field $\mathbb{Z}_P$ with $P = p_1 \cdots p_n$ whose order is hopefully big enough for the map $\mathbb{Z}_P \to \mathbb{Q}$ to assign to all the coefficients of the function to be reconstructed their correct images. This means that if one needs the size of P to grow, one has to perform the entire reconstruction again over a different field defined by p_{n+1}. Thus considering the relaxed condition eq. (3.37), having noticed that often the correct result is returned anyway, may greatly reduce the time needed to obtain the final result since less reconstructions are to be performed, because the size of P is smaller.



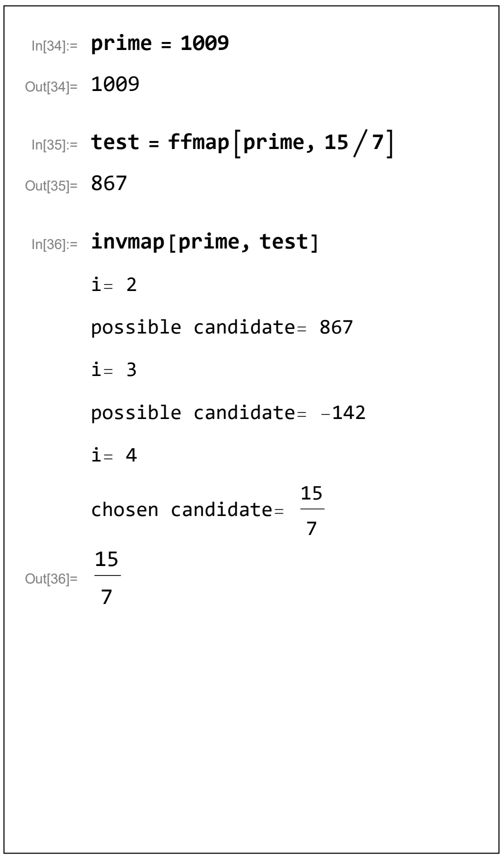
```
In[5]:=  prime = 101
Out[5]=  101
In[6]:=  test = ffmap[prime, 15/7]
Out[6]=  31
In[7]:=  invmap[prime, test]
         i= 2
         possible candidate= 31
         i= 3
         chosen candidate= -8/3
Out[7]=  -8/3
```

(a) The prime 101 is too small



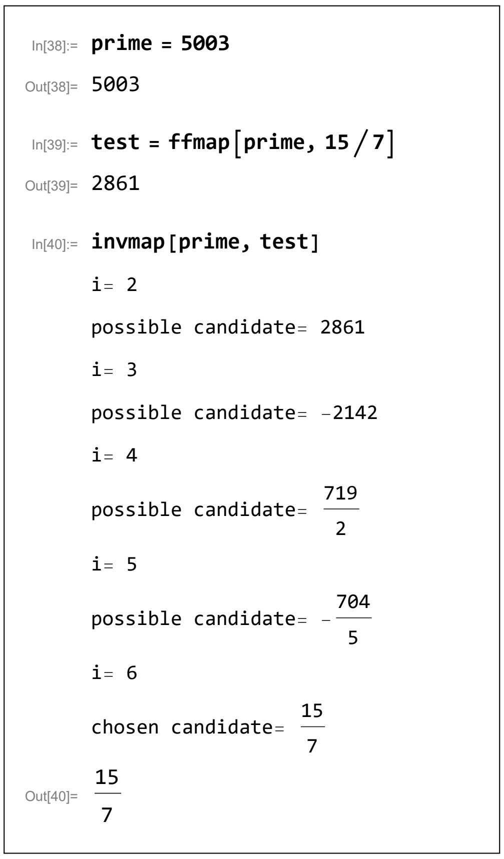
```
In[8]:=  prime = 503
Out[8]=  503
In[9]:=  test = ffmap[prime, 15/7]
Out[9]=  74
In[10]:= invmap[prime, test]
         i= 2
         possible candidate= 74
         i= 3
         possible candidate= -59/6
         i= 4
         chosen candidate= 15/7
Out[10]= 15/7
```

(b) The prime 503 is big enough



```
In[34]:= prime = 1009
Out[34]= 1009
In[35]:= test = ffmap[prime, 15/7]
Out[35]= 867
In[36]:= invmap[prime, test]
         i= 2
         possible candidate= 867
         i= 3
         possible candidate= -142
         i= 4
         chosen candidate= 15/7
Out[36]= 15/7
```

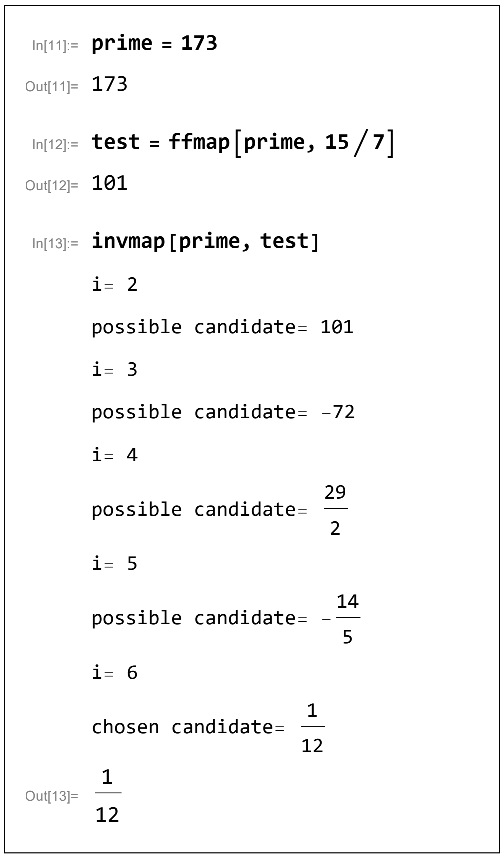
(c) The prime 1009 is more than big enough



```
In[38]:= prime = 5003
Out[38]= 5003
In[39]:= test = ffmap[prime, 15/7]
Out[39]= 2861
In[40]:= invmap[prime, test]
         i= 2
         possible candidate= 2861
         i= 3
         possible candidate= -2142
         i= 4
         possible candidate= 719/2
         i= 5
         possible candidate= -704/5
         i= 6
         chosen candidate= 15/7
Out[40]= 15/7
```

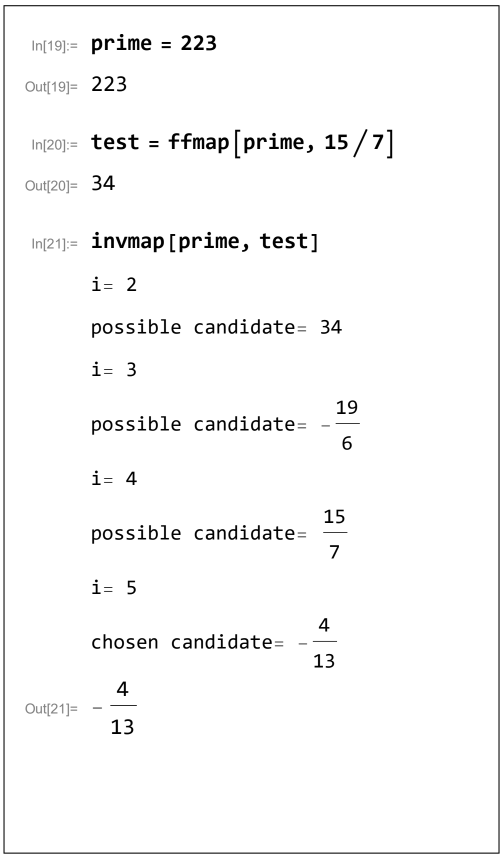
(d) The prime 5003 is more than big enough

Figure 3.18: Examples of map from $\mathbb{Z}_p$ to $\mathbb{Q}$.



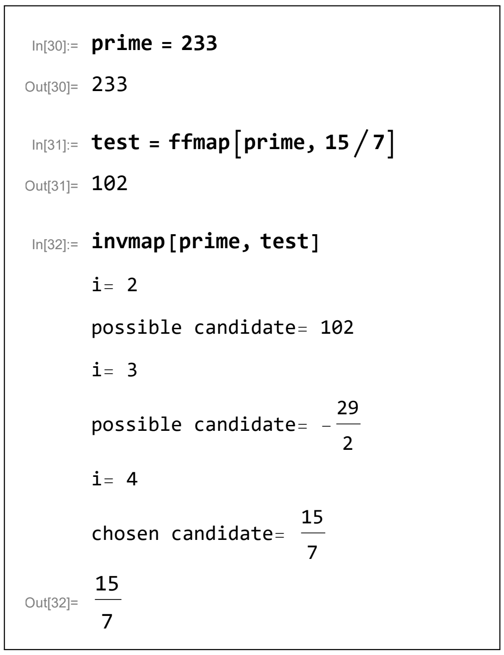
```
In[11]:= prime = 173
Out[11]= 173

In[12]:= test = ffmap[prime, 15/7]
Out[12]= 101

In[13]:= invmap[prime, test]
         i= 2
         possible candidate= 101
         i= 3
         possible candidate= -72
         i= 4
                              29
         possible candidate=  ──
                               2
         i= 5
                               14
         possible candidate= - ──
                                5
         i= 6
                            1
         chosen candidate=  ──
                            12
          1
Out[13]= ──
         12
```

(a) $p = 173 < 15^2$, the inverse is wrong



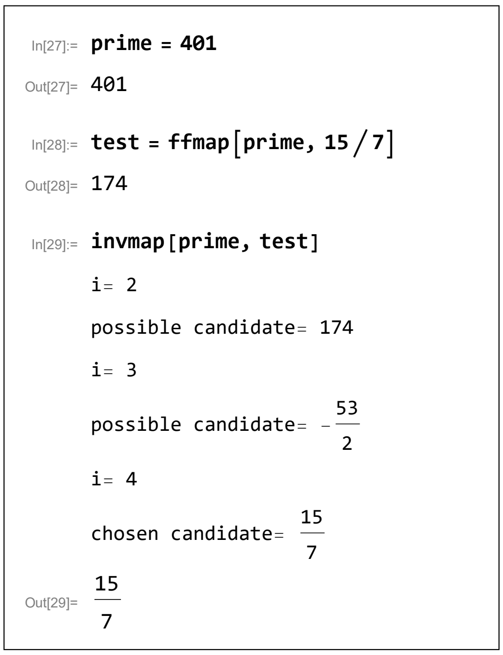
```
In[19]:= prime = 223
Out[19]= 223

In[20]:= test = ffmap[prime, 15/7]
Out[20]= 34

In[21]:= invmap[prime, test]
         i= 2
         possible candidate= 34
         i= 3
                               19
         possible candidate= - ──
                                6
         i= 4
                              15
         possible candidate=  ──
                               7
         i= 5
                             4
         chosen candidate= - ──
                             13
           4
Out[21]= - ──
           13
```

(b) $p = 223 < 15^2$, the inverse is wrong



```
In[30]:= prime = 233
Out[30]= 233

In[31]:= test = ffmap[prime, 15/7]
Out[31]= 102

In[32]:= invmap[prime, test]
         i= 2
         possible candidate= 102
         i= 3
                               29
         possible candidate= - ──
                                2
         i= 4
                            15
         chosen candidate=  ──
                             7
         15
Out[32]= ──
          7
```

(c) $p = 233 > 15^2$ but $p = 233 < 2 \times 15^2$, the inverse is correct



```
In[27]:= prime = 401
Out[27]= 401

In[28]:= test = ffmap[prime, 15/7]
Out[28]= 174

In[29]:= invmap[prime, test]
         i= 2
         possible candidate= 174
         i= 3
                               53
         possible candidate= - ──
                                2
         i= 4
                            15
         chosen candidate=  ──
                             7
         15
Out[29]= ──
          7
```

(d) $p = 401 > 15^2$ but $p = 401 < 2 \times 15^2$, the inverse is correct

Figure 3.19: Example of map from $\mathbb{Z}_p$ to $\mathbb{Q}$ for different fields. Notice that the relaxed condition gives the correct result even though a priori it could not be predicted to be so.

3.4 Reconstruction over finite fields

In this section we will be describing the few modifications one needs to apply to the reconstruction algorithm in order to perform all the internal operations over a finite field $\mathbb{Z}_p$.

First we consider a single $\mathbb{Z}_p$. Then we will combine the information provided by the reconstruction over m different fields $\{\mathbb{Z}_{p_1}, \cdots, \mathbb{Z}_{p_m}\}$ to get the analytic expression of the input function on $\mathbb{Q}$, using the Chinese remainder theorem. In principle one could perform rational reconstruction over finite fields without introducing this auxiliary tool, since it is completely unrelated to the reconstruction process itself. This is true if one needs not to worry about the size of the primes defining the fields $\mathbb{Z}_p$. However since we are bound to machine size integers we *need* the Chinese remainder theorem in order to access larger fields, so that the map $\mathbb{Z}_P \to \mathbb{Q}$ becomes uniquely defined for all coefficients present in the analytic expression of the reconstructed function. This topic will be treated in the next section.

Reconstruction algorithm on a single $\mathbb{Z}_p$

When going to finite fields the definitions of `h`, `l` and `g` in section 3.2 must be replaced with the following:

```
h[0] = 1;
h[i_] := h[i] = ffsum[p, h[i - 1], 1];
```

```
l[z_, n_, 0] := a[n];
l[z_, n_, m_] := ffsum[p, a[n - m], ffmap[p, ffsum[p, z, -h[n - m]] / l[z, n, m - 1]]];
```

```
g[x_, 0] := Fin[x];
g[x_, n_] := ffmap[p, ffsum[p, x, -h[n - 1]] / ffsum[p, g[x, n - 1], -a[n - 1]]];
```

These are obtained by the previous ones by replacing standard addition and multiplication with $+_p$ and $\times_p$. Divisions $\frac{a}{b}$ are considered as the product $a \times_p b^{-1}$, which is performed internally by `ffmap`. The prime p needs to be specified before running these functions.

The new routine performing the reconstruction over finite fields will be called `Recoverff` and will take an additional argument with respect to `rationalrec`, being the order p of the field over which the reconstruction is to be performed. The black-box algorithm called `Fin`, which is given as an input to `Recoverff` is different than that given to `rationalrec`. On $\mathbb{Q}$ we had:

$$x \in \mathbb{Q} \quad \xrightarrow{input for} \quad \texttt{Fin[x\_]} \quad \xrightarrow{returns} \quad \texttt{out} \in \mathbb{Q} \tag{3.38}$$

and

$$\begin{gathered}\texttt{Fin[x\_]} \quad \xrightarrow{input for} \quad \texttt{rationalrec[Fin[x\_]]} \quad \xrightarrow{returns} \\ \to \text{analytic expression of } \texttt{Fin} \text{ on } \mathbb{Q}\end{gathered} \tag{3.39}$$

Whereas on $\mathbb{Z}_p$ we have

$$(p = \text{prime}, x \in \mathbb{Z}_p) \xrightarrow{input for} \quad \texttt{Fin[p\_,x\_]} \quad \xrightarrow{returns} \quad \texttt{out} \in \mathbb{Z}_p \tag{3.40}$$

we want `Fin` to be still function of `x` only and thus fix p beforehand:

$$\begin{gathered}\text{fix } p\text{=prime} \to \texttt{Fin[p,x\_]=Fin[x\_]} \quad \xrightarrow{input for} \quad \texttt{Recoverff[p,Fin[x\_]]} \\ \xrightarrow{returns} \text{coefficients of analytic expression of } \texttt{Fin[p,x\_]} \text{ on } \mathbb{Z}_p\end{gathered} \tag{3.41}$$

The output returned is the list of coefficients a_i computed and sampling points used y_i, $\{\{a_1, \cdots, a_N\}, \{y_1, \cdots, y_N\}\}$. This is equivalent to giving the Thiele form of the function, but it is more suitable for our purposes. After reconstructing on $\mathbb{Z}_p$ we need to map the result to $\mathbb{Q}$, and this is done mapping the single coefficients one by one. Once we are on $\mathbb{Q}$ the analytic expression of the function can be constructed through `thieleform`.

Vanishing denominators

As seen in section 3.2 fractions with vanishing denominators may appear throughout the calculation. On $\mathbb{Q}$ these give rise to an infinity. On $\mathbb{Z}_p$ infinity does not even exist, instead vanishing denominators lead to a problem related to the definition of division itself, which is inverse multiplication.

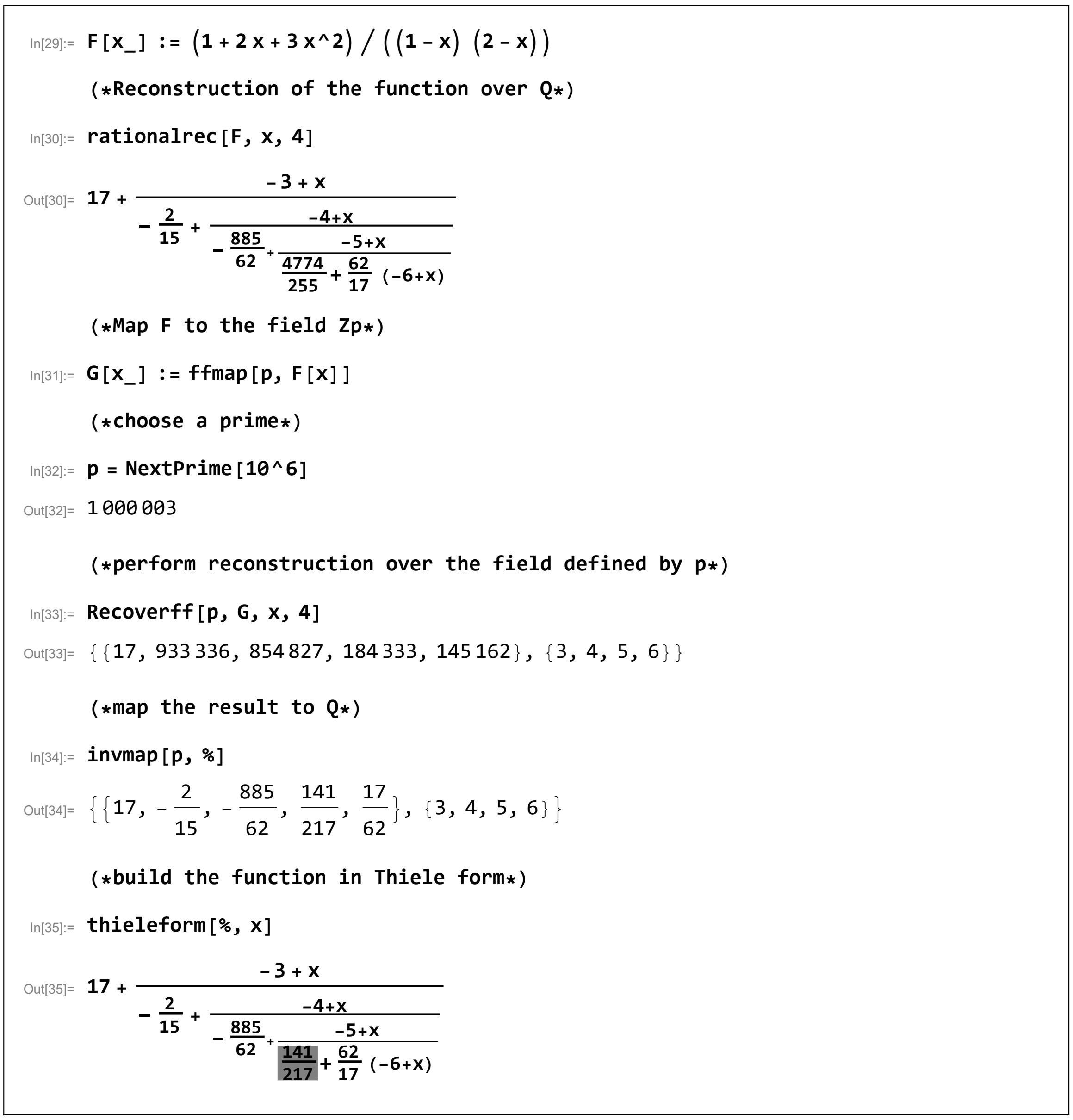
$$\left[\frac{1}{0}\right]_p = 1 \times_p 0^{-1} \tag{3.42}$$

By definition of field the identity element of addition, i.e. zero, does not admit a multiplicative inverse.[6] Thus the operation in eq. (3.42) cannot be performed. The reconstruction algorithm on $\mathbb{Z}_p$ checks each division for vanishing denominators and each time one is found the associated sampling point is discarded.

Example

An example of reconstruction on a single $\mathbb{Z}_p$ is reported in figure 3.20. The prime chosen for the order p of the finite field is the first prime greater than 10^6. Compare the output of the reconstruction over $\mathbb{Z}_p$ mapped back to $\mathbb{Q}$ with that computed directly over $\mathbb{Q}$. It can be seen that all coefficients are correct a part from the highlighted one. This means that the map $\mathbb{Z}_p \to \mathbb{Q}$ associated the wrong rational to the coefficient $184333 \in \mathbb{Z}_p$, whose correct image would have been $\frac{4774}{255}$. The reason is that $4774^2 = 20016676$ which exceeds $p = 1000003$, and thus not even the relaxed condition eq. (3.37) is satisfied, so guessing the correct inverse is highly unlikely. All other coefficients have been obtained correctly since none of their numerators or denominators squared exceeds p. Notice that another coefficient contains 885, and $885^2 = 783225 < p$ but $2 \times 885^2 = 1566450 > p$. The relaxed condition is satisfied but the uniqueness condition eq. (3.36) is not, none the less the coefficient was computed correctly, which is again in favour of considering the relaxed condition as sufficient. Consider a new $\mathbb{Z}_p$ defined by the prime $p = 1000000000039$, which is the smallest prime greater than 10^{12}. The function can be seen to be reconstructed correctly on this field, or more properly the map $\mathbb{Z}_p \to \mathbb{Q}$ yields the correct image for all coefficients, see figure 3.21.

[6]Notice that also on $\mathbb{Q}$ exact zeros would lead to an inversion problem rather than an infinity.



```
In[29]:= F[x_] := (1 + 2 x + 3 x^2) / ((1 - x) (2 - x))

         (*Reconstruction of the function over Q*)

In[30]:= rationalrec[F, x, 4]
```

$$\text{Out[30]= } 17 + \cfrac{-3 + x}{-\frac{2}{15} + \cfrac{-4+x}{-\frac{885}{62} + \cfrac{-5+x}{\frac{4774}{255} + \frac{62}{17}(-6+x)}}}$$

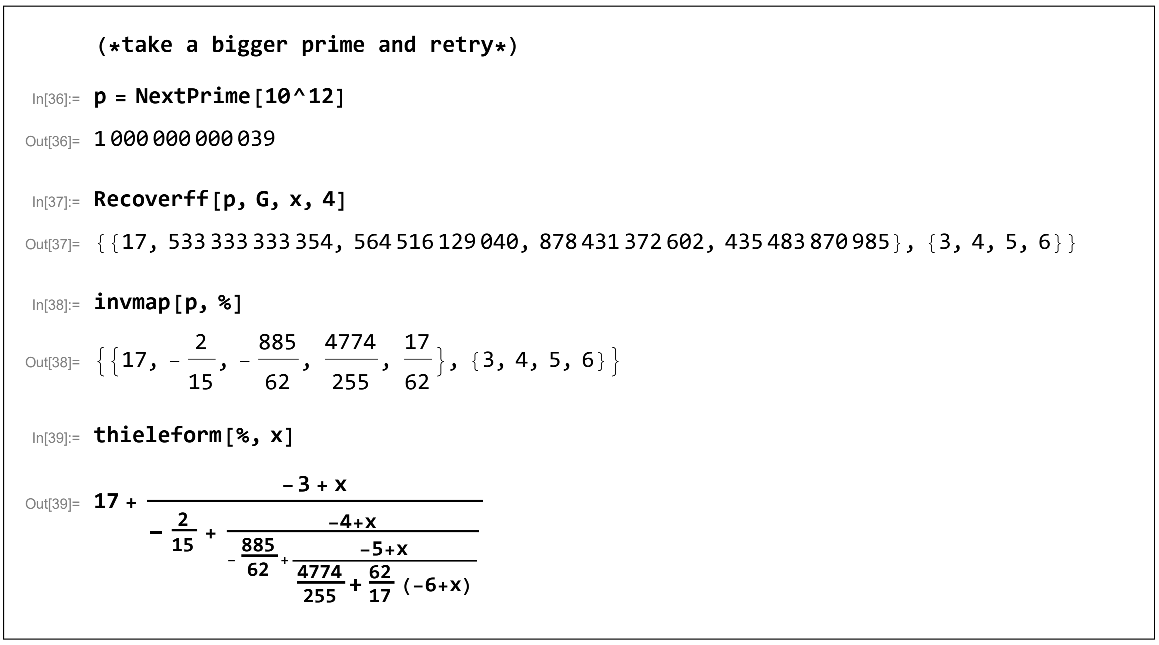
```
         (*Map F to the field Zp*)

In[31]:= G[x_] := ffmap[p, F[x]]

         (*choose a prime*)

In[32]:= p = NextPrime[10^6]

Out[32]= 1 000 003

         (*perform reconstruction over the field defined by p*)

In[33]:= Recoverff[p, G, x, 4]

Out[33]= {{17, 933 336, 854 827, 184 333, 145 162}, {3, 4, 5, 6}}

         (*map the result to Q*)

In[34]:= invmap[p, %]
```

$$\text{Out[34]= } \left\{\left\{17, -\frac{2}{15}, -\frac{885}{62}, \frac{141}{217}, \frac{17}{62}\right\}, \{3, 4, 5, 6\}\right\}$$

```
         (*build the function in Thiele form*)

In[35]:= thieleform[%, x]
```

$$\text{Out[35]= } 17 + \cfrac{-3 + x}{-\frac{2}{15} + \cfrac{-4+x}{-\frac{885}{62} + \cfrac{-5+x}{\frac{141}{217} + \frac{62}{17}(-6+x)}}}$$

Figure 3.20: Example of rational reconstruction of a simple function over $\mathbb{Z}_p$. The prime p chosen is too small and the reconstructed function is wrong.



```
        (*take a bigger prime and retry*)

In[36]:= p = NextPrime[10^12]

Out[36]= 1 000 000 000 039

In[37]:= Recoverff[p, G, x, 4]

Out[37]= {{17, 533 333 333 354, 564 516 129 040, 878 431 372 602, 435 483 870 985}, {3, 4, 5, 6}}

In[38]:= invmap[p, %]
```

Out[38]= $\left\{\left\{17, -\frac{2}{15}, -\frac{885}{62}, \frac{4774}{255}, \frac{17}{62}\right\}, \{3, 4, 5, 6\}\right\}$

```
In[39]:= thieleform[%, x]
```

Out[39]= $$17+\cfrac{-3+x}{-\frac{2}{15}+\cfrac{-4+x}{-\frac{885}{62}+\cfrac{-5+x}{\frac{4774}{255}+\frac{62}{17}(-6+x)}}}$$

Figure 3.21: Example of rational reconstruction of a simple function over $\mathbb{Z}_p$. The prime p chosen is big enough and the reconstructed function is correct.

Termination condition for the reconstruction

In the example of figure 3.20 in order to check whether the reconstruction was performed correctly, we compared the coefficients obtained over $\mathbb{Z}_p$ and mapped to $\mathbb{Q}$ with the coefficients of the Thiele form of the function which was already known. Clearly when performing a reconstruction we do not know the correct coefficients in advance. What we do then is evaluate the reconstructed function and the black-box algorithm over $\mathbb{Z}_{p'}$, where $p' \neq p$, and compare them. If the evaluations agree on all the values of the variable tested, the reconstructed function is considered to be the correct one. More in detail, assign the following names to the functions:

- F^p_{BB} is the black-box algorithm when evaluated over $\mathbb{Z}_p$
- F^p_{test} is the function given by the reconstruction over the field $\mathbb{Z}_p$, which is a function from $\mathbb{Z}_p$ to $\mathbb{Z}_p$
- F is the function obtained by mapping the coefficients of F^p_{test} to $\mathbb{Q}$, F goes from $\mathbb{Q}$ to $\mathbb{Q}$.
- F^p is the function F when evaluated over the finite field $\mathbb{Z}_p$

In order to compute F^p_{test} one needs to evaluate F^p_{BB} many times. The reconstruction of F^p_{test} only terminates when its evaluations agree with those of F^p_{BB} on several consecutive sampling points, in other words when its analytical form on $\mathbb{Z}_p$ exactly reproduces the values given by F^p_{BB}, meaning the two functions overlap completely, i.e. they are the same function on $\mathbb{Z}_p$. Thus there would be no point in checking for further agreement of the black-box algorithm and the reconstructed function over that same field since they would tautologically match. One thus maps (the coefficients of) F^p_{test} to $\mathbb{Q}$ obtaining a function F, and then maps the function over a new field defined by the prime p', $F^{p'}$. Now we can compare the values of $F^{p'}_{BB}$ and of $F^{p'}$ for several different sampling points.

These two functions now have no reason to agree any more, unless they coincide on $\mathbb{Q}$ and thus coincide also on any finite field they are mapped to. If one gets only positive matches, take F and map it to a third field defined by $\tilde{p}$ and compare $F^{\tilde{p}}$ and $F_{BB}^{\tilde{p}}$. After getting only positive matches on m different fields, were m is arbitrary,[7] one may terminate the reconstruction. If instead there is no agreement, take the field $\mathbb{Z}_{p'}$ over which at least one evaluation has already been performed in the checking procedure, and start the reconstruction over that field. By using the already computed values of $F_{BB}^{p'}$ to identify the first coefficients one saves some evaluations and thus time.

The routine performing this complete reconstruction needs to run `Recoverff` multiple times, it is shown in algorithm 5 and its flowchart is represented in figure 3.22. The two integers given as an input define respectively the number of evaluation points over which we require two functions to coincide before recognizong them as the same, and the number of fields over which to check the reconstructed function before terminating the algorithm.[8] Recall that `thieleform` transforms the coefficients defining a function in Thiele form, into the function itself. The routine `checkff` works exactly like `check` in algorithm 4, but takes as additional argument the prime p defining the field over which the functions must be evaluated.

The function `GenerateNewPrime` can be any function generating increasing prime numbers. Since the size of the coefficients of a function in Thiele form is typically huge, it is convenient to start already with a large prime for the first reconstruction and generate successive primes which are at least one order of magnitude larger that the preceding one.

[7]Clearly the larger m the less chances there are of accepting the wrong function as the correct one.
[8]This argument is optional and fixed by default to 4.

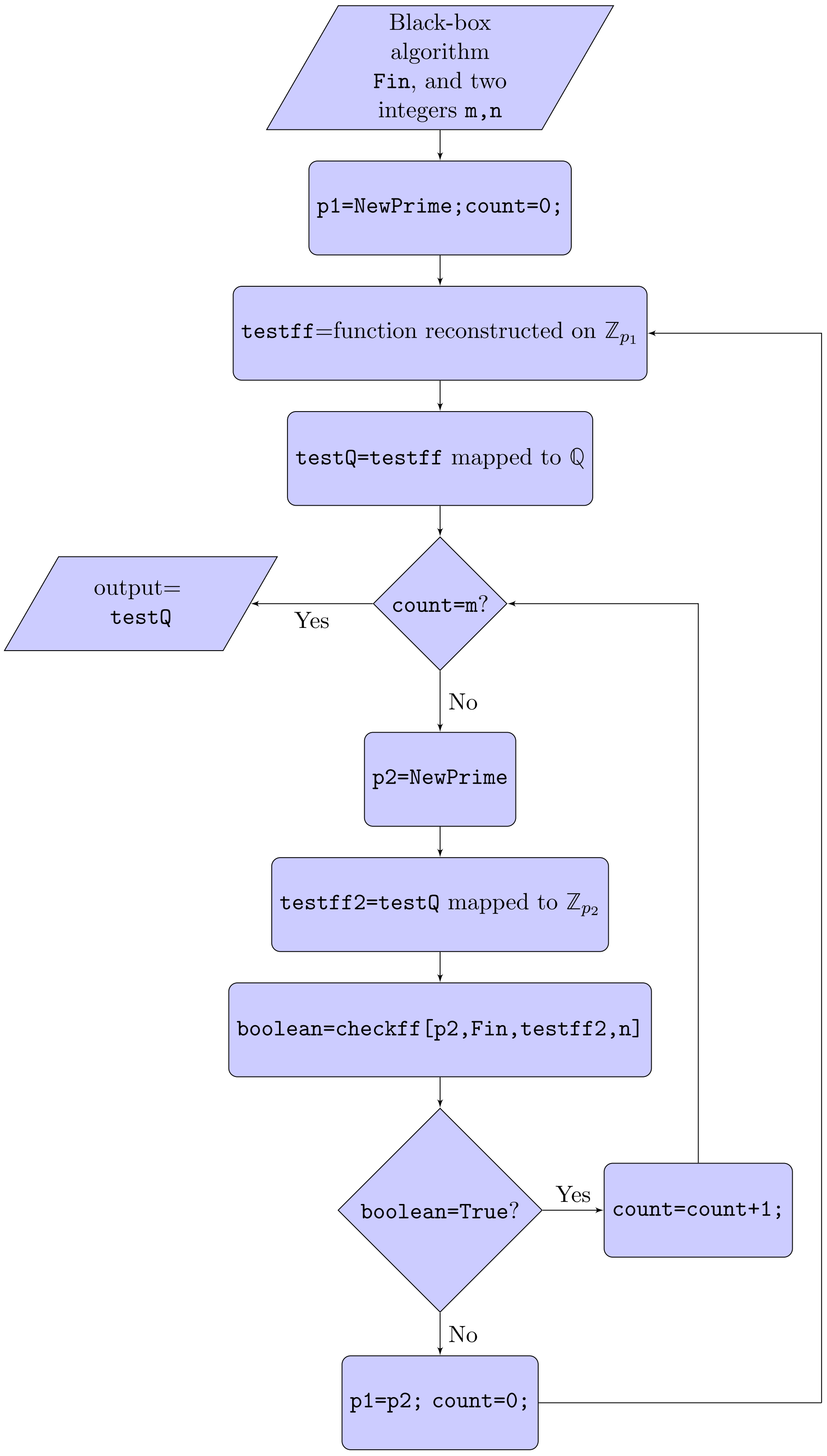

Black-box algorithm Fin, and two integers m,n
p1=NewPrime;count=0;
testff=function reconstructed on $\mathbb{Z}_{p_1}$
testQ=testff mapped to $\mathbb{Q}$
count=m?
Yes
output= testQ
No
p2=NewPrime
testff2=testQ mapped to $\mathbb{Z}_{p_2}$
boolean=checkff[p2,Fin,testff2,n]
boolean=True?
Yes
count=count+1;
No
p1=p2; count=0;

Figure 3.22: Flowchart of the reconstruction procedure on finite fields without using the Chinese remainder theorem.

```
input  : Fin=black-box algorithm, n=integer, m=integer
output: f=analytic form of reconstructed function

count=0;
prime=GenerateNewPrime;
while True do
    testff=Recoverff[prime,Fin,n];
    ;  /* Recoverff returns a list of coefficients defining a function
     */
    test=maptoQ[prime,testff];
    ;                              /* the coefficients are mapped to Q */
    while count < m do
        prime2=GenerateNewPrime;
        testff2=ffmap[prime2,test];
        ;       /* the coefficients are mapped to a new finite field */
        ftest=thieleform[testff2];
        ;        /* the coefficients are converted into the function */
        if checkff[prime2,ftest,Fin,n]=True then
            count=count+1;
        else
            count=0;
            Break;
        end
    end
    prime=prime2;
end
```
Algorithm 5: Rational function reconstruction algorithm over finite fields without Chinese remainder theorem.

3.5 Analytic reconstruction via Chinese remainder theorem

The function used in the example of figure 3.20 was a really simple one, so the coefficients appearing in its Thiele form are not huge numbers, nevertheless one already has to consider primes of order 10^8 to perform a correct reconstruction. One can easily see how important it is to access larger fields when dealing with more involved functions, and for this purpose we need the Chinese remainder theorem.

Once the reconstruction was performed over N different fields $\mathbb{Z}_{p_i}$ with p_i of machine size, we end up with N lists of coefficients and sampling points which identify uniquely a function in Thiele's form over $\mathbb{Z}_{p_i}$. We can then use the Chinese remainder theorem to merge these results into new coefficients over a *ring* whose order is the product of all the p_i.

Merging $\{\mathbb{Z}_{p_1}, \cdots, \mathbb{Z}_{p_n}\}$ into $\mathbb{Z}_{p_1 \cdots p_n}$

The merging procedure is the same adopted for the constructive proof of theorem 2.2.3. Recall that given the following n congruences

$$\begin{cases} X &= x_1 \mod p_1 \\ & \vdots \\ X &= x_n \mod p_n \end{cases} \tag{3.43}$$

and introducing the following notation:

$$P = \prod_i p_i \tag{3.44}$$

$$N_i = \frac{P}{p_i} \tag{3.45}$$

N_i and p_i are pairwise co-prime thus we write the Bezout identities

$$M_i N_i + m_i p_i = 1 \tag{3.46}$$

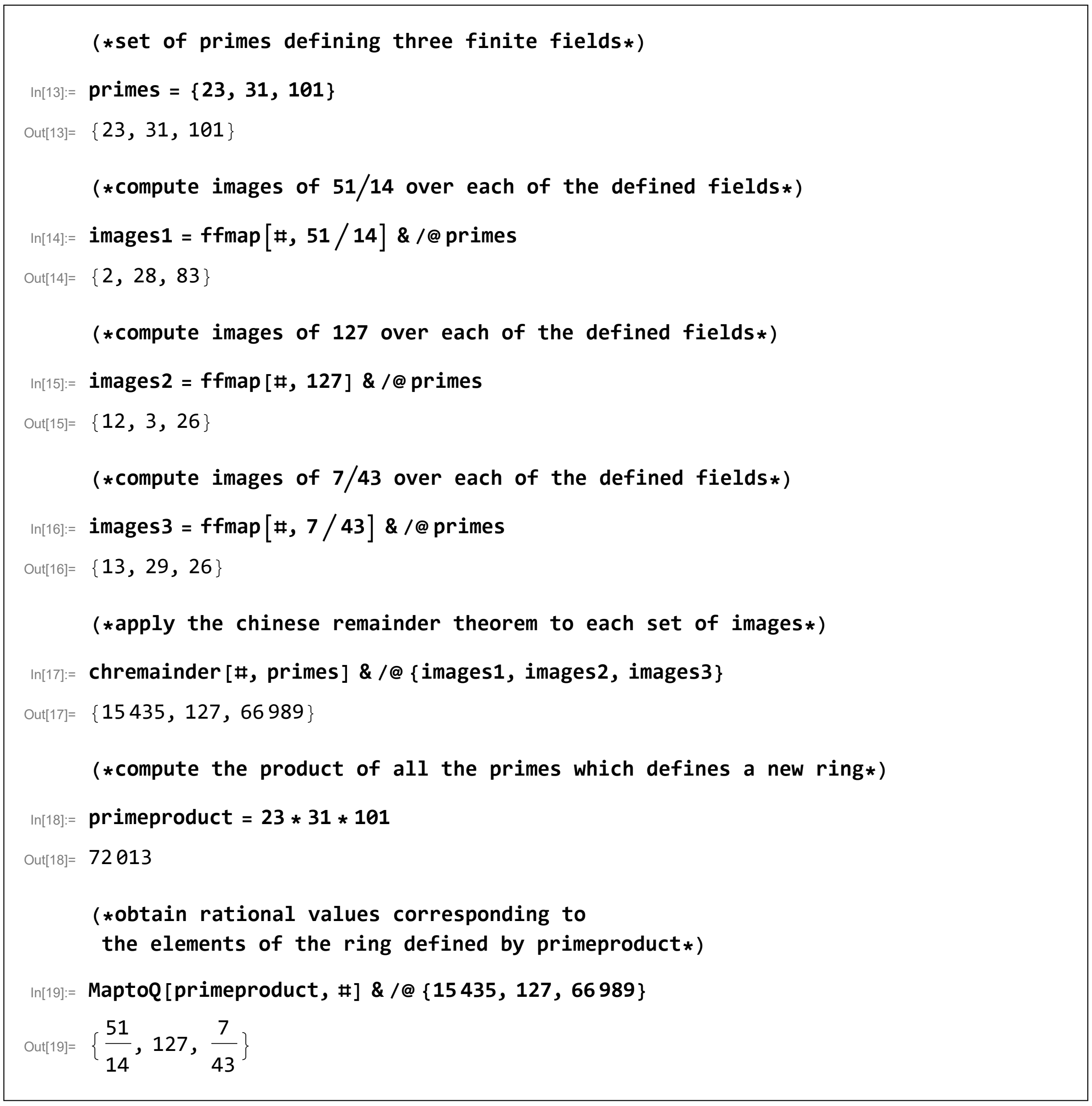
and X defined as in eq. (3.47) solves the system eq. (3.43).

$$X \equiv \sum_i^n x_i M_i N_i \mod P \tag{3.47}$$

In our case the images x_i are lists of coefficients. The routine which computes eq. (3.47) will thus only need to compute all the M_i via extended Euclidean algorithm, given the list of images of X, $\{x_1, \cdots, x_n\}$, and the list of primes $\{p_1, \cdots, p_n\}$. This algorithm will be called `chremainder` and its output will be a list of integers, one corresponding to each set of images $\{x_1, \cdots, x_n\}$ given as an input. An example is displayed in figure 3.23, the notation `f[#]&/@List` applies the function `f` to all the elements of `List`.

A more efficient reconstruction procedure

The Chinese remainder theorem can be used to make the reconstruction algorithm 5 much more efficient. The corresponding flowchart is depicted in figure 3.24. First one performs the reconstruction over a given $\mathbb{Z}_{p_1}$ and checks the reconstructed function over another field $\mathbb{Z}_{p_2}$ as already explained. If this test fails, one reconstructs the function over $\mathbb{Z}_{p_2}$.



```
        (*set of primes defining three finite fields*)

In[13]:= primes = {23, 31, 101}

Out[13]= {23, 31, 101}

        (*compute images of 51/14 over each of the defined fields*)

In[14]:= images1 = ffmap[#, 51 / 14] & /@ primes

Out[14]= {2, 28, 83}

        (*compute images of 127 over each of the defined fields*)

In[15]:= images2 = ffmap[#, 127] & /@ primes

Out[15]= {12, 3, 26}

        (*compute images of 7/43 over each of the defined fields*)

In[16]:= images3 = ffmap[#, 7 / 43] & /@ primes

Out[16]= {13, 29, 26}

        (*apply the chinese remainder theorem to each set of images*)

In[17]:= chremainder[#, primes] & /@ {images1, images2, images3}

Out[17]= {15 435, 127, 66 989}

        (*compute the product of all the primes which defines a new ring*)

In[18]:= primeproduct = 23 * 31 * 101

Out[18]= 72 013

        (*obtain rational values corresponding to
         the elements of the ring defined by primeproduct*)

In[19]:= MaptoQ[primeproduct, #] & /@ {15 435, 127, 66 989}
```

Out[19]= $\left\{\frac{51}{14}, 127, \frac{7}{43}\right\}$

Figure 3.23: Example application of the Chinese remainder theorem. Starting from three rationals we map them to different finite fields and verify that the inverse image obtained using the Chinese remainder theorem is correct.

```
input  : Fin=black-box algorithm, n=integer, m=integer
output: f=analytic form of reconstructed function
count=0;
primeproduct=1;
primelist={};
testfflist={};
prime=GenerateNewPrime;
while True do
    testff=Recoverff[prime,Fin,n] ;            /* Recoverff returns a list of
     coefficients defining a function */
    testfflist=Append[testfflist,testff] ;   /* add the list of coefficients to
     the list of those already computed over the preceding fields */
    chlist=tochineseform[testfflist];
    primelist=Append[primelist,prime];
    testchinese=chremainder[chlist,primelist] ;     /* apply Chinese remainder
     th.  to the coefficients computed over different fields */
    primeproduct=primeproduct*prime;
    test=maptoQ[primeproduct,testchinese] ; /* the coefficients are mapped
     to Q */
    while $count < m$ do
        prime2=GenerateNewPrime;
        testff2=ffmap[prime2,test] ; /* the coefficients are mapped to a new
         finite field */
        ftest=thieleform[testff2] ;    /* the coefficients are converted into
         the function */
        if checkff[prime2,ftest,Fin,n]=True then
            count=count+1;
        else
            count=0;
            Break;
        end
    end
    prime=prime2;
end
```
Algorithm 6: Rational function reconstruction algorithm over finite fields using the Chinese remainder theorem. All the functions used have been defined in the preceding subsections a part from `tochineseform` which is defined by eq. (3.48).

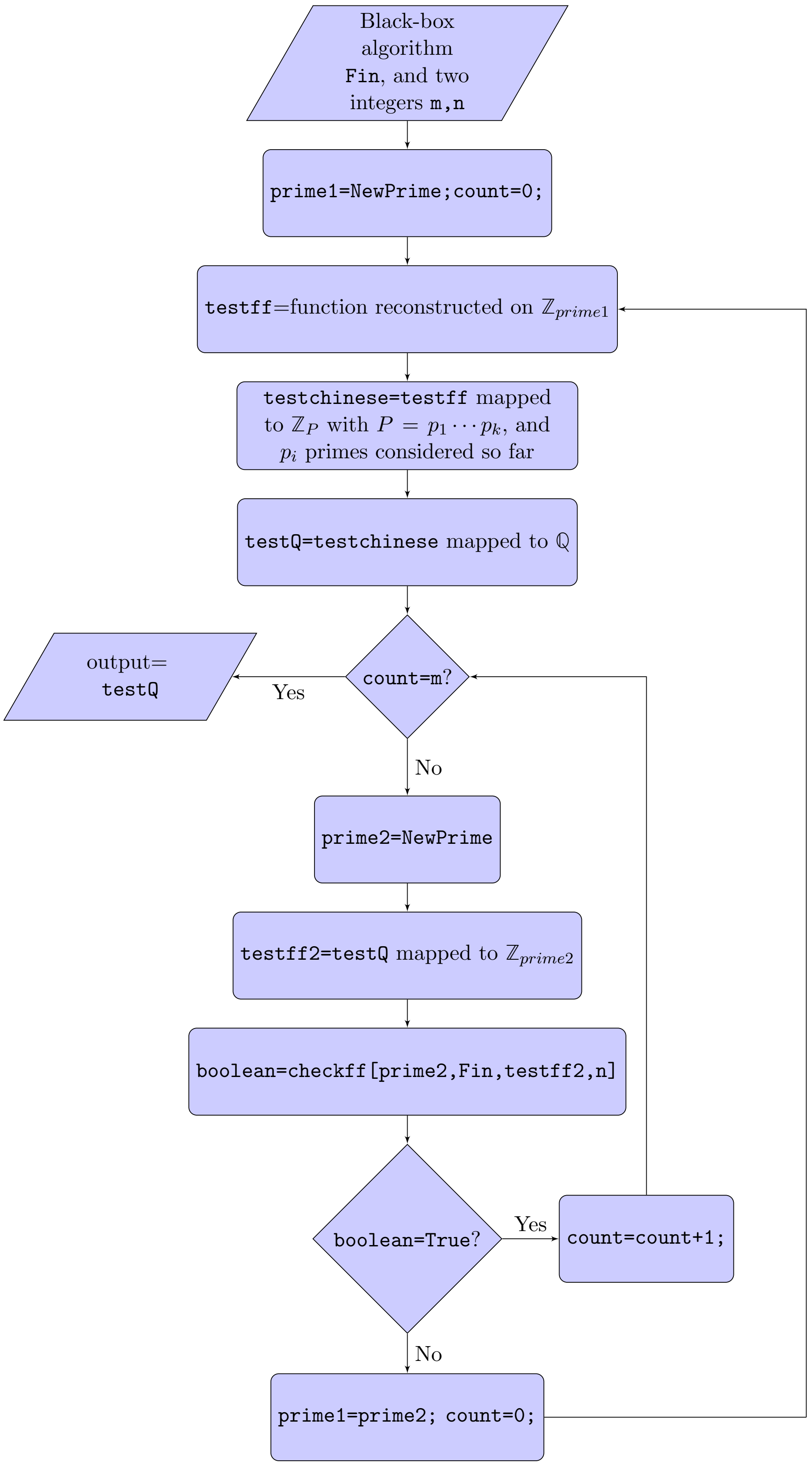

Black-box algorithm Fin, and two integers m,n
prime1=NewPrime;count=0;
testff=function reconstructed on $\mathbb{Z}_{prime1}$
testchinese=testff mapped to $\mathbb{Z}_P$ with $P = p_1 \cdots p_k$, and p_i primes considered so far
testQ=testchinese mapped to $\mathbb{Q}$
count=m?
Yes
output= testQ
No
prime2=NewPrime
testff2=testQ mapped to $\mathbb{Z}_{prime2}$
boolean=checkff[prime2,Fin,testff2,n]
boolean=True?
Yes
count=count+1;
No
prime1=prime2; count=0;

Figure 3.24: Flowchart of the reconstruction procedure on finite fields with Chinese remainder theorem.

Now we map the coefficients, defining the reconstructed function , to $\mathbb{Z}_{p_1p_2}$ and from there to $\mathbb{Q}$ and then to $\mathbb{Z}_{p_3}$ for the check. Since p_1p_2 is clearly much larger than p_2 itself the chances of getting the correct image of all coefficients on $\mathbb{Q}$, and thus getting the correct function, are far greater.

The function `tochineseform` performs the replacement shown in eq. (3.48)

$$\begin{array}{c}\{\{\{a_1^1,\cdots,a_n^1\},\{y_1^1,\cdots,y_n^1\}\},\cdots,\{\{a_1^m,\cdots,a_n^m\},\{y_1^m,\cdots,y_n^m\}\}\}\\ \Big\downarrow \\ \{\{\{a_1^1,\cdots,a_1^m\},\{y_1^1,\cdots,y_1^m\}\},\cdots,\{\{a_n^1,\cdots,a_n^m\},\{y_n^1,\cdots,y_n^m\}\}\}\end{array} \tag{3.48}$$

where a_i^j is the i-th coefficient of the function when reconstructed over the j-th field, and same goes for the sampling points y_i^j.

When performing the reconstruction of a rational function over a finite field, the reconstruction itself will never be wrong.[9] What may lead to an incorrect analytic form of the function over $\mathbb{Q}$ is the mapping from $\mathbb{Z}_P$ to $\mathbb{Q}$. The larger the rational coefficients of the function and the larger the order of the field P must be so that the map $F_\mathbb{Q}$ returns the correct values. Using the Chinese remainder theorem, the value of P is given by the product of all the orders of the fields over which the reconstruction has been performed so far. Thus after each new reconstruction, P grows of the same order of magnitude of the new prime p considered. This means that even with few reconstructions very large P's can be accessed. Of course it is also convenient to choose the single p_i as large as possible. Thus one may consider for the first reconstruction the largest representable machine size prime, and then generate through `GenerateNewPrime` the successively smaller primes in decreasing order. For a 64-bit double precision floating-point system the largest representable prime is

$$\begin{aligned}9007199254740881 &= 2^{53} - 111\\ &= (2^{53}-1) - 110 = M_{53} - 110\end{aligned} \tag{3.49}$$

where $M_n \equiv 2^n - 1$ is called n-th Mersenne number. It has been found that many Mersenne numbers are primes. M_{53} is not, else it would have been the largest representable prime.

An example computation is shown in figure 3.25. The input function `Fin` is the black-box algorithm for the evaluation over $\mathbb{Z}_p$ of the function:

$$\begin{aligned}F(x) &= \frac{\frac{1}{2}+\frac{1}{3}x+\frac{1}{7}x^2}{\frac{7}{13}-\frac{13}{7}x^2}\\ &= \frac{273+182x+78x^2}{294-1014x^2}\end{aligned} \tag{3.50}$$

which over $\mathbb{Z}_p$ becomes

$$F^p(x) = \left[\frac{\left[\frac{1}{2}\right]_p +_p \left[\frac{1}{3}\right]_p \times_p x +_p \left[\frac{1}{7}\right]_p \times_p x \times_p x}{\left[\frac{7}{13}\right]_p +_p \left[\frac{-13}{7}\right]_p \times_p x \times_p x}\right]_p \tag{3.51}$$

[9]Provided the order of the field is larger than the number of sampling points needed for the reconstruction to properly terminate. Otherwise the reconstruction will always inevitably fail, see section 3.2.

where $[x]_p$ stands for `ffmap[p,x]`, and $+_p$ and $\times_p$ for addition and multiplication over $\mathbb{Z}_p$.

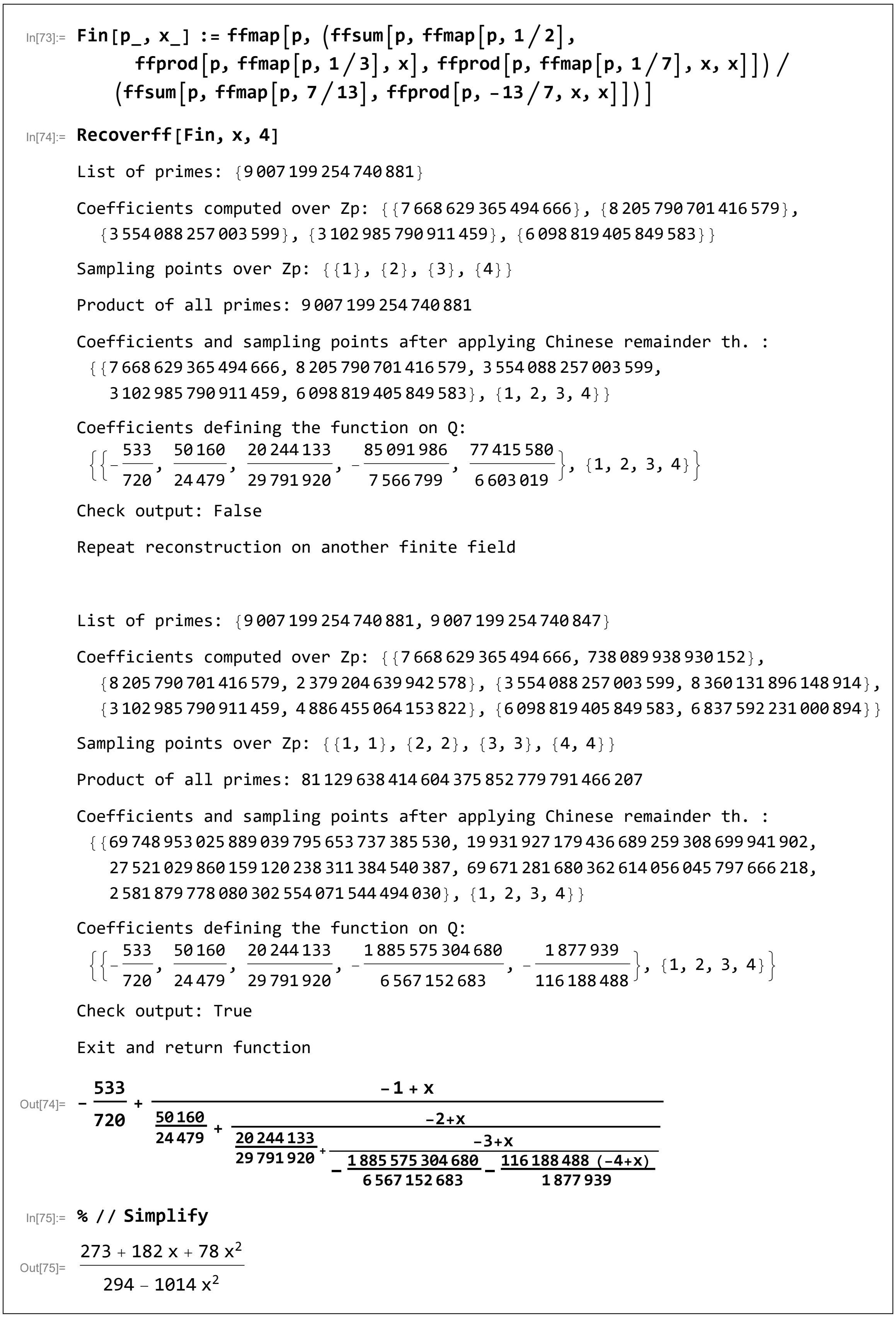
The size of the coefficients of F in Thiele form is such that the correct result is obtained after reconstructing the function over two different finite fields. Notice how fast the size of the ring $\mathbb{Z}_P$ grows using the Chinese remainder theorem. In general starting from the prime eq. (3.49) and generating new primes in decreasing order, we get that for example after reconstructing over five fields $P \sim 10^{75}$.

Arbitrary-precision arithmetic

The whole point of introducing finite fields in the reconstruction procedure was so that we could bypass the use of arbitrary precision arithmetic in the calculations, since this is quite time expensive. Arbitrary-precision arithmetic must be used nonetheless:

- when applying the Chinese remainder theorem to map the results over $\{\mathbb{Z}_{p_1}, \cdots, \mathbb{Z}_{p_m}\}$ to $\mathbb{Z}_P$ with $P = p_1 \cdots p_m$
- when mapping the coefficients from $\mathbb{Z}_P$ to $\mathbb{Q}$

In both these processes integers beyond machine size are involved. Even so, the subroutine of the Chinese remainder theorem is called only few times and the operations it performs are few and simple, so the time invested in this operation is irrelevant compared to that necessary for the reconstructions themselves. The same is true for the map $\mathbb{Z}_P \to \mathbb{Q}$.



```
In[73]:= Fin[p_, x_] := ffmap[p, (ffsum[p, ffmap[p, 1/2],
              ffprod[p, ffmap[p, 1/3], x], ffprod[p, ffmap[p, 1/7], x, x]]) /
           (ffsum[p, ffmap[p, 7/13], ffprod[p, -13/7, x, x]])]

In[74]:= Recoverff[Fin, x, 4]

         List of primes: {9 007 199 254 740 881}

         Coefficients computed over Zp: {{7 668 629 365 494 666}, {8 205 790 701 416 579},
           {3 554 088 257 003 599}, {3 102 985 790 911 459}, {6 098 819 405 849 583}}

         Sampling points over Zp: {{1}, {2}, {3}, {4}}

         Product of all primes: 9 007 199 254 740 881

         Coefficients and sampling points after applying Chinese remainder th. :
          {{7 668 629 365 494 666, 8 205 790 701 416 579, 3 554 088 257 003 599,
            3 102 985 790 911 459, 6 098 819 405 849 583}, {1, 2, 3, 4}}

         Coefficients defining the function on Q:
```

$$\left\{\left\{-\frac{533}{720}, \frac{50160}{24479}, \frac{20244133}{29791920}, -\frac{85091986}{7566799}, \frac{77415580}{6603019}\right\}, \{1, 2, 3, 4\}\right\}$$

```
         Check output: False

         Repeat reconstruction on another finite field


         List of primes: {9 007 199 254 740 881, 9 007 199 254 740 847}

         Coefficients computed over Zp: {{7 668 629 365 494 666, 738 089 938 930 152},
           {8 205 790 701 416 579, 2 379 204 639 942 578}, {3 554 088 257 003 599, 8 360 131 896 148 914},
           {3 102 985 790 911 459, 4 886 455 064 153 822}, {6 098 819 405 849 583, 6 837 592 231 000 894}}

         Sampling points over Zp: {{1, 1}, {2, 2}, {3, 3}, {4, 4}}

         Product of all primes: 81 129 638 414 604 375 852 779 791 466 207

         Coefficients and sampling points after applying Chinese remainder th. :
          {{69 748 953 025 889 039 795 653 737 385 530, 19 931 927 179 436 689 259 308 699 941 902,
            27 521 029 860 159 120 238 311 384 540 387, 69 671 281 680 362 614 056 045 797 666 218,
            2 581 879 778 080 302 554 071 544 494 030}, {1, 2, 3, 4}}

         Coefficients defining the function on Q:
```

$$\left\{\left\{-\frac{533}{720}, \frac{50160}{24479}, \frac{20244133}{29791920}, -\frac{1885575304680}{6567152683}, -\frac{1877939}{116188488}\right\}, \{1, 2, 3, 4\}\right\}$$

```
         Check output: True

         Exit and return function
```

Out[74]=

$$-\frac{533}{720}+\cfrac{-1+x}{\cfrac{50160}{24479}+\cfrac{-2+x}{\cfrac{20244133}{29791920}+\cfrac{-3+x}{-\frac{1885575304680}{6567152683}-\frac{116188488\,(-4+x)}{1877939}}}}$$

```
In[75]:= % // Simplify
```

Out[75]=

$$\frac{273+182\,x+78\,x^2}{294-1014\,x^2}$$

Figure 3.25: Example of rational reconstruction over finite fields using the Chinese remainder theorem.

Chapter 4

Scattering Amplitudes over Finite Fields

In this chapter we are going to discuss first the numerical evaluation of scattering amplitudes via BCFW and Berends-Giele recursion, then their transposition to finite fields and finally the reconstruction techniques applied to them. Each of these steps was already addressed bearing in mind what was to follow. An example of this is the special attention given to the factors of i and to the square roots, whose presence is perfectly fine in a general computation of an amplitude through Berends-Giele or BCFW, but is not allowed when performing rational reconstruction over finite fields. For this reason we introduced appropriate overall factors allowing to restore (if needed) the appropriate powers of i and of eventually present square roots at the end of the computation, while omitting them through out the calculation. Similarly we introduced the use of the light-cone basis instead of the canonical Minkowski basis. Further details will be given when first introducing these tools.

4.1 Numerical evaluation of scattering amplitudes

4.1.1 Spinor helicity formalism

One of the tools we made use of for computations involving spinor-helicity formalism was the Mathematica package S@M [17]. This package allows computations with both two and four-dimensional angle and square brackets, being however better suited for the four-dimensional case. The only draw-back of this package is that some operations are only performed numerically, for example the scalar product. Since in some instances we needed analytical expressions instead, we wrote our own functions performing the simple operations we usually make use of, i.e.: scalar product, contractions among two-component spinors and computation of the spinor components given the 4-momentum.

We shall define the following class of objects:

$$\begin{aligned}\texttt{Components[i]} &= \{\{\texttt{SR[i]},\texttt{SL[i]}\},\{\texttt{LS[i]},\texttt{RS[i]}\}\} \\ &= \{\{|i\rangle,|i]\},\{\langle i|,[i|\}\}\end{aligned} \tag{4.1}$$

From `Components[i]` the momentum can be extracted as[1]

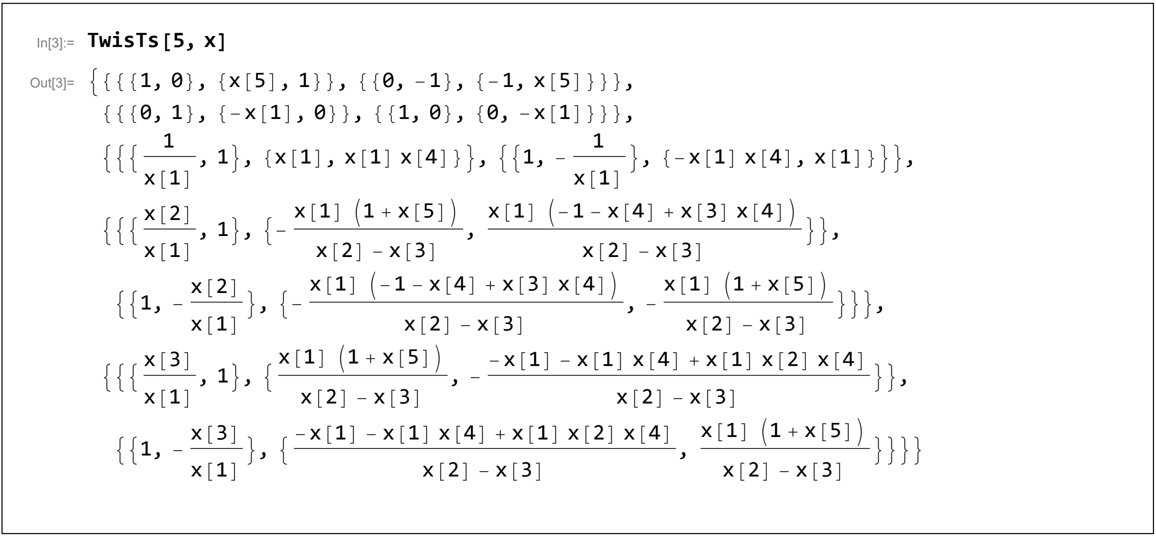
$$\texttt{MomFromSpinor[i]} = \frac{1}{2}\begin{pmatrix}\texttt{LS[i].IdentityMatrix[2].SL[i]}\\ \texttt{LS[i].PauliMatrix[1].SL[i]}\\ \texttt{LS[i].PauliMatrix[2].SL[i]}\\ \texttt{LS[i].PauliMatrix[3].SL[i]}\end{pmatrix} \tag{4.2}$$

which makes use of $p^\mu = \frac{1}{2}\langle p|\sigma^\mu|p]$. We will also be needing the inverse function of eq. (4.2) which is called `SpinorFromMom` and extracts the spinor components given the momentum of a particle. It uses a generalisation of eq. (1.66), which does not present singularities for any value of the momentum components. We shall provide the expression used later on, since further remarks about it will be given when talking about BCFW recurrence.

Our goal is to perform a rational reconstruction of the amplitude given its numerical value on some phase-space points. These points will be computed using either the Berends-Giele or the BCFW recursion as functions of the angle and square spinors written in terms of the momentum-twistor variables. We need thus to generate the momentum-twistor parametrization for any given number of external legs. This is done through the function called `TwisTs`, which takes as input an integer `n` defining the number of legs, and whose output is a list of objects belonging to the class `Components` defined in eq. (4.1). These are expressed in terms of $3n - 10$ variables `x[i]`.[2] The parametrization used by `TwisTs` is the one by Tiziano Peraro.

In order to check the correctness of the spinors generated we first tested whether the associated momenta were massless and momentum conservation applied as it should by construction. As a second less tautological check we generated a set of on- shell conserved momenta for a given number of external legs using S@M [17] and computed all the scalar products among them. Then we generated the spinors through our code as functions of the x_i, computed the associated momenta, computed the numerical values of the x_i through the variable changes of eq. (1.115-1.120) and then computed the scalar products among these momenta comparing them with the first ones.

As an example output we displayed in figure 4.1 a five-point amplitude parametrization.



```
In[3]:= TwisTs[5, x]
```

$$\text{Out[3]=}\ \{\{\{\{1,0\},\{x[5],1\}\},\{\{0,-1\},\{-1,x[5]\}\}\},$$
$$\{\{\{0,1\},\{-x[1],0\}\},\{\{1,0\},\{0,-x[1]\}\}\},$$
$$\{\{\{\frac{1}{x[1]},1\},\{x[1],x[1]\,x[4]\}\},\{\{1,-\frac{1}{x[1]}\},\{-x[1]\,x[4],x[1]\}\}\},$$
$$\{\{\{\frac{x[2]}{x[1]},1\},\{-\frac{x[1]\,(1+x[5])}{x[2]-x[3]},\frac{x[1]\,(-1-x[4]+x[3]\,x[4])}{x[2]-x[3]}\}\},$$
$$\{\{1,-\frac{x[2]}{x[1]}\},\{-\frac{x[1]\,(-1-x[4]+x[3]\,x[4])}{x[2]-x[3]},-\frac{x[1]\,(1+x[5])}{x[2]-x[3]}\}\}\},$$
$$\{\{\{\frac{x[3]}{x[1]},1\},\{\frac{x[1]\,(1+x[5])}{x[2]-x[3]},-\frac{-x[1]-x[1]\,x[4]+x[1]\,x[2]\,x[4]}{x[2]-x[3]}\}\},$$
$$\{\{1,-\frac{x[3]}{x[1]}\},\{\frac{-x[1]-x[1]\,x[4]+x[1]\,x[2]\,x[4]}{x[2]-x[3]},\frac{x[1]\,(1+x[5])}{x[2]-x[3]}\}\}\}\}$$

Figure 4.1: Output example of the function `TwisTs`, using Peraro's choice of variables

[1]The dot expresses the matrix product.

[2]The name of the variables may be chosen through an optional argument to be given to `TwisTs`.

4.1.2 Berends-Giele recursion

Overall multiplicative factor

As already mentioned the reconstruction over finite fields is compromised by i and square root factors. In the Berends-Giele recurrence these factors appear in the gluon vertex and propagator definitions, as well as in the polarization vector normalizations. However it is possible to completely remove these factors from inside the current provided the final amplitude is multiplied by an appropriate pre-factor which takes into account all the removed constants. Thus we omitted these factors through out the computation, in order for the code to be ready for rational reconstruction over finite fields, and restored the final resulting pre-factor at the end of the calculation.[3] More in detail

$$\mathbf{M}_n(1,\cdots,n) = C^{BG}(i,\sqrt{2})\widetilde{\mathbf{M}}_n^{BG}(1,\cdots,n) \tag{4.3}$$

Where the superscript BG stays for Berends-Giele, since later we will provide an analogous relation for the BCFW recurrence. Now we are going to prove that all the irrational and imaginary factors *arising in the calculation of the amplitude* can be reabsorbed in C^{BG} and compute its value.

In $\widetilde{\mathbf{M}}_n^{BG}$ we applied the following substitutions:

$$\begin{cases} P \mapsto c_p P = iP \\ V_3 \mapsto c_3 V_3 = (-i)\sqrt{2}V_3 \\ V_4 \mapsto c_4 V_4 = (-i)2V_4 \\ \epsilon^\mu \mapsto c_\epsilon \epsilon^\mu = \sqrt{2}\epsilon^\mu \end{cases} \tag{4.4}$$

The value of $\widetilde{M}_n^{BG}$ will be given by a sum over many addenda corresponding to different diagrams resulting from the application of the Berends-Giele recursion, these will cover all the possible allowed combinations of three- and four-point vertices and propagators. To recover the correct value of the amplitude with the appropriate $\sqrt{2}$ and i dependence one has to multiply each addendum by a product of $\frac{1}{c_P}$, $\frac{1}{c_3}$,$\frac{1}{c_4}$ and $\frac{1}{c_\epsilon}$ each with an appropriate power, depending on the number of propagators, three-point vertices, four-point vertices and polarization vectors appearing in the associated diagram. Thus a priori all these pre-factors appear different, however we are going to show that they have all the same value. Consider the form of the Berends-Giele current eq. (1.122), after having performed the complete recurrence there will be three different types of terms: some involving only three-point vertixes, some only four-point vertexes and for the most part terms with both types of vertexes. Of course all of these are contracted with the same number n of polarization vectors. First notice that the terms involving only three-point vertexes will all have the same number of vertexes as well as propagators, in particular if we have n external legs there will be $n-2$ three-point vertexes and $n-3$ propagators. Thus for these terms C must be of the form

$$\begin{aligned} C^{BG}(i,\sqrt{2}) &= \Big(\frac{1}{c_3}\Big)^{n-2}\Big(\frac{1}{c_P}\Big)^{n-3}\Big(\frac{1}{c_\epsilon}\Big)^{n} \\ &= i\Big(\frac{1}{\sqrt{2}}\Big)^{2n-2} \end{aligned} \tag{4.5}$$

Now consider the mixed terms. The crucial fact is that each time a four vertex appears in a given term, there will be also the corresponding diagram with the four-point vertex

[3]By the end of the calculation we mean either once the amplitude was computed through Berends-Giele, if we simply wanted to know the numerical value of that given amplitude, or after rational reconstruction was performed if we wanted the analytical form of the amplitude.

substituted with two three-point vertexes connected by a propagator. One can easily check that

$$c_4 = c_3^2 c_P \tag{4.6}$$

In other words the mixed terms can be obtained from the pure three-point terms by contracting two vertexes and a propagator into a four point vertex and taking all the possible combinations of this operation. The same is true the other way around, expanding the four-point vertexes in the mixed terms or pure four-point terms into two three-point vertexes and a propagator we can get all the pure three-point terms. We then see that due to eq. (4.6) the mixed and pure four-point terms will have the same overall factor eq. (4.5).

Clearly there are still imaginary (and possibly irrational) factors appearing in the amplitude, namely those descending from the polarization vectors whose components are in general complex. To take care of these i's we introduce the light-cone basis.

Light-cone basis

The advantage of using light-cone basis is that starting from real spinor components,[4] we get real momenta and polarization vectors as well. The light cone basis is defined by the variable change[5]

$$p = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} \quad \mapsto \quad p_l = \begin{pmatrix} p_0 + p_3 \\ p_0 - p_3 \\ p_1 + ip_2 \\ p_1 - ip_2 \end{pmatrix} \tag{4.7}$$

Now since we are computing p out of the expression

$$p^\mu = \frac{\langle p|\sigma^\mu|p]}{2}$$

if $\langle p|$ and $|p]$ are real, the component p^2 will be imaginary due to the product of a real vector and co-vector with the Pauli matrix σ^2 which has imaginary components. Thus by taking the momentum in light-cone basis as defined above, all its components will be real since imaginary factors cancel out.

The recurrence

There are two possible approaches for a code implementation of this recurrence relation. Either one proceeds top-down, i.e. writing the n-point current in terms of lower-point ones until only one-gluon currents are left. Or bottom-up, constructing first all the numerical two-point currents, then three-point, four-point and so on up to n. The analytic form of the current J^μ leads to an immediate implementation of the first type, however this presents a major disadvantage: we have to deal with analytical expressions of the currents, which need to be stored by Mathematica, until the end of the computation is reached and numerical values for the $J(i)$ single-gluon currents are substituted back inside each of them. The resulting code is highly inefficient in terms of time, thus we adopted the bottom-up approach.

We defined both, the Berends-Giele current J^μ and the auxiliary current J_a^μ, in terms of three arguments: the label of the first and last gluon appearing in the current and the

[4]Using momentum-twistor parametrization the spinor components will all be rational.

[5]Usually there is also a $\sqrt{2}$ factor for normalization, however we do not include it because as already explained we do not want square roots to appear.

label assigned to the momenta. In other words `J[i,j,P]` $\equiv J(i,\cdots,j)$ and `Ja[i,j,P]` $\equiv J_a(i,\cdots,j)$ with momenta called $P(i)$. An example is shown in figure 4.2.

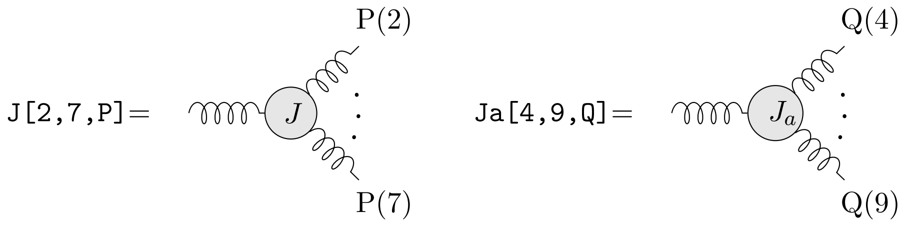


Figure 4.2: Example of Berends-Giele and auxiliary current definition

From eq. (1.122) and eq. (1.136) we see that `J[i,j,P]` and `Ja[i,j,P]` are defined in terms of `J[a,b,P]`, with $b-a<j-i$ and $i\le a,b\le j$, and of the three and four-gluon vertexes. These are defined as follows:

$$\texttt{V3}[P_{\{1,i\}},P_{\{i+1,n\}},J^\nu,J^\rho]=V_3^{\mu\nu\rho}J^\nu(1,\cdots,i)J^\rho(i+1,\cdots,n)|_{\mu=\{0,1,2,3\}} \tag{4.8}$$

$$\texttt{V4}[J^\nu,J^\rho,J^\sigma]=V_4^{\mu\nu\rho\sigma}J^\nu(1,\cdots,i)J^\rho(i+1,\cdots,j)J^\sigma(j+1,\cdots,n)|_{\mu=\{0,1,2,3\}} \tag{4.9}$$

Both `V3` and `V4` are lists of four elements, i.e. they represent the entire four-vector.. A graphical representation of these relations is given in figure 4.3 and figure 4.4.

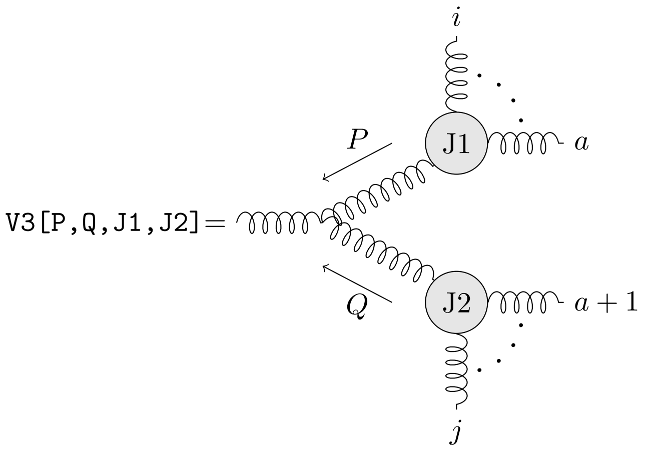


Figure 4.3: Graphical representation of the three-gluon vertex object definition.

We can then rewrite eq. (1.122) as

$$\begin{aligned}\texttt{J[1,n,q]}=\frac{-1}{P_{1,n}^2}\Bigg[&\sum_{i=2}^{n-1}\texttt{V3}[P_{\{1,i\}},P_{\{i+1,n\}},\texttt{J[1,i,q]},\texttt{J[i+1,n,q]}]\\&+\sum_{i=2}^{n-2}\sum_{j=i+1}^{n-1}\texttt{V4[J[1,i,q],J[i+1,j,q],J[j+1,n,q]]}\Bigg]\end{aligned} \tag{4.10}$$

Recall that all these functions use the modified versions of vertexes and propagators, without imaginary and irrational factors, defined in eq. (4.4). So, once all these definitions are given we start from

$$\texttt{J[i,i,P]}\equiv\epsilon(i) \tag{4.11}$$

with $i=\{1,\cdots,n+1\}$. To compute the amplitude we consider the first n gluons and compute the associated Berends-Giele current from bottom-up. To do so we need all the

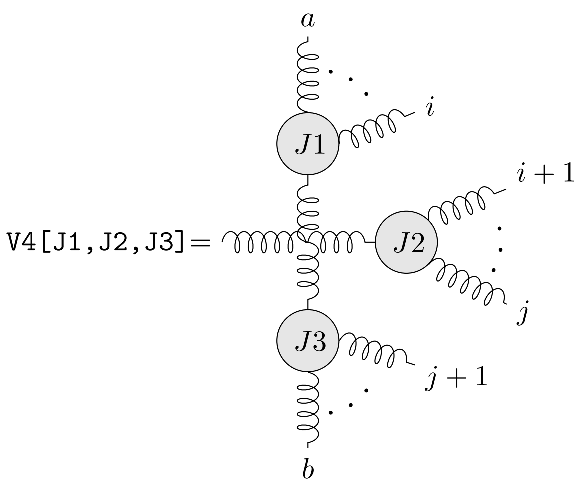


Figure 4.4: Graphical representation of the four-gluon vertex object definition.

possible contractions among the lower point currents, these are given by the `bgcurrent` shown in algorithm 7. Its flowchart is represented in figure 4.5. An example is displayed in figure 4.6. The algorithm takes as input the labels of the first and last gluon entering the colour ordered amplitude, say `x,y`, and performs all the possible contractions of lower point currents in the range from `x` to `y`. Once the one-point currents are defined as the polarization vectors, `bgcurrent` is used to compute the currents in increasing order numerically.

```
input  : x,y Integers
output: out=Berends-Giele current

length=y-x ;
for i=0 to length-1 do
    for k=x to y-i do
        out=J[k,k+i];
    end
end
out=Ja[x,y];
```

Algorithm 7: `bgcurrent` computes the numerical values of the Berends-Giele currents in increasing order.

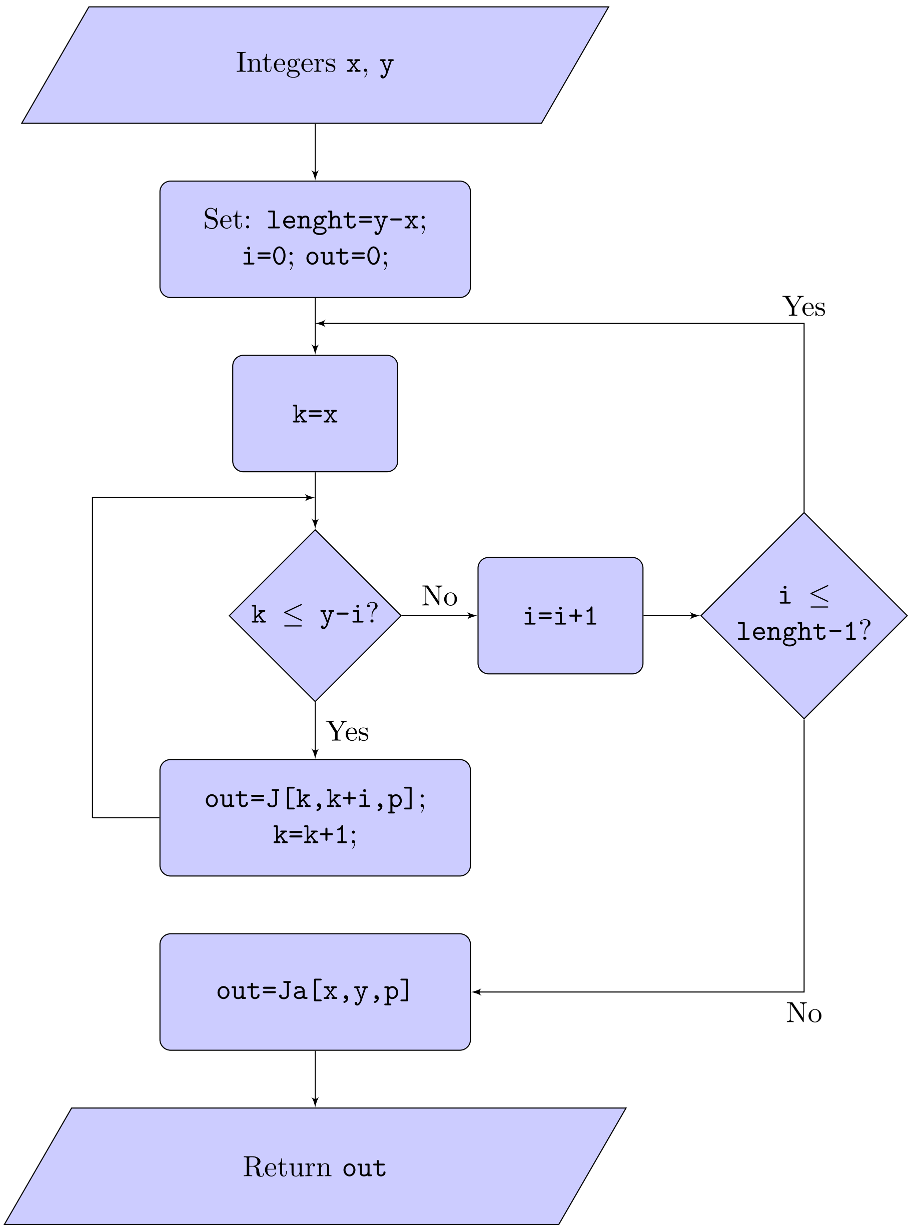
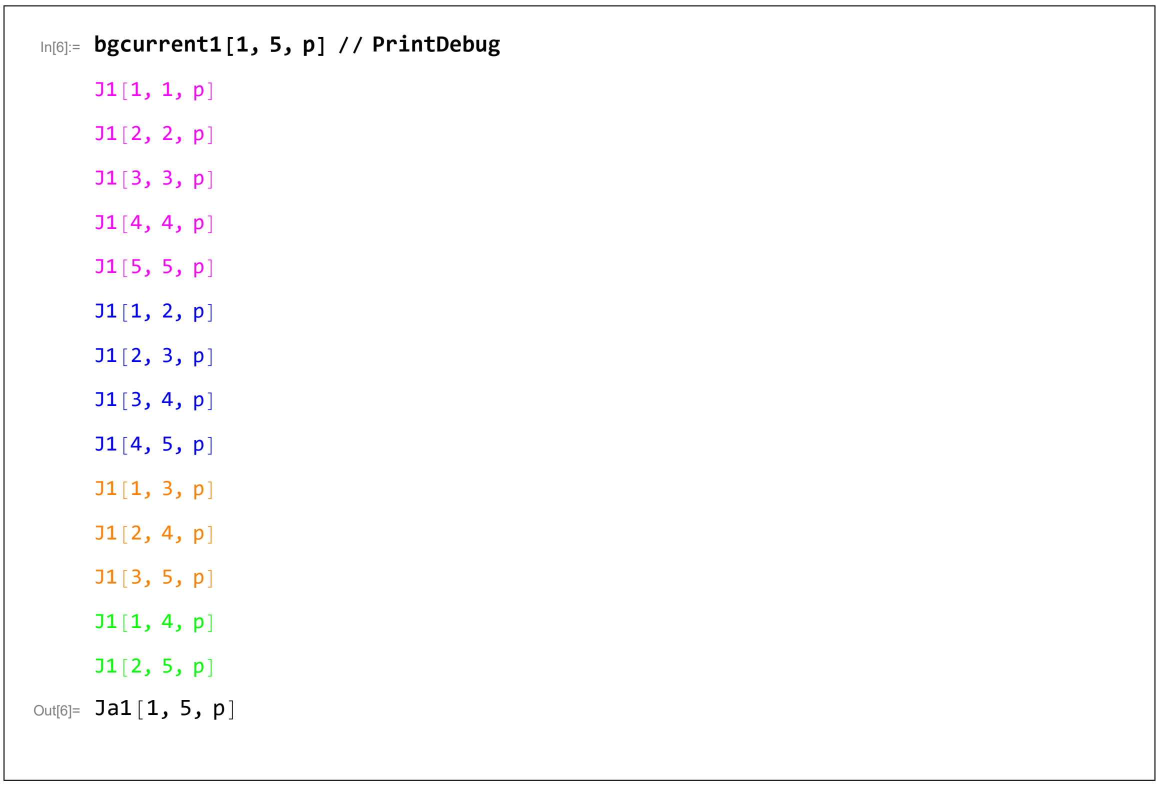


Figure 4.5: Flowchart of the algorithm `bgcurrent` computing the Berends-Giele current



```
In[6]:= bgcurrent1[1, 5, p] // PrintDebug
        J1[1, 1, p]
        J1[2, 2, p]
        J1[3, 3, p]
        J1[4, 4, p]
        J1[5, 5, p]
        J1[1, 2, p]
        J1[2, 3, p]
        J1[3, 4, p]
        J1[4, 5, p]
        J1[1, 3, p]
        J1[2, 4, p]
        J1[3, 5, p]
        J1[1, 4, p]
        J1[2, 5, p]
Out[6]= Ja1[1, 5, p]
```

Figure 4.6: Example of application of `bgcurrent` to a five gluon current. The routine computes the currents `J` in the order printed: one gluon (magenta), two gluon (blue), three gluon (orange), four gluon (green) and finally the five gluon auxiliary current, returned as the output. If numerical values for all the $J[i, i, p]$ (magenta) were entered before applying `bgcurrent`, all the symbolical values printed would be numbers instead.

If `bgcurrent` is run after having assigned to each single gluon current the numerical value of the corresponding polarization vector, the output of the routine will be a number. The amplitude can then be computed contracting `bgcurrent[1,n,p]`, which is the numerical value of the Berends-Giele current for the first `n` gluons, with the polarization vector $\epsilon(n+1)$. Recall that we are using light-cone basis thus the scalar product must use the appropriate metric. In our code this scalar product is called `mplc`.

```
input  : {Components[1], ··· ,Components[n+1]}, list of positive helicity
         gluons and negative helicity gluons
output: out

out=0;
for i=1 to n+1 do
    J[i,i,p]=$\epsilon(i)$ ;                /* The $\epsilon$ are computed via eq. (1.44) */
end
out=bgcurrent[1,n,p];
out=mplc[out,$\epsilon(n+1)$];
```

Algorithm 8: Berends-Giele recursion `BGAt2`

Some examples

As an example we report in figure 4.7 the six-point amplitude computed starting from the spinor components generated using Peraro's choice of variables, with numerical values $x_i = i$. We considered the following helicity configurations:

- $(1^+, 2^+, 3^+, 4^+, 5^+, 6^+)$
- $(1^+, 2^+, 3^+, 4^+, 5^-, 6^+)$
- $(1^+, 2^-, 3^+, 4^-, 5^+, 6^+)$
- $(1^+, 2^-, 3^+, 4^-, 5^+, 6^-)$
- $(1^-, 2^+, 3^-, 4^-, 5^+, 6^+)$

To get the correct value of the amplitude $\mathbf{M}_6 = C_6(i, \sqrt{2})\widetilde{\mathbf{M}}_6^{BG}$ one must multiply the displayed values by

$$C_6^{BG}(i, \sqrt{2}) = \frac{i}{32} \tag{4.12}$$

Checking the results

In order to check the correctness of our code, we compared the results obtained with those given by the MHV and anti-MHV amplitudes, whose values can be computed easily once the spinor components are known. For a further check we used the analytical values of next to MHV six-point amplitudes reported in the appendix of [8].

4.1.3 BCFW recursion

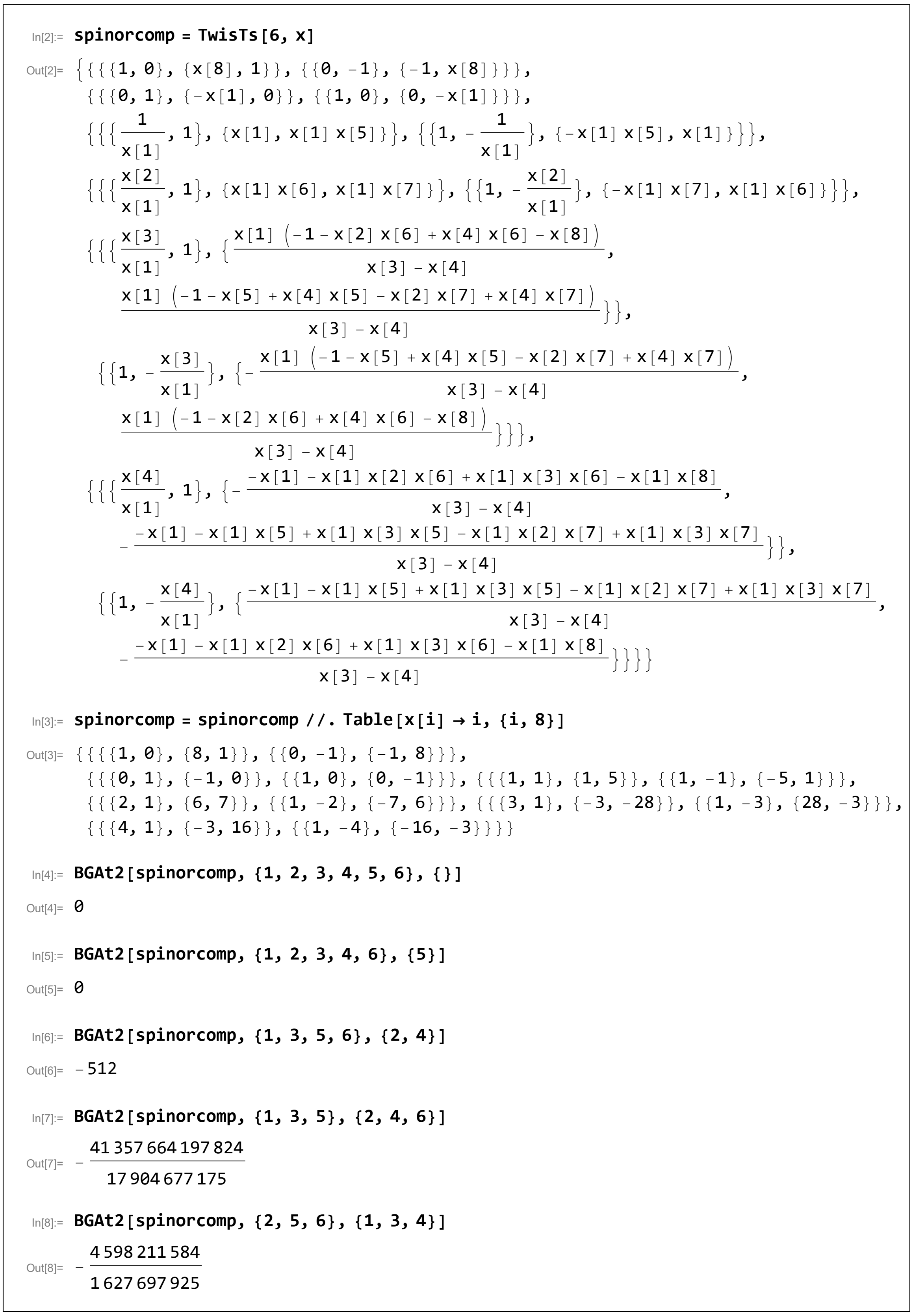
For the BCFW recursion the bottom up approach, i.e. computing numerically lower-point functions working up to the desired number of legs, is not viable. The reason for this is that differently from Berends-Giele where external momenta and spinor components stay fixed, and all we have to do is combine them in an appropriate way, in BCFW after each step of the recursion two external momenta get to be redefined and become new complex momenta and two more external particles are added, namely the internal particle going on-shell with the two possible helicities. Moreover the way this happens depends on the "history" of the computation, depending on the order with which the propagators were put on-shell. So we are forced to perform a semi-analytical computation, which in Berends-Giele could be avoided.

Overall factor and little group scaling

As in the case of Berends-Giele we want to eliminate all the i and square root dependence. Looking at eq. (1.166) we can see that the i dependence is easily factored out: at each step of the iteration a pair of new i's appear, the number of steps needed for a complete factorization is given by the number of propagators which need to be put on-shell. With n external lines there are $n-3$ propagators in an amplitude comprised of only three-point vertices, thus we can omit all the internal i factors if we compensate with an $i^{2(n-3)}$ at the end of the computation. So we set

$$\mathbf{M}_n(1, \cdots, n) = C^{BCFW}(i)\widetilde{\mathbf{M}}_n^{BCFW}(1, \cdots, n) \tag{4.13}$$

$$C^{BCFW}(i) = i^{2n-6} \tag{4.14}$$



```
In[2]:= spinorcomp = TwisTs[6, x]

Out[2]= {{{{1, 0}, {x[8], 1}}, {{0, -1}, {-1, x[8]}}},
  {{{0, 1}, {-x[1], 0}}, {{1, 0}, {0, -x[1]}}},
  {{{1/x[1], 1}, {x[1], x[1] x[5]}}, {{1, -1/x[1]}, {-x[1] x[5], x[1]}}},
  {{{x[2]/x[1], 1}, {x[1] x[6], x[1] x[7]}}, {{1, -x[2]/x[1]}, {-x[1] x[7], x[1] x[6]}}},
  {{{x[3]/x[1], 1}, {(x[1] (-1 - x[2] x[6] + x[4] x[6] - x[8]))/(x[3] - x[4]),
     (x[1] (-1 - x[5] + x[4] x[5] - x[2] x[7] + x[4] x[7]))/(x[3] - x[4])}},
   {{1, -x[3]/x[1]}, {-(x[1] (-1 - x[5] + x[4] x[5] - x[2] x[7] + x[4] x[7]))/(x[3] - x[4]),
     (x[1] (-1 - x[2] x[6] + x[4] x[6] - x[8]))/(x[3] - x[4])}}},
  {{{x[4]/x[1], 1}, {-(-x[1] - x[1] x[2] x[6] + x[1] x[3] x[6] - x[1] x[8])/(x[3] - x[4]),
     -(-x[1] - x[1] x[5] + x[1] x[3] x[5] - x[1] x[2] x[7] + x[1] x[3] x[7])/(x[3] - x[4])}},
   {{1, -x[4]/x[1]}, {(-x[1] - x[1] x[5] + x[1] x[3] x[5] - x[1] x[2] x[7] + x[1] x[3] x[7])/(x[3] - x[4]),
     -(-x[1] - x[1] x[2] x[6] + x[1] x[3] x[6] - x[1] x[8])/(x[3] - x[4])}}}}

In[3]:= spinorcomp = spinorcomp //. Table[x[i] → i, {i, 8}]

Out[3]= {{{{1, 0}, {8, 1}}, {{0, -1}, {-1, 8}}},
  {{{0, 1}, {-1, 0}}, {{1, 0}, {0, -1}}}, {{{1, 1}, {1, 5}}, {{1, -1}, {-5, 1}}},
  {{{2, 1}, {6, 7}}, {{1, -2}, {-7, 6}}}, {{{3, 1}, {-3, -28}}, {{1, -3}, {28, -3}}},
  {{{4, 1}, {-3, 16}}, {{1, -4}, {-16, -3}}}}

In[4]:= BGAt2[spinorcomp, {1, 2, 3, 4, 5, 6}, {}]

Out[4]= 0

In[5]:= BGAt2[spinorcomp, {1, 2, 3, 4, 6}, {5}]

Out[5]= 0

In[6]:= BGAt2[spinorcomp, {1, 3, 5, 6}, {2, 4}]

Out[6]= -512

In[7]:= BGAt2[spinorcomp, {1, 3, 5}, {2, 4, 6}]

Out[7]= -41 357 664 197 824/17 904 677 175

In[8]:= BGAt2[spinorcomp, {2, 5, 6}, {1, 3, 4}]

Out[8]= -4 598 211 584/1 627 697 925
```

Figure 4.7: Example of phase-free amplitudes $\widetilde{\mathbf{M}}_6^{BG}$ computed with Berends-Giele recursion, starting from numerical momentum-twistor components.

A part from these imaginary factors there are also some square roots to account for, which arise when computing the spinor components starting from the momentum of a particle. In fact the analytical expression of the spinor components in terms of the momentum is given by

$$|p\rangle = \begin{pmatrix} \sqrt{p^+} \\ \dfrac{p^1 + ip^2}{\sqrt{p^+}} \end{pmatrix} \qquad |p] = \begin{pmatrix} \dfrac{p^1 - ip^2}{\sqrt{p^+}} \\ -\sqrt{p^+} \end{pmatrix} \tag{4.15}$$

With $p^\pm = p_0 \pm p_3$. Since the square root does not involve a constant factor but momentum components, there is no way of factoring out this dependence. However the external spinor components are given in terms of momentum-twistor variables and are already rational numbers, and eq. (4.15) needs only to be used to compute the spinor components associated to the internal particles which were put on-shell. Because of the opposite helicities of these particle when entering the left and right sub-amplitude which arise from factorization, we have that each time a $|p\rangle$ appears also a $[p|$ arises, and similar for $|p]$ and $\langle p|$. So we can perform a little group transformation, which does not affect the momentum of the particle by construction, and because of what we just said does not affect the amplitude itself either. So map

$$|p\rangle \mapsto \frac{1}{\sqrt{p^+}}|p\rangle\,, \qquad |p] \mapsto \sqrt{p^+}|p] \tag{4.16}$$

After the transformation of eq. (4.16) is applied the square roots all cancel. Another possible problem arising is the fact that there are values of the momentum for which these components are singular. So we use the following case dependent definitions:

- for $p^+ \neq 0$

$$|p\rangle = \begin{pmatrix} 1 \\ \dfrac{p^1 + ip^2}{p^+} \end{pmatrix} \qquad |p] = \begin{pmatrix} p^1 - ip^2 \\ -p^+ \end{pmatrix} \tag{4.17a}$$

- for $p^+ = 0$ and $p_1 + ip_2 \neq 0$

$$|p\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |p] = \begin{pmatrix} p^- \\ -(p^1 + ip^2) \end{pmatrix} \tag{4.17b}$$

- for $p^+ = 0$ and $p_1 + ip_2 = 0$

$$|p\rangle = \begin{pmatrix} p_1 - ip_2 \\ p^- \end{pmatrix} \qquad |p] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{4.17c}$$

Using these conventions all the intermediate steps as well as the output of the BCFW algorithm are rationals, provided the input spinor components are parametrized by momentum-twistor variables.

The numerical computation of spinor components starting from the momentum is performed by the routine called `SpinorFromMom`. It takes as input the list of four momentum components and returns an object of the class `Components`, where $|p\rangle, |p]$ are computed via eq. (4.17) and $\langle p|, [p|$ through the map to dual space defined by the two dimensional ϵ^{ij} tensor. In matrix notation:

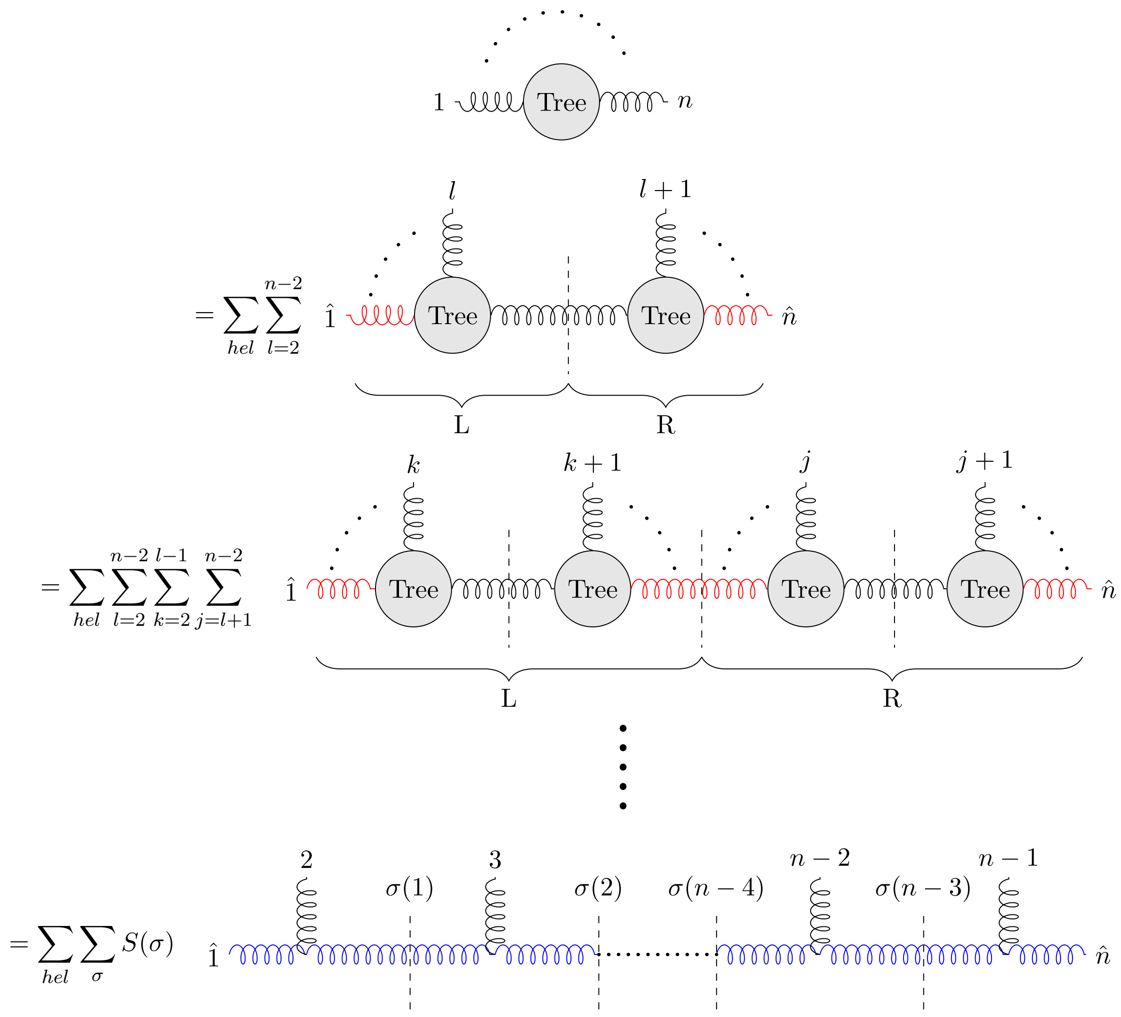
$$\epsilon = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \epsilon^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{4.18}$$

$$\langle p| = \epsilon|p\rangle \qquad [p| = \epsilon^{-1}|p] \tag{4.19}$$

The recurrence relation

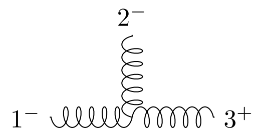
A diagrammatic representation of the recursion is given in figure 4.8. We adopted the convention of always injecting a complex component into the momenta of the first and last particle involved in the amplitude. In the figure we have that:

- red lines are associated to the particles whose momentum is shifted in the given step of the recurrence. These particles get and additional complex component which allows the intermediate states cut by the dashed line to go on-shell.
- Blue lines in the output are on-shell particles with complex momenta.
- The sum $\sum_{hel}$ runs over all the possible helicity configurations of all the intermediate states present
- The sum $\sum_{\sigma}$ is a sum over all the possible permutations of $n-3$ objects, where $n-3$ is the total number of propagators which are put on-shell once the recurrence is terminated.

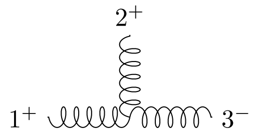


1
Tree
n
$= \sum_{hel} \sum_{l=2}^{n-2}$
l
$l+1$
$\hat{1}$
$\hat{n}$
L
R
$= \sum_{hel} \sum_{l=2}^{n-2} \sum_{k=2}^{l-1} \sum_{j=l+1}^{n-2}$
k
$k+1$
j
$j+1$
$= \sum_{hel} \sum_{\sigma} S(\sigma)$
2
3
$n-2$
$n-1$
$\sigma(1)$
$\sigma(2)$
$\sigma(n-4)$
$\sigma(n-3)$

Figure 4.8: Diagrammatic representation of the BCFW algorithm. Red lines are associated to the particles whose momentum is shifted at the given step of the recursion. Blue lines in the output are associated to complex on- shell states.



$$= \frac{\langle 1|2\rangle^4}{\langle 1|2\rangle\langle 2|3\rangle\langle 3|1\rangle}$$



$$= -\frac{[1|2]^4}{[1|2][2|3][3|1]}$$

Figure 4.9: Three-point vertexes with complex kinematics.

For a given helicity configuration of all the particles, the addenda of the output of the BCFW factorization are distinguished by the order in which the intermediate states were put on-shell. Recall that the starting point of the recursive relation was the equation

$$i\mathbf{M}(z=0) = -\sum_{poles\ \alpha} \underset{z=z_\alpha}{\mathbf{Res}} \frac{i\mathbf{M}(z)}{z} \tag{4.20}$$

and each factorization in two sub-amplitudes corresponds to one of these residues. Each residue is characterized by a different value of the pole. Each pole in turn is defined by the momenta of the external particles appearing in the amplitude, and defines new complex momenta of the particles of the sub-amplitudes resulting from the factorization. Thus in general the i-th pole depends on the values of all the j-th poles with $j < i$ computed that far, which is why we can distinguish the completely factorized output diagrams basing on the order the internal states were put on-shell. Some of the permutations however yield repeating results, due to the specific symmetry of the factorized amplitudes. In order to avoid double counting a symmetry factor $S(\sigma)$ is introduced. This is necessary only because of the specific labelling of the diagrams in terms of permutations of propagators. When generating the result recursively no double counting occurs.

There are specific orderings of the propagators which result in the same value of one or more poles along the computation. These values also depend on the specific helicity configuration which picks the correct expression out of two possibilities, see section 1.5. When the two particles which are shifted are in the same helicity state, both possibilities are admitted. So for a smart choice of the pole expressions, the double sum $\sum_{hel}\sum_{\sigma}$ can be reduced to a sum over few independent diagrams only. These cannot be immediately spotted when writing the recurrence output in terms of the complete factorization in three-point amplitudes, since all diagrams have exactly the same form. However it is possible to establish an analogy between the BCFW recurrence and the maximum-cut of multi-loop diagrams, which is a tool of the unitarity-based framework for the computation of higher-order corrections to scattering amplitudes. Through this analogy BCFW recursion can be represented in terms of multi-loop diagrams, and the independent ones can be easily found. We will talk about this in more detail in chapter 5.

The routine performing the computation of BCFW recurrence over $\mathbb{Q}$ is called `BCFW`. Differently from Berends-Giele recursion, where the helicities of the particles were used only in the beginning to compute the polarization vectors, here helicity labels need to be kept track of through out the entire computation. This is due to the fact that the momentum shift as well as the final numerical evaluation of the three-point amplitudes, arising in the factorization of the complete amplitude, depend on the helicities of external as well as internal on-shell states. It is thus useful to define a new class of objects, providing all the spinor components in the same form as `Components`, and additionally the helicity:

$$\begin{aligned} \texttt{Part[i]} &= \{\{\texttt{SR[i]},\texttt{SL[i]}\},\{\texttt{LS[i]},\texttt{RS[i]}\},\texttt{hel[i]}\} \\ &= \{\texttt{Components[i]},\texttt{hel[i]}\} \end{aligned} \tag{4.21}$$

where `hel[i]` assumes the two values `Hplus` and `Hminus` for gluons with positive/negative helicity respectively.

The routine BCFW is shown in algorithm 9 and the corresponding flowchart is represented in figure 4.10. Once a three-point amplitude is encountered by BCFW it returns its numerical value evaluated through M3, which takes as input three elements of the class Part and returns the numerical value of the associated MHV or anti-MHV depending on the helicities computed via eq. (1.140) and eq. (1.141). The output of BCFW is either a number, in the case of $n = 3$, or again a function of BCFW which is thus recursively substituted until only numbers remain. Calling cxmom the complex momentum injected into the first and last particle, and Q the momentum flowing along the propagator which is put on-shell:

$$\begin{aligned}\texttt{BCFW[Part[1],...,Part[n]]} =& \\ &\sum_{hel}\texttt{BCFW[Part[1+cxmom],...,Part[Q+cxmom]]}\frac{-1}{Q^2} \\ &\texttt{BCFW[Part[-(Q+cxmom)],...,Part[n-cxmom]]}\end{aligned} \tag{4.22}$$

Where $\sum_{hel}$ runs over the possible helicity states of the internal particle going on-shell. The value of the spinor components appearing in Part[i $\pm$ cxmom] are computed through eq. (4.17). The analytic expressions of the pole and the associated complex momentum injection can be found in section 1.5.4.

```
input  : {Part[1], ··· ,Part[n]}
output: bcfw

bcfw=0;
if n=3 then
    bcfw=M3[Part[1],Part[2],Part[3]];
else
    for i=1 to n do
        p[i]=MomFromSpinor[i];
    end
    for l=2 to n-2 do
        compute value of the pole z;
        cxmom= complex momentum injected depending on z;
        bcfw=bcfw+eq. (4.22);
    end
end
```

Algorithm 9: BCFW recursion, the analytic expressions of the quantities used can be found in section 1.5.4.

An example of a six-point amplitude computed via BCFW is displayed in figure 4.11, the helicity configurations used are the same as those of figure 4.7. The function tobcfwform converts the input given in the form of a list of Components plus two lists of positive and negative helicity gluons, suitable for the BGAt2, into the Part[i] form required by BCFW.

Notice that again all the results are rational numbers as desired. The values we computed here are the amplitudes with an overall factor stripped off, which is why the values computed with the BCFW recurrence differ from those computed via the Berend-Giele recursion. The actual value of the amplitudes can be obtained by restoring the appropriate factors called respectively overallfactorbcfw and overallfactor, which depend only on the number of external legs. See figure 4.12.

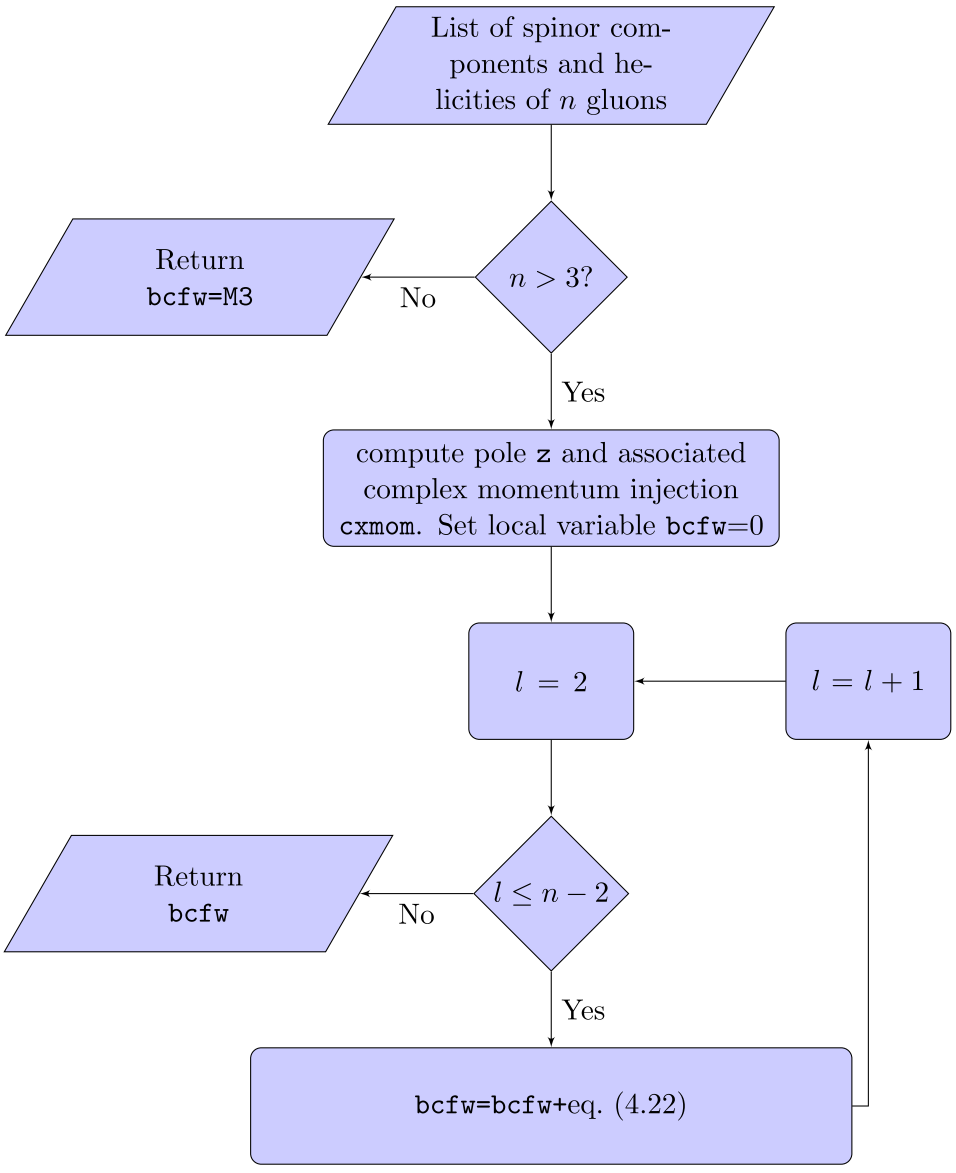
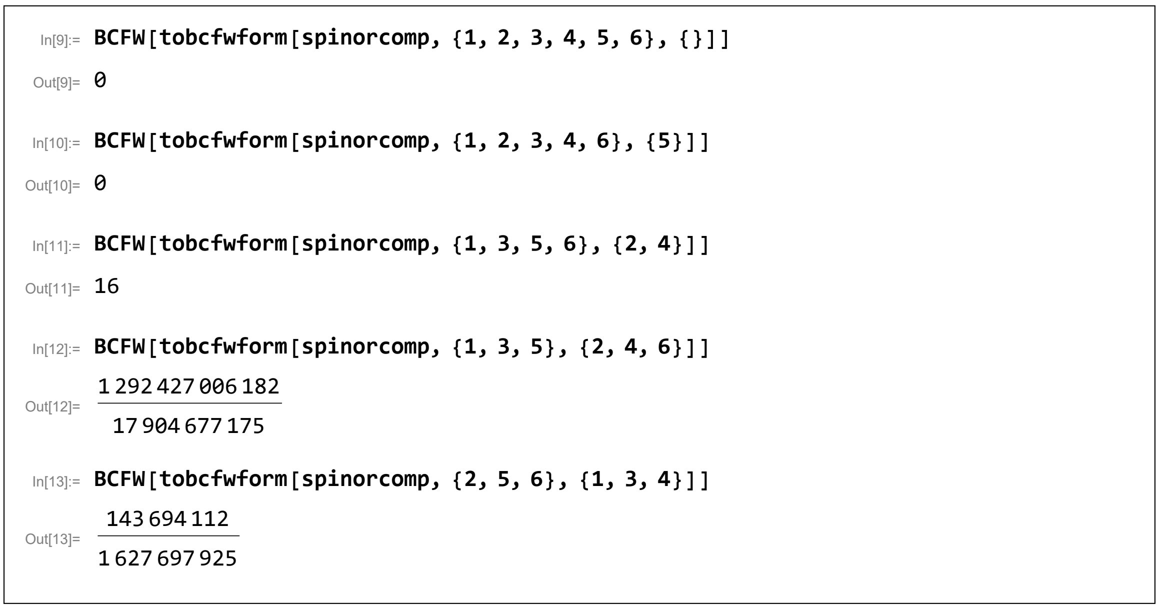
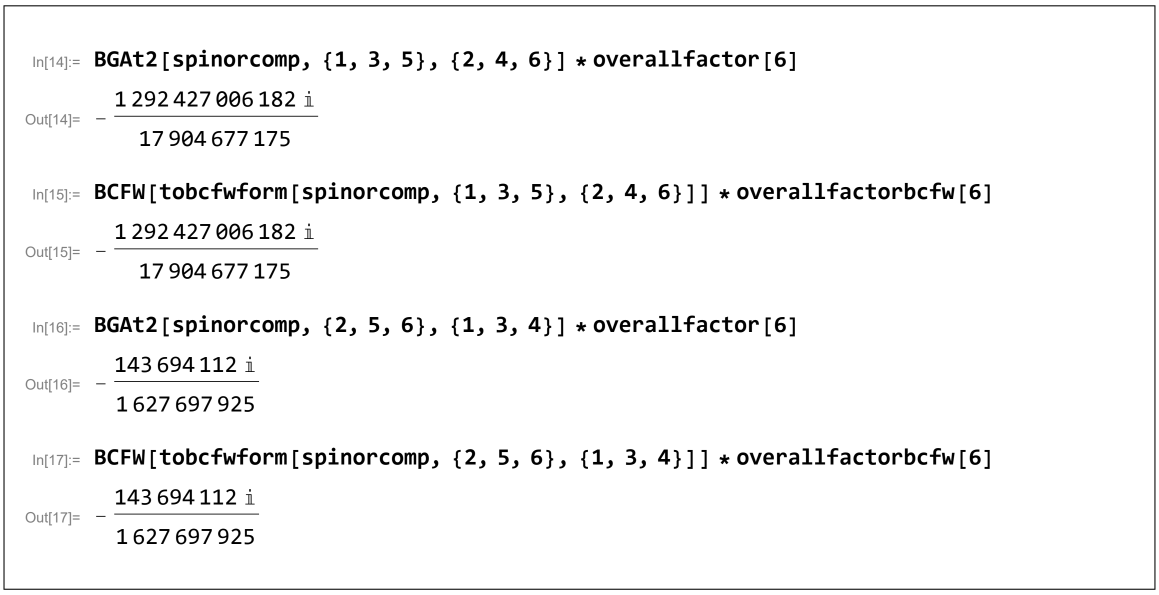

List of spinor components and helicities of n gluons
$n > 3$?
No
Return bcfw=M3
Yes
compute pole z and associated complex momentum injection cxmom. Set local variable bcfw=0
$l = 2$
$l = l + 1$
$l \le n - 2$
No
Return bcfw
Yes
bcfw=bcfw+eq. (4.22)

Figure 4.10: Flowchart of BCFW algorithm.



```
In[9]:=  BCFW[tobcfwform[spinorcomp, {1, 2, 3, 4, 5, 6}, {}]]

Out[9]=  0

In[10]:= BCFW[tobcfwform[spinorcomp, {1, 2, 3, 4, 6}, {5}]]

Out[10]= 0

In[11]:= BCFW[tobcfwform[spinorcomp, {1, 3, 5, 6}, {2, 4}]]

Out[11]= 16

In[12]:= BCFW[tobcfwform[spinorcomp, {1, 3, 5}, {2, 4, 6}]]

         1 292 427 006 182
Out[12]= -----------------
           17 904 677 175

In[13]:= BCFW[tobcfwform[spinorcomp, {2, 5, 6}, {1, 3, 4}]]

          143 694 112
Out[13]= -------------
         1 627 697 925
```

Figure 4.11: Example of six-point amplitudes in different helicity configurations computed through BCFW recursion. The spinor components are parametrized by momentum-twistor variables and generated via `TwisTs`.



```
In[14]:= BGAt2[spinorcomp, {1, 3, 5}, {2, 4, 6}] * overallfactor[6]

           1 292 427 006 182 ⅈ
Out[14]= - -------------------
             17 904 677 175

In[15]:= BCFW[tobcfwform[spinorcomp, {1, 3, 5}, {2, 4, 6}]] * overallfactorbcfw[6]

           1 292 427 006 182 ⅈ
Out[15]= - -------------------
             17 904 677 175

In[16]:= BGAt2[spinorcomp, {2, 5, 6}, {1, 3, 4}] * overallfactor[6]

           143 694 112 ⅈ
Out[16]= - -------------
           1 627 697 925

In[17]:= BCFW[tobcfwform[spinorcomp, {2, 5, 6}, {1, 3, 4}]] * overallfactorbcfw[6]

           143 694 112 ⅈ
Out[17]= - -------------
           1 627 697 925
```

Figure 4.12: Comparison of some of the numerical values of a six-point amplitude computed via BCFW and Berends-Giele recurrence

4.2 Amplitudes from finite fields

Now we get to discuss the rational reconstruction of scattering amplitudes over finite fields. In order for the Berends-Giele and BCFW recurrences to be applicable over finite fields and to be a suitable input for the reconstruction algorithm one needs to consider that

- the output of the recurrences is not the object we are going to reconstruct, an overall phase Φ_n must still be removed

- the internal operations of the routines performing the computation of the numerical value of the amplitude must be transposed onto finite fields
- the standard output of these routines must be compatible with the reconstruction algorithm even when the evaluation failed

After dealing with each of these topics we will be giving an example of a four-point function.

4.2.1 Factorization of the amplitude

The complete color-ordered amplitude can be factorized as

$$\mathbf{M}_n(1, \cdots, n) = C(i, \sqrt{2})\widetilde{\mathbf{M}}_n^{R.R.}(1, \cdots, n) \tag{4.23}$$

where the coefficient $C(i, \sqrt{2})$ contains the entire dependence of the amplitude on imaginary and irrational factors. The analytic expression of C depends on whether we are using Berends-Giele or BCFW recurrence to compute the amplitude, and was given respectively in section 4.1.2 and section 4.1.3. It depends only on the number of external legs of the considered process.

$\widetilde{\mathbf{M}}_n^{R.R.}$ is a rational number and it is the output of the recurrence relations as we implemented them. However $\widetilde{\mathbf{M}}_n^{R.R.}$ depends on $4n$ variables, and not only on $3n-10$ independent ones. As discussed in section 1.3 a phase Φ_n can be extracted, the resulting function depends only on the independent variables x_i:

$$\widetilde{\mathbf{M}}_n(x_1, \cdots, x_{3n-10}) = \frac{\widetilde{\mathbf{M}}_n^{R.R.}(1, \cdots, n)}{\Phi_n(1^{h_1}, \cdots, n^{h_n})} \tag{4.24}$$

This is the object whose analytic expression we are going to reconstruct.

The value of Φ_n can be computed once the spinor components and the helicities of all the external particles are known. This phase is not uniquely defined, a possible expression[6] was given in section 1.3.1 we report it once again:

$$\Phi_n(1^{h_1}, \cdots, n^{h_n}) = \left(\frac{\langle 1|3\rangle}{[1|2]\langle 2|3\rangle}\right)^{h_1} \prod_{i=2}^{n} \left(\frac{\langle 1|i\rangle^2 [1|2]\langle 2|3\rangle}{\langle 1|3\rangle}\right)^{-h_i} \tag{4.25}$$

Since we are using momentum-twistor parametrization for the amplitude, the spinor components in eq. (4.25) are all rational. But at the same time Φ_n is a phase, so it can take on only the values

$$\Phi_n(1^{h_1}, \cdots, n^{h_n}) = \pm 1 \tag{4.26}$$

In the analytic expression of the phase, eq. (4.25), divisions are present so also Φ_n may present vanishing denominators and must be checked each time it is computed.

4.2.2 Spinor helicity formalism over finite fields

Inside the recurrence relations all the four-vectors have been defined in light-cone components. In particular this was done so that the momenta, constructed in terms of the

[6]Derived by Simon Badger in [2].

two-dimensional rational spinor components, were rational themselves. The associated routine has the form

$$\texttt{MomFromSpinorLightC[i]} = \frac{1}{2}\begin{pmatrix} \texttt{LS[i].IdentityMatrix[2].SL[i]+LS[i].PauliMatrix[3].SL[i]} \\ \texttt{LS[i].IdentityMatrix[2].SL[i]-LS[i].PauliMatrix[3].SL[i]} \\ \texttt{LS[i].PauliMatrix[1].SL[i]+I*LS[i].PauliMatrix[2].SL[i]} \\ \texttt{LS[i].PauliMatrix[1].SL[i]-I*LS[i].PauliMatrix[2].SL[i]} \end{pmatrix} \tag{4.27}$$

This needs to be transposed to finite fields. The matrix dot product can be seen as the standard product applied multiple times, and can thus be implemented in terms of `ffprod`, call it `ffprodMatrix`. For a generic matrix product `M.N` we define `ffprodMatrix` as

```
input  : p=prime, M=matrix, N=matrix
output: matrix A
rowsM=number of rows of M;
colsM=number of columns of M;
rowsN=number of rows of N;
colsN=number of columns of N;
if $rowsN \neq colsM$ then
    return error message;
else
    for $i = 1$ to $rowsM$ do
        for $k = 1$ to $colsN$ do
            for $j = 1$ to $colsM$ do
                $A_{ik}$ = ffprod[$p, M_{ij}, N_{jk}$];
            end
        end
    end
end
```

Algorithm 10: Matrix product over finite fields `ffprodMatrix`

We will call this matrix product symbolically $\circ_p$.

The matrix σ^2 in eq. (4.27) contains imaginary factors so we cannot map everything directly to $\mathbb{Z}_p$ and then perform matrix multiplication. Instead we define a new object

$$\begin{aligned} &\texttt{ModifiedPauliM[i]=PauliMatrix[i]} \quad \text{for} \quad i = 1, 3 \\ &\texttt{ModifiedPauliM[i]=I*PauliMatrix[i]} \quad \text{for} \quad i = 2 \end{aligned} \tag{4.28}$$

All the components of `ModifiedPauliM` are integers. We can then compute the momenta in light-cone basis in two steps, first compute:

$$\texttt{ModifiedSL[i,j]=ModifiedPauliM[i]} \circ_p \texttt{SL[j]} \tag{4.29}$$

and then the complete form of the momentum

$$\texttt{MomFromSpinorff[i]} = \frac{1}{2}\begin{pmatrix} \texttt{LS[i]} \circ_p \texttt{SL[i]+LS[i]} \circ_p \texttt{ModifiedSL[3,i]} \\ \texttt{LS[i]} \circ_p \texttt{SL[i]-LS[i]} \circ_p \texttt{ModifiedSL[3,i]} \\ \texttt{LS[i]} \circ_p \texttt{ModifiedSL[1,i]+LS[i]} \circ_p \texttt{ModifiedSL[2,i]} \\ \texttt{LS[i]} \circ_p \texttt{ModifiedSL[1,i]-LS[i]} \circ_p \texttt{ModifiedSL[2,i]} \end{pmatrix} \tag{4.30}$$

4.2.3 Recurrence relations over finite fields

The algorithms performing the Berends-Giele and BCFW recurrences on finite fields will be called `BGoverff` and `BCFWff` respectively. Since imaginary and irrational factors were already removed from the recurrences factoring out $C(i, \sqrt{2})$, we just need to substitute the standard operations on $\mathbb{Q}$ with the $+_p$, $\times_p$ and $\circ_p$ defined so far. The prime p characterizing the field over which the computation is taking place will be given as an additional input to the routines. The recurrences are then ready to be run on $\mathbb{Z}_p$. Their output will be an integer corresponding to the numerical value of $\widetilde{\mathbf{M}}_n^{R.R.}$ on $\mathbb{Z}_p$.

Vanishing denominators

Along the computation of both the recurrence relations divisions are performed and vanishing denominators may appear. In particular in Berends-Giele non-invertible zeros can be found when

- computing the polarization vectors from the spinor components associated to the external particles
- computing propagators in the intermediate steps of the recurrence

whereas for BCFW when

- computing the value of the pole which takes intermediate states on-shell

If a non-invertible zero is encountered the evaluation of the amplitude is interrupted and as output an appropriate error message string is returned. In the following example two computations are displayed: a successful one returning an integer and a failed one returning an error message.

```
In[15]:= BGoverff[101, test, {1, 3, 5}, {2, 4, 6}]
Out[15]= 44

In[16]:= BGoverff[23, test, {1, 3, 5}, {2, 4, 6}]
Out[16]= Non invertible zero in B.G.
```

The input called `test` is the list of spinor components defining the external particles.

The spinor components themselves could present vanishing denominators for particular values of the `x[i]`. For example generating a parametrization of a five-point amplitude with `TwisTs` and considering the spinor components of the last particle, we see that `x[1]=0` and `x[2]=x[3]` are not acceptable:

```
In[5]:= TwisTs[5, x][[5]]
```

Out[5]= $$\left\{\left\{\left\{\frac{x[3]}{x[1]}, 1\right\}, \left\{\frac{x[1]\,(1+x[5])}{x[2]-x[3]}, -\frac{-x[1]-x[1]\,x[4]+x[1]\,x[2]\,x[4]}{x[2]-x[3]}\right\}\right\}, \left\{\left\{1, -\frac{x[3]}{x[1]}\right\}, \left\{\frac{-x[1]-x[1]\,x[4]+x[1]\,x[2]\,x[4]}{x[2]-x[3]}, \frac{x[1]\,(1+x[5])}{x[2]-x[3]}\right\}\right\}\right\}$$

So the input of the recurrences must also be checked.

The reconstruction

The function which removes the phase and checks the evaluation for error messages, call it `RecurrenceChecked`,[7] is described in algorithm 11. It takes as input one prime `p` and three lists: `comp` is the list of spinor components in form `Components[i]`,[8] `lplus` is the list of positive helicity gluons and `lminus` of negative helicity gluons. The output is either the numerical value of $\widetilde{\mathbf{M}}_n(x_1, \cdots, x_{3n-10})$ or $\frac{1}{0}$. The latter corresponds to a non-invertible zero encountered either in the spinor components, or during the evaluation of the recurrence, or the computation of the phase Φ_n. This form of output is compatible with the reconstruction algorithm which recognizes the $\frac{1}{0}$ of the failed evaluation and skips the associated sampling point moving on to the next one.

Φ_n is evaluated via `phase`, which takes the same input as `RecurrenceChecked`, and returns as output the numerical value of the phase computed through eq. (4.25) on finite fields.

The spinor components given as input to `RecurrenceChecked` must be already mapped to $\mathbb{Z}_p$. The function `Recurrence` can be either `BGoverff` or `BCFWff`.

```
input  : p,comp,lplus,lminus
output: out
if any element of comp contains $\frac{1}{0}$ then
    out=$\frac{1}{0}$;
else
    ph=phase[p,comp,lplus,lminus];
    if ph contains $\frac{1}{0}$ then
        out=$\frac{1}{0}$;
    else
        rec=Recurrence[p,comp,lplus,lminus];
        if rec=String then
            out=$\frac{1}{0}$;
        else
            out=rec/ph ;
        end
    end
end
```

Algorithm 11: `RecurrenceChecked` removes the phase and checks the recurrences for error messages.

The whole reconstruction process of an n-point tree-level scattering amplitude, given a certain helicity configuration, proceeds as follows:

[7] `Recurrence` stands for either `BG` or `BCFW`.

[8] These were defined in section 4.1.

$$\begin{tikzcd}[row sep=large]
\text{Generate momentum-twistor parametrization} \arrow[d] \\
\text{Map the parametrization to } \mathbb{Z}_p \arrow[d, "\text{input for}"] \\
\left[\widetilde{\mathbf{M}}_n(x_1, \cdots, x_{3n-10})\right]_p \arrow[d, "\text{input for}"] \\
\text{Rational reconstruction algorithm on } \mathbb{Z}_p \arrow[d, "\text{returns}"] \\
\text{Analytic expression of } \widetilde{\mathbf{M}}_n(x_1, \cdots, x_{3n-10}) \text{ on } \mathbb{Q} \arrow[d, "\text{multiply by } \Phi_n \times C(i, \sqrt{2})"] \\
\text{Analytic expression of the colour-ordered amplitude } \mathbf{M}_n(1^{h_1}, \cdots, n^{h_n})
\end{tikzcd}$$

Figure 4.13: Schematic overview of the complete reconstruction process of a tree-level color-ordered scattering amplitude. The objects Φ_n, $C(i, \sqrt{2})$ and $\widetilde{\mathbf{M}}_n$ are defined in section 4.2.1.

A more detailed description of the procedure outlined in figure 4.13 is given in algorithm 12.

```
input  : n=number of external legs, hel=helicity configuration
output: out=analytic expression of amplitude
comp[x]=TwisTs[n,x] ;          /* momentum-twistor parametrization of an
  n-point amplitude */
compff[p,x]=ffmap[p,comp[x]] ;                   /* image of comp on $\mathbb{Z}_p$ */
Fin[p',x]=RecurrenceCheceked[p,compff[p,x],hel] ; /* black-box algorithm for
  the reconstruction */
out=ChineseRec[Fin] ;             /* run the reconstruction algorithm */
out=out*phase[comp]*overallfactor[n] ;   /* reintroduce the overall factor
  and the phase */
```
Algorithm 12: Complete reconstruction of the nalytic expression of a scattering amplitude

In the following section an example is presented.

4.2.4 A four-point example

Consider a four-point amplitude with the helicity configuration $\{1^+, 2^-, 3^+, 4^-\}$. It can be described by $3 \times 4 - 10 = 2$ independent variables $\{x_1, x_2\}$. As seen in section 1.3 these can be taken to be

$$x_1 = s_{12} \qquad x_2 = \frac{s_{23}}{s_{12}} \tag{4.31}$$

Set $x_1 = 1$ and perform the reconstruction in x_2. The result will be the analytic expression of $\mathbf{M}_4$ in terms of x_2. The full amplitude can then be obtained by dimensional analysis multiplying by the appropriate power of x_1 which has mass dimension two. It is thus possible to obtain the expression of the amplitude performing only an univariate reconstruction. For the numerical calculation of the amplitude we are going to use the BCFW recurrence relation. The functions in the example correspond to:

- `test` is the symbolic form of the momentum-twistor parametrization generated via `TwisTs` after substitution $x_1 = 1$

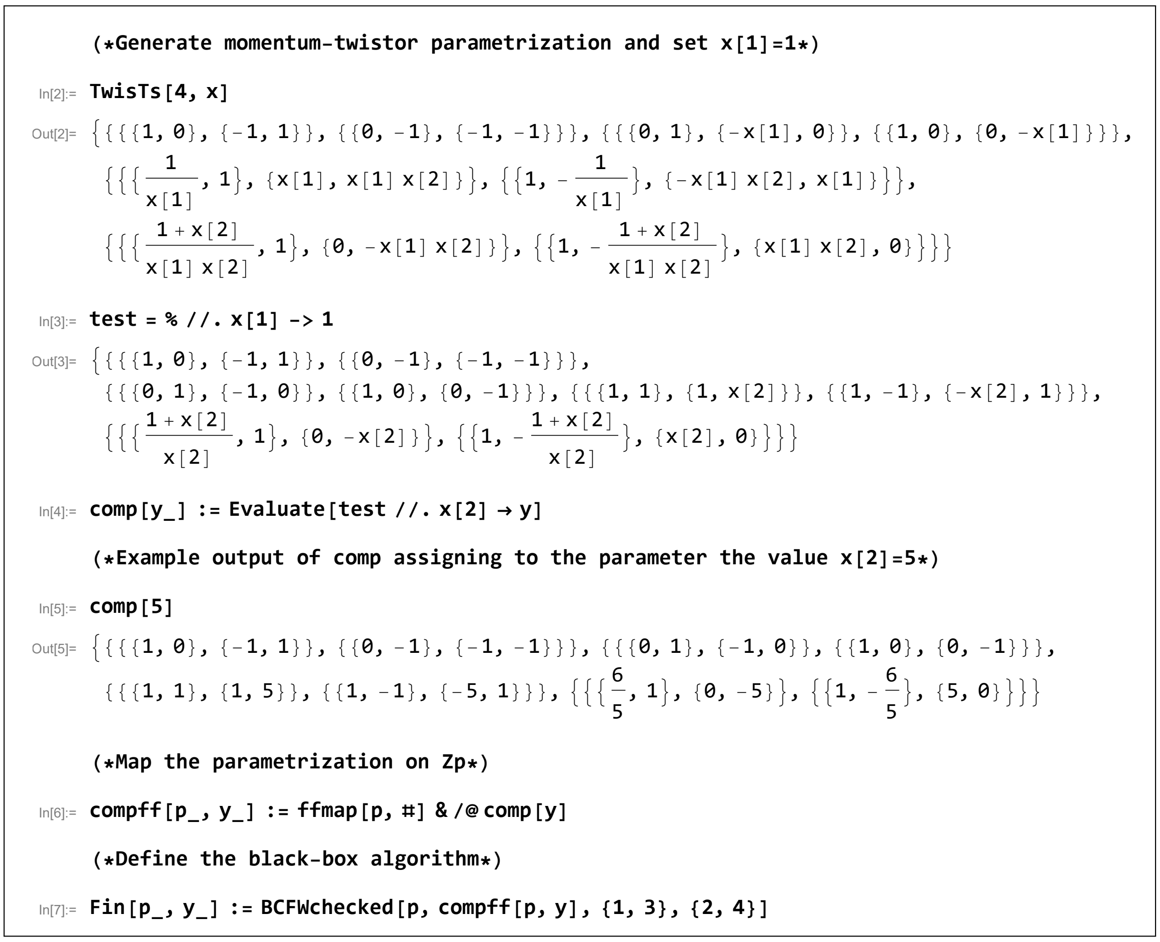
- `comp` is the function associated to `test`

- `compff` is the image of the parametrization on $\mathbb{Z}_p$

- `Fin` is the black-box algorithm which will be given as input to the reconstruction. It is defined in terms of `BCFWchecked` which takes as input the spinor components on $\mathbb{Z}_p$ generated by `compff`.

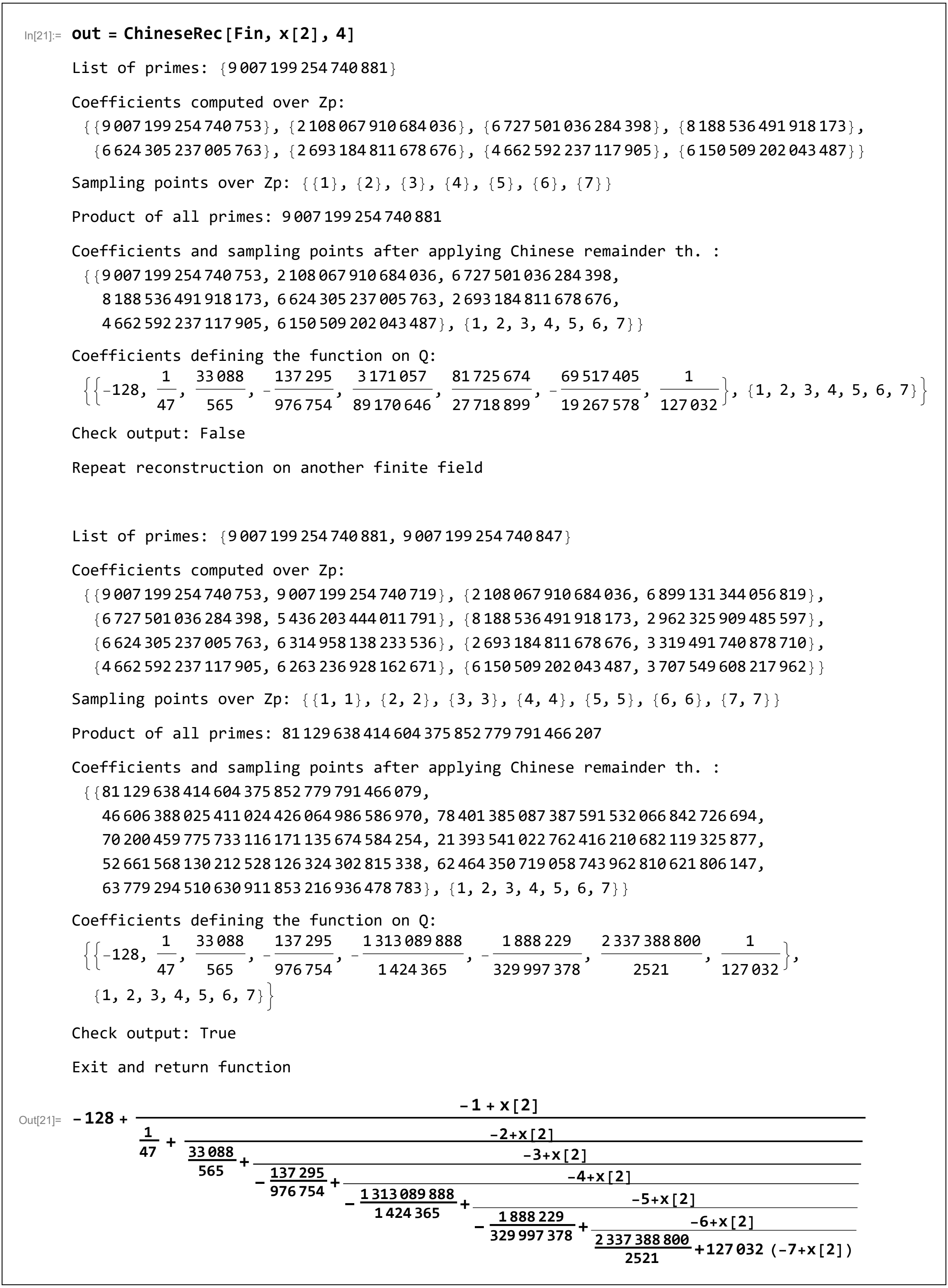
- `out` is the analytic expression of $\widetilde{\mathbf{M}}_n(x_1 = 1, x_2)$ obtained by `ChineseRec`, which performs the rational reconstruction on finite fields using the Chinese remainder theorem

- `out` is then simplified and multiplied by the appropriate phase and overall factor $\Phi_n \times C(i, \sqrt{2})$



```
        (*Generate momentum-twistor parametrization and set x[1]=1*)

In[2]:= TwisTs[4, x]

Out[2]= {{{{1, 0}, {-1, 1}}, {{0, -1}, {-1, -1}}}, {{{0, 1}, {-x[1], 0}}, {{1, 0}, {0, -x[1]}}},
             1                                       1
         {{{----, 1}, {x[1], x[1] x[2]}}, {{1, - ----}, {-x[1] x[2], x[1]}}},
            x[1]                                    x[1]
             1 + x[2]                                     1 + x[2]
         {{{---------, 1}, {0, -x[1] x[2]}}, {{1, - ---------}, {x[1] x[2], 0}}}}
            x[1] x[2]                                   x[1] x[2]

In[3]:= test = % //. x[1] -> 1

Out[3]= {{{{1, 0}, {-1, 1}}, {{0, -1}, {-1, -1}}},
         {{{0, 1}, {-1, 0}}, {{1, 0}, {0, -1}}}, {{{1, 1}, {1, x[2]}}, {{1, -1}, {-x[2], 1}}},
             1 + x[2]                         1 + x[2]
         {{{--------, 1}, {0, -x[2]}}, {{1, - --------}, {x[2], 0}}}}
               x[2]                             x[2]

In[4]:= comp[y_] := Evaluate[test //. x[2] → y]

        (*Example output of comp assigning to the parameter the value x[2]=5*)

In[5]:= comp[5]

Out[5]= {{{{1, 0}, {-1, 1}}, {{0, -1}, {-1, -1}}}, {{{0, 1}, {-1, 0}}, {{1, 0}, {0, -1}}},
                                                  6                          6
         {{{1, 1}, {1, 5}}, {{1, -1}, {-5, 1}}}, {{{-, 1}, {0, -5}}, {{1, - -}, {5, 0}}}}
                                                  5                          5

        (*Map the parametrization on Zp*)

In[6]:= compff[p_, y_] := ffmap[p, #] & /@ comp[y]

        (*Define the black-box algorithm*)

In[7]:= Fin[p_, y_] := BCFWchecked[p, compff[p, y], {1, 3}, {2, 4}]
```

Figure 4.14: Set up for a 4-point amplitude reconstruction using BCFW recursion as black-box algorithm.



```
In[21]:= out = ChineseRec[Fin, x[2], 4]

         List of primes: {9 007 199 254 740 881}

         Coefficients computed over Zp:
           {{9 007 199 254 740 753}, {2 108 067 910 684 036}, {6 727 501 036 284 398}, {8 188 536 491 918 173},
            {6 624 305 237 005 763}, {2 693 184 811 678 676}, {4 662 592 237 117 905}, {6 150 509 202 043 487}}

         Sampling points over Zp: {{1}, {2}, {3}, {4}, {5}, {6}, {7}}

         Product of all primes: 9 007 199 254 740 881

         Coefficients and sampling points after applying Chinese remainder th. :
           {{9 007 199 254 740 753, 2 108 067 910 684 036, 6 727 501 036 284 398,
             8 188 536 491 918 173, 6 624 305 237 005 763, 2 693 184 811 678 676,
             4 662 592 237 117 905, 6 150 509 202 043 487}, {1, 2, 3, 4, 5, 6, 7}}

         Coefficients defining the function on Q:
```

$$\left\{\left\{-128, \frac{1}{47}, \frac{33\,088}{565}, -\frac{137\,295}{976\,754}, \frac{3\,171\,057}{89\,170\,646}, \frac{81\,725\,674}{27\,718\,899}, -\frac{69\,517\,405}{19\,267\,578}, \frac{1}{127\,032}\right\}, \{1, 2, 3, 4, 5, 6, 7\}\right\}$$

```
         Check output: False

         Repeat reconstruction on another finite field


         List of primes: {9 007 199 254 740 881, 9 007 199 254 740 847}

         Coefficients computed over Zp:
           {{9 007 199 254 740 753, 9 007 199 254 740 719}, {2 108 067 910 684 036, 6 899 131 344 056 819},
            {6 727 501 036 284 398, 5 436 203 444 011 791}, {8 188 536 491 918 173, 2 962 325 909 485 597},
            {6 624 305 237 005 763, 6 314 958 138 233 536}, {2 693 184 811 678 676, 3 319 491 740 878 710},
            {4 662 592 237 117 905, 6 263 236 928 162 671}, {6 150 509 202 043 487, 3 707 549 608 217 962}}

         Sampling points over Zp: {{1, 1}, {2, 2}, {3, 3}, {4, 4}, {5, 5}, {6, 6}, {7, 7}}

         Product of all primes: 81 129 638 414 604 375 852 779 791 466 207

         Coefficients and sampling points after applying Chinese remainder th. :
           {{81 129 638 414 604 375 852 779 791 466 079,
             46 606 388 025 411 024 426 064 986 586 970, 78 401 385 087 387 591 532 066 842 726 694,
             70 200 459 775 733 116 171 135 674 584 254, 21 393 541 022 762 416 210 682 119 325 877,
             52 661 568 130 212 528 126 324 302 815 338, 62 464 350 719 058 743 962 810 621 806 147,
             63 779 294 510 630 911 853 216 936 478 783}, {1, 2, 3, 4, 5, 6, 7}}

         Coefficients defining the function on Q:
```

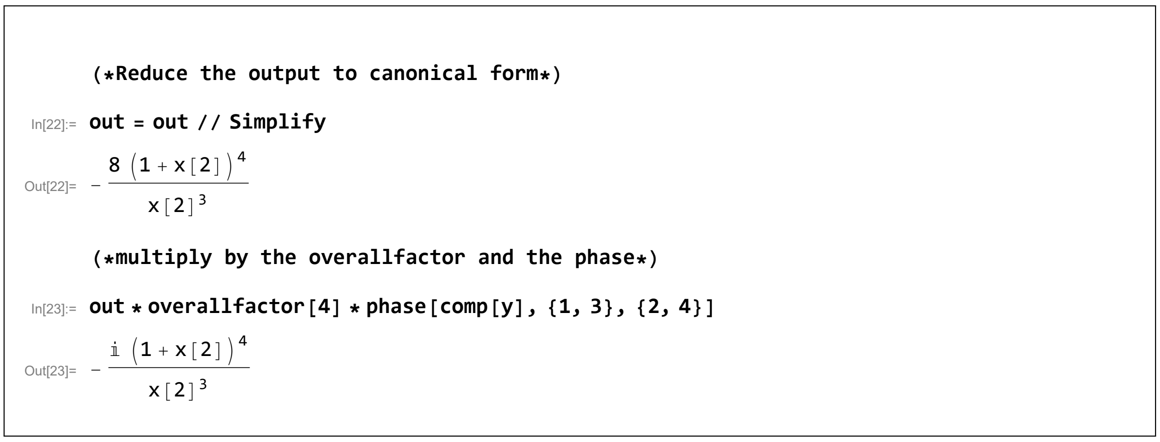
$$\left\{\left\{-128, \frac{1}{47}, \frac{33\,088}{565}, -\frac{137\,295}{976\,754}, -\frac{1\,313\,089\,888}{1\,424\,365}, -\frac{1\,888\,229}{329\,997\,378}, \frac{2\,337\,388\,800}{2521}, \frac{1}{127\,032}\right\}, \{1, 2, 3, 4, 5, 6, 7\}\right\}$$

```
         Check output: True

         Exit and return function
```

Out[21]=

$$-128+\cfrac{-1+x[2]}{\frac{1}{47}+\cfrac{-2+x[2]}{\frac{33\,088}{565}+\cfrac{-3+x[2]}{-\frac{137\,295}{976\,754}+\cfrac{-4+x[2]}{-\frac{1\,313\,089\,888}{1\,424\,365}+\cfrac{-5+x[2]}{-\frac{1\,888\,229}{329\,997\,378}+\cfrac{-6+x[2]}{\frac{2\,337\,388\,800}{2521}+127\,032\,(-7+x[2])}}}}}}$$

Figure 4.15: Reconstruction of $\widetilde{\mathbf{M}}_4$ using the Chinese remainder theorem.



```
        (*Reduce the output to canonical form*)
In[22]:= out = out // Simplify
```

$$\text{Out[22]=}\ -\frac{8\,(1+x[2])^4}{x[2]^3}$$

```
        (*multiply by the overallfactor and the phase*)
In[23]:= out * overallfactor[4] * phase[comp[y], {1, 3}, {2, 4}]
```

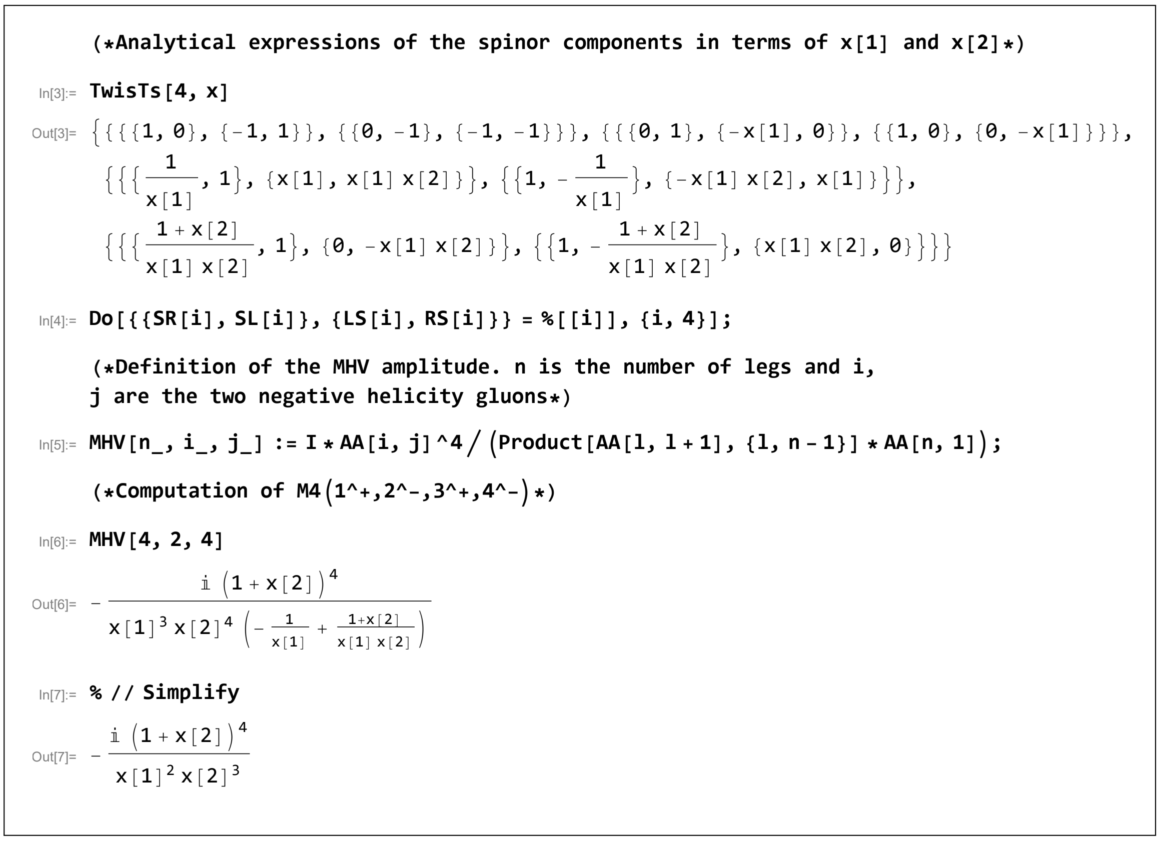
$$\text{Out[23]=}\ -\frac{i\,(1+x[2])^4}{x[2]^3}$$

Figure 4.16: Simplification of the result and extraction of the analytical expression of the 4-point amplitude.

The four-point tree-level amplitude has mass dimension -4, thus we expect its analytic expression to be given by the result obtained in figure 4.16 times x_1^{-2}:

$$\mathbf{M}_4(1^+,2^-,3^+,4^-) = -\frac{i(1+x_2)^4}{x_1^2 x_2^3} \tag{4.32}$$

This result can be checked easily since the amplitude considered is an MHV, and can thus be computed using eq. (1.79a). This has been done in Mathematica, and as can be seen from figure 4.17 our result is correct.



```
        (*Analytical expressions of the spinor components in terms of x[1] and x[2]*)
In[3]:= TwisTs[4, x]
```

$$\text{Out[3]=}\ \Big\{\{\{\{1,0\},\{-1,1\}\},\{\{0,-1\},\{-1,-1\}\}\},\{\{\{0,1\},\{-x[1],0\}\},\{\{1,0\},\{0,-x[1]\}\}\},$$
$$\Big\{\Big\{\Big\{\frac{1}{x[1]},1\Big\},\{x[1],x[1]\,x[2]\}\Big\},\Big\{\Big\{1,-\frac{1}{x[1]}\Big\},\{-x[1]\,x[2],x[1]\}\Big\}\Big\},$$
$$\Big\{\Big\{\Big\{\frac{1+x[2]}{x[1]\,x[2]},1\Big\},\{0,-x[1]\,x[2]\}\Big\},\Big\{\Big\{1,-\frac{1+x[2]}{x[1]\,x[2]}\Big\},\{x[1]\,x[2],0\}\Big\}\Big\}\Big\}$$

```
In[4]:= Do[{{SR[i], SL[i]}, {LS[i], RS[i]}} = %[[i]], {i, 4}];

        (*Definition of the MHV amplitude. n is the number of legs and i,
        j are the two negative helicity gluons*)
In[5]:= MHV[n_, i_, j_] := I * AA[i, j]^4 / (Product[AA[l, l + 1], {l, n - 1}] * AA[n, 1]);

        (*Computation of M4(1^+,2^-,3^+,4^-)*)
In[6]:= MHV[4, 2, 4]
```

$$\text{Out[6]=}\ -\frac{i\,(1+x[2])^4}{x[1]^3\,x[2]^4\left(-\frac{1}{x[1]}+\frac{1+x[2]}{x[1]\,x[2]}\right)}$$

```
In[7]:= % // Simplify
```

$$\text{Out[7]=}\ -\frac{i\,(1+x[2])^4}{x[1]^2\,x[2]^3}$$

Figure 4.17: Check of the result obtained from reconstruction using the MHV expression.

Chapter 5

On-shell recurrence relations and Maximum-cuts

The BCFW tree-level recurrence relation as well as the maximum unitarity-cut[1] are efficient tools for the analytic computation of tree-level and higher-order corrections to scattering amplitudes. Both the techniques exploit the singularity structure of the amplitudes: the multi-particle collinear behaviour in the case of tree-level amplitudes, and the leading Landau singularity in the case of one-loop amplitudes. The reconstruction of the amplitude is performed by inverting the corresponding factorization limit, from the knowledge of the sub-amplitudes which are sewn back together to form the parent amplitude that has, by constructions, the correct analytic properties.

Both techniques make use of complex momenta for the particles in order for them to be on-shell. We are going to show that the complex momenta needed to establish the tree-level recursion fulfil the four on-shellness conditions associated to the quadruple-cut of an appropriately defined one-loop diagram.

For the time being we will focus on this aspect only, however the goal of further studies will be to determine whether it is possible to extend the analogy to an exact equality modulo overall multiplicative factors for any arbitrary number of external legs. This would represent a striking relation between on-shell tree-level and multi-loop amplitudes. Moreover maximum-cut diagrams present some unique properties, in particular their residue can be shown to be a polynomial which admits a univariate representation [21].

5.1 Integrand decomposition methods

When computing loop corrections to tree-level scattering amplitudes, usually very complicated integrals appear. When a direct integration of these is prohibitive, the evaluation of scattering amplitudes beyond the leading order is in general addressed in two stages:

- the reduction in terms of an integral basis
- the evaluation of the elements of such a basis, called Master Integrals (MI)

The extraction of the coefficients of the amplitude in this basis can be achieved for example by matching the cuts on the amplitude with the cuts on the MI and solving the resulting system of equations. However, the integrand-decomposition method, proposed for the first time for the one-loop case in [23] and extended to two-loops in [18], allows to replace any

[1]An introduction to this topic can be found for example in [5].

explicit integration procedure and/or any matching procedure between cuts of amplitudes and cuts of MI's by polynomial fitting, which is a simpler operation.

Integrand-reduction methods allow the extractions of the coefficients of the MI by exploiting the multi-particle pole expansion for the integrand of the scattering amplitude, i.e. a representation where the numerator of each Feynman integral is expressed as a combination of products of the corresponding denominators, with polynomial coefficients. The crucial aspect is the shape of the residue on the multi-particle pole. Each residue is a multivariate polynomial in the irreducible scalar products (ISPs) of loop momenta and either external momenta or polarization vectors. In the residues reducible scalar products, i.e. products which can be written in terms of denominators, are not allowed to appear: else further simplification would be possible, and this contradicts the definition of residue. Thus any monomial formed by ISP's is the numerator of a potential MI which may appear in the final result.

What is to follow has been shown in [21]. For simplicity we consider 4-dimensional loop momenta. The problem of computing l-loop amplitudes with n denominators can be recast as the reconstruction of integrand functions of the type

$$\mathcal{I}_{i_1\cdots i_n} = \frac{\mathcal{N}_{i_1\cdots i_n}(k_1,\cdots,k_l)}{D_{i_1}(k_1,\cdots,k_l)\cdots D_{i_l}(k_1,\cdots,k_l)} \tag{5.1}$$

with $k_1,\cdots,k_l$ loop-momenta. The generic denominator can be written as

$$D_i = \Big(\sum_{j=1}^{l} \alpha_j k_j + p_j\Big)^2 - m_i^2\,, \qquad \alpha_i = \{0,\pm 1\} \tag{5.2}$$

with p_i external momenta.The numerator $\mathcal{I}_{i_1\cdots i_n}$ and all the denominators D_i are polynomials in the components of the loop- momenta, let us call them $\mathbf{z} = (z_1,\cdots,z_{4l})$.

Consider the ideal generated by the denominators, which is defined as

$$\begin{aligned}\mathcal{J}_{i_1\cdots i_n} &= \langle D_1,\cdots,D_n\rangle \\ &\equiv \left\{\sum_{t=1}^{n} h_t(\mathbf{z})D_{i_t}(\mathbf{z}) : h_t(\mathbf{z}) \in P[\mathbf{z}]\right\}\end{aligned} \tag{5.3}$$

where $P[\mathbf{z}]$ is the ring of polynomials in $\mathbf{z}$. The common zeros of the elements in $\mathcal{J}_{i_1\cdots i_n}$ are the same as the common zeros of the denominators. By performing multivariate-polynomial division it is possible to achieve the multi-pole decomposition of eq. (5.1), resulting in an expression of $\mathcal{N}_{i_1\cdots i_n}$ in terms of denominators and residues.

Next construct a Gröbner basis [10] (see also [22], ch. 2) generating the ideal $\mathcal{J}_{i_1\cdots i_n}$ with respect to a chosen monomial order[2]

$$\mathcal{G}_{i_1\cdots i_n} = \{g_i(\mathbf{z}),\cdots,g_m(\mathbf{z})\} \tag{5.4}$$

Then n-ple cut conditions $D_{i_1} = \cdots = D_{i_n} = 0$ are then equivalent to $g_1 = \cdots = g_m = 0$. The number of elements m of the Gröbner basis is in general different from n. Considering then the multivariate division of $\mathcal{N}_{i_1\cdots i_n}$ modulo $\mathcal{G}_{i_1\cdots i_n}$ one can write

$$\mathcal{N}_{i_1\cdots i_n}(\mathbf{z}) = \Gamma_{i_1\cdots i_n} + \Delta_{i_1\cdots i_n}(\mathbf{z}) \tag{5.5}$$

where $\Gamma_{i_1\cdots i_n}$ is a sum of products of quotients $\mathcal{Q}_i$ and divisors g_i

$$\Gamma_{i_1\cdots i_n} = \sum_{i=1}^{m} \mathcal{Q}_i(\mathbf{z})g_i(\mathbf{z}) \tag{5.6}$$

[2]We will assume lexicographic order as in [21].

whereas $\Delta_{i_1\cdots i_n}$ is the remainder of the division. Having used the Gröbner basis this remainder is uniquely defined once the monomial order is fixed. Since $\Gamma_{i_1\cdots i_n}$ belongs to the ideal $\mathcal{J}_{i_1\cdots i_n}$ it can be written in terms of the denominators as

$$\Gamma_{i_1\cdots i_n} = \sum_{t=1}^{n} \mathcal{N}_{i_1\cdots i_{t-1}i_{t+1}\cdots i_n}(\mathbf{z}) D_{i_t}(\mathbf{z}) \tag{5.7}$$

The explicit form of $\mathcal{N}_{i_1\cdots i_{t-1}i_{t+1}\cdots i_n}(\mathbf{z})$ can be found by expressing the elements of the Gröbner basis in terms of the denominators D_{i_n}.

Reducibility criterion

An integrand $\mathcal{I}_{i_1\cdots i_n}$ is said to be reducible if it can be written in terms of lower-point integrands: that happens when the numerator can be written in terms of denominators. From eq. (5.5) and eq. (5.7) it follows that an integrand is reducible if and only if the remainder $\Delta_{i_1\cdots i_n}(\mathbf{z})$ of the division modulo a Gröbner basis $\mathcal{G}_{i_1\cdots i_m}$ vanishes, in other words when the integrand belongs to the ideal $\mathcal{I}_{i_1\cdots i_n} \in \mathcal{J}_{i_1\cdots i_n}$. It can be shown [21], using the Nullstellenansatz theorem (see ch.4 of [22]), that this happens when the system of equations (5.8) defined by the cut conditions of the denominators admits no solutions.

$$\begin{cases} D_{i_1}(\mathbf{z}) = 0 \\ \quad\vdots \\ D_{i_n}(\mathbf{z}) = 0 \end{cases} \tag{5.8}$$

Substituting back eq. (5.5) and eq. (5.7) in eq. (5.1) one gets a non-homogeneous recurrence relation in the n-denominator integrand

$$\mathcal{I}_{i_1\cdots i_n} = \sum_{t=1}^{n} \mathcal{I}_{i_1\cdots i_{t-1}i_{t+1}i_n} + \frac{\Delta_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}} \tag{5.9}$$

where the integrands $\mathcal{I}_{i_1\cdots i_{t-1}i_{t+1}i_n}$ have $n-1$ denominators, and the numerator of the non- homogeneous term is the remainder $\Delta_{i_1\cdots i_n}$ of the polynomial division eq. (5.5). By construction this contains only irreducible terms with respect to $\mathcal{G}_{i_1\cdots i_n}$, thus it can be identified with the residue at the cut $(i_1, \cdots, i_n)$.

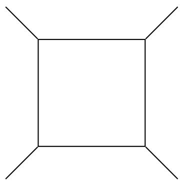
The remainders $\mathcal{I}_{i_1\cdots i_{t-1}i_{t+1}i_n}$ can be decomposed applying the same procedure, and by successive iterations the unique structure of the remainders Δ is extracted. The procedure naturally stops when all cuts are exhausted, and no denominator is left, leaving us with the integrand reduction formula.

The maximum-cut

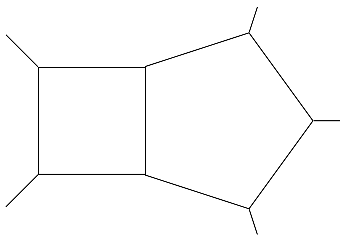
In four space-time dimensions, if the integrand $\mathcal{I}_{i_1\cdots i_n}$ is associated to a diagram presenting l loops, since each cut condition constrains one loop-momentum component the system of eq. (5.8) admits no solutions if more than $4l$ equations are present. We define then the *maximum cut* as the $4l$-cut

$$\begin{cases} D_{i_1}(\mathbf{z}) = 0 \\ \quad\vdots \\ D_{i_{4l}}(\mathbf{z}) = 0 \end{cases} \tag{5.10}$$

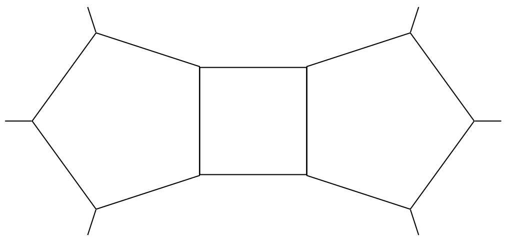
Assuming that, in non-exceptional phase space points, a maximum-cut has a finite number n_s of solutions each with multiplicity one, it can be shown [21] that



$$n_s = 2 \qquad \Delta = c_0 + c_1 z$$



$$n_s = 4 \qquad \Delta = \sum_{i=0}^{3} c_i z^i$$



$$n_s = 8 \qquad \Delta = \sum_{i=0}^{8} c_i z^i$$

Figure 5.1: Three on-shell diagrams with the associated form of the residues, corresponding to one,two and three-loop maximum cuts respectively

Theorem 5.1.1 (Maximum-cut)**.** *The residue of the maximum-cut is a polynomial parametrized by n_s coefficients, which admit an univariate representation of degree $n_s - 1$.*

The maximum-cut theorem ensures that the maximum-cut residue, at any loop, is completely determined by the n_s distinct solutions of the cut conditions. In particular it can be reconstructed by sampling the integrand on the solutions of the maximum cut itself.

5.2 One-loop 4-ple cut and BCFW

5.2.1 BCFW tree-level recurrence

In section 1.5 we saw that given a tree-level color-ordered scattering amplitude, upon injection of a complex-component momentum l^μ to the real Minkowsky kinematic values of two of the external particles, say i and j, the amplitude can be computed as a sum over the residues of an auxiliary function of one complex variable $\mathbf{M}(z)$. For an appropriate choice of l these residues are products of lower-point on-shell amplitudes with some of the external momenta taking on complex values. The analytic expression of $l(z)$ as well as that of the z^* defining the pole of a given propagator depend on the helicity configuration considered.

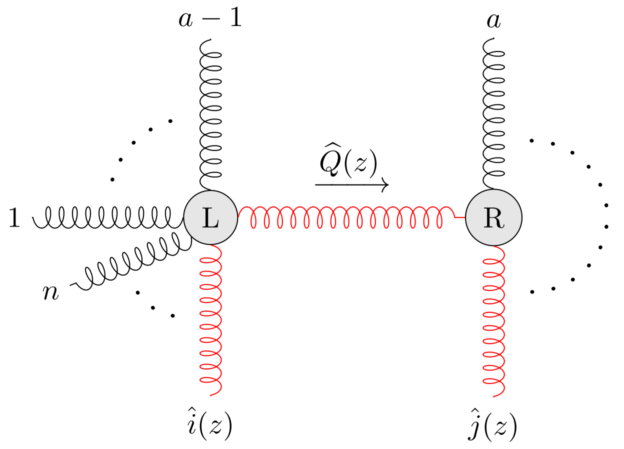


Figure 5.2: Diagram associated to one of the residues in eq. (**??**).

In particular, consider shifting *adjacent* legs. With labels of figure 5.2, define

$$p_i^\mu(z^*) = p_i^\mu + l^\mu(z^*) \tag{5.11}$$
$$p_j^\mu(z^*) = p_j^\mu - l^\mu(z^*) \tag{5.12}$$

$$Q_{m,n} = p_m + \cdots p_n \tag{5.13}$$

Then we choose the following shifts:

- For $(i,j) = \{(-,+),(+,+),(-,-)\}$

$$\begin{aligned} |\hat{i}\rangle &= |i\rangle & |\hat{i}] &= |i] + z|j] \\ |\hat{j}\rangle &= |j\rangle - z|i\rangle & |\hat{j}] &= |j] \end{aligned} \tag{5.14}$$

$$l_1^\mu(z) \equiv \frac{z}{2}\langle i|\sigma^\mu|j] \tag{5.15}$$

  and the resulting pole is given by

$$z_1^* = -\frac{Q_{i,a-1}^2}{\langle i|Q_{i,a-1}|j]} \tag{5.16}$$

- For $(i,j) = \{(+,-),(+,+),(-,-)\}$

$$\begin{aligned} |\hat{i}\rangle &= |i\rangle + z|j\rangle & |\hat{i}] &= |i] \\ |\hat{j}\rangle &= |j\rangle & |\hat{j}] &= |j] - z|i] \end{aligned} \tag{5.17}$$

$$l_2^\mu(z) \equiv \frac{z}{2}\langle j|\sigma^\mu|i] \tag{5.18}$$

and the resulting pole

$$z_2^* = -\frac{Q_{i,a-1}^2}{\langle j|Q_{i,a-1}|i]} \tag{5.19}$$

It can be shown that these shifts satisfy the border condition eq. (1.150) for the given helicity configurations. When $(i,j) = \{(+,+),(-,-)\}$ both shifts are admitted.

Notice that due to the form of $l(z)$ we have that by construction

$$\begin{cases} l(z)^2 = 0 \\ \hat{p}_i^2 = (p_i + l(z))^2 = 0 \\ \hat{p}_j^2 = (p_j - l(z))^2 = 0 \end{cases} \tag{5.20}$$

which can be shown using Fierz rearrangement eq. (1.41) and antisymmetry of the spinor product. For example considering the first type of shift with complex momentum $l_1(z)$:

$$l_1^2 = \frac{z^2}{4}2\langle i|i\rangle[j|j] = 0 \tag{5.21}$$

$$\begin{aligned} \hat{p}_i^2 &= p_i^2 + 2p_i \cdot l_1 + l_1^2 \\ &= 2\frac{\langle i|\sigma_\mu|i]}{2} z \frac{\langle i|\sigma^\mu|j]}{2} \\ &\propto \langle i|i\rangle[j|i] = 0 \end{aligned} \tag{5.22}$$

$$\begin{aligned} \hat{p}_j^2 &= p_j^2 + 2p_j \cdot l_1 + l_1^2 \\ &= 2\frac{\langle j|\sigma_\mu|j]}{2} z \frac{\langle i|\sigma^\mu|j]}{2} \\ &\propto \langle i|j\rangle[j|j] = 0 \end{aligned} \tag{5.23}$$

The equation defining the pole of the propagator z^* is

$$\widehat{Q}(z)^2 = (Q + l(z))^2 \overset{!}{=} 0 \tag{5.24}$$

Thus $l_i(z_i^*)$ for $i = 1,2$ solves the system of equations

$$\begin{cases} l(z)^2 = 0 \\ (p_i + l(z))^2 = 0 \\ (p_j - l(z))^2 = 0 \\ (Q + l(z))^2 = 0 \end{cases} \tag{5.25}$$

5.2.2 Quadruple-cut of a one-loop diagram

Consider now the one-loop diagram represented in figure 5.3.

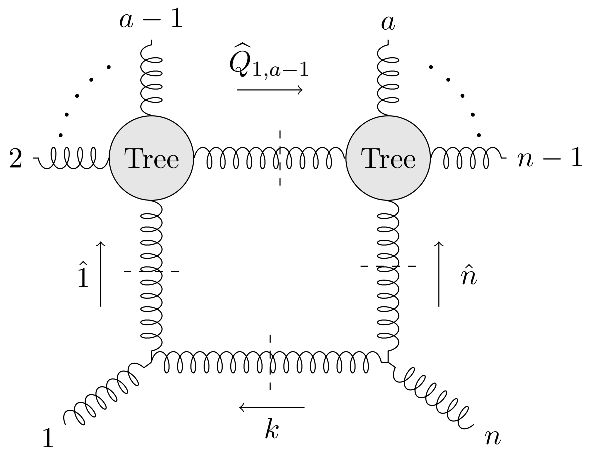


Figure 5.3: One-loop quadruple-cut of a diagram with two adjacent massless legs.

The quadruple-cut coefficient associated to this diagram is obtained by the product of the tree-level amplitudes sitting at the corners of the figure, with momenta that satisfy the cut conditions of eq. (5.26). These are obtained by requiring that the denominators of the propagators cut by dashed lines vanish. In other words that the associated intermediate states go on-shell.

$$\begin{cases} D_1 = 0 \\ D_2 = 0 \\ D_3 = 0 \\ D_4 = 0 \end{cases} \tag{5.26}$$

where D_i is the denominator associated to the i-th propagator represented in the figure. Starting from the lower propagator and proceeding clockwise, these equations read

$$\begin{cases} k^2 = 0 \\ \hat{p}_1^2 = (p_1 + k)^2 = 0 \\ \widehat{Q}_{1,a-1}^2 = (Q_{1,a-1} + k)^2 = 0 \\ \hat{p}_n^2 = (p_n - k)^2 = 0 \end{cases} \tag{5.27}$$

Upon substitution $k \to l$ the system of eq. (5.27) can be seen to coincide with that of eq. (5.25), which defines the value of the pole z^* and thus through $l(z^*)$ the shifted and internal momenta of the BCFW recursion. We are now going to solve eq. (5.27) to show that the solutions obtained are exactly the values of $l(z^*)$ presented in the preceding section.

Start by defining the four-vector

$$\epsilon_{ij}^{\mu} \equiv \frac{\langle i|\gamma^{\mu}|j]}{2} \tag{5.28}$$

and write the loop momentum k in the light-cone bases:

$$k = x_1 p_1 + x_2 p_n + x_3 \epsilon_{1n} + x_4 \epsilon_{n1} \tag{5.29}$$

Since the external particles are massless, using the first equation of the system eq. (5.27), the second and last become:

$$\begin{cases} 0 = (p_1 + k)^2 = 2x_2 p_1 \cdot p_n & \to x_2 = 0 \\ 0 = (p_n - k)^2 = -2x_1 p_1 \cdot p_n & \to x_1 = 0 \end{cases} \tag{5.30}$$

Then the equation $k^2 = 0$ can be written explicitly as:

$$\begin{aligned} 0 = k^2 &= 2p_1 \cdot p_n(x_1x_2 - x_3x_4) \\ &= -2p_1 \cdot p_n(x_3x_4) \quad \to x_3x_4 = 0 \to x_3 = 0 \vee x_4 = 0 \end{aligned} \tag{5.31}$$

Finally the last equation $\widehat{Q}^2_{1,a-1} = 0$ fixes the values of the non-zero variable among x_3 and x_4:

$$\begin{aligned} 0 = (Q_{1,a-1} + k)^2 &= Q^2_{1,a-1} + Q_{1,a-1} \cdot k \\ &= Q^2_{1,a-1} + Q_{1,a-1} \cdot (x_3\epsilon_{1n} + x_4\epsilon_{n1}) \end{aligned} \tag{5.32}$$

and so

$$x_3 = -\frac{Q^2_{1,a-1}}{Q_{1,a-1} \cdot \epsilon_{1n}} \;\wedge\; x_4 = 0 \tag{5.33}$$

or

$$x_3 = 0 \;\wedge\; x_4 = -\frac{Q^2_{1,a-1}}{Q_{1,a-1} \cdot \epsilon_{n1}} \tag{5.34}$$

If we choose $i = 1$ and $j = n$ in the BCFW shift equations of the preceding section, the two $k(x_1, x_2, x_3, x_4)$ which solve eq. (5.27) read:

$$k(0,0,x_3,0) = -\frac{Q^2_{1,a-1}}{Q_{1,a-1} \cdot \epsilon_{1n}}\epsilon_{1n} = l_1(z_1^*) \tag{5.35}$$

and

$$k(0,0,0,x_4) = -\frac{Q^2_{1,a-1}}{Q_{1,a-1} \cdot \epsilon_{n1}}\epsilon_{n1} = l_2(z_2^*) \tag{5.36}$$

We can then establish the following correspondence:

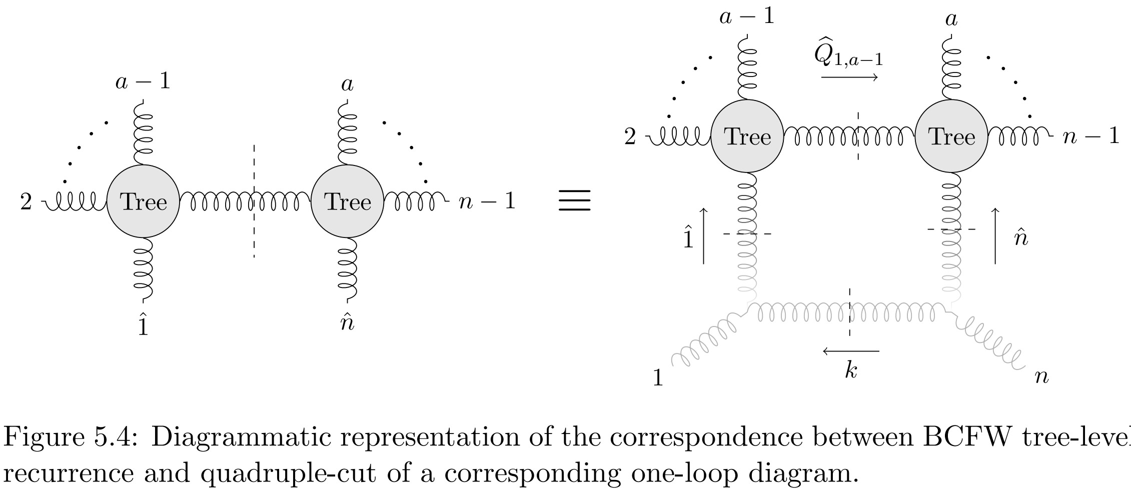


Figure 5.4: Diagrammatic representation of the correspondence between BCFW tree-level recurrence and quadruple-cut of a corresponding one-loop diagram.

The three-point amplitudes sitting at the bottom corners of the one-loop diagram have been represented in a semi-transparent color, since they do not contribute to the value of the parent amplitude. They are just needed to define the system of equations which fix the complex loop momentum k and thus by momentum conservation $\hat{1}$, $\hat{n}$ and $\widehat{Q}_{1,a-1}$. The momentum k corresponds to the complex-component momentum $l(z^*)$ which was to be injected into the external legs of the parent amplitude, in order to apply BCFW recursion. The sub-amplitudes sitting at the upper-corners of the diagram coincide with the sub-amplitudes obtained from factorization via BCFW in the left-hand side of the represented identity.

5.2.3 The BCFW recursion in diagrammatic form

Using the correspondence represented in figure 5.4, the BCFW tree-level recursion:

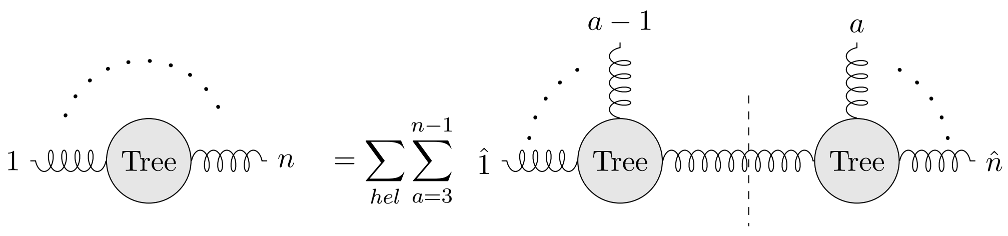


Figure 5.5: Canonical representation of the BCFW recurrence relation.

can be represented in an alternative diagrammatic way as:

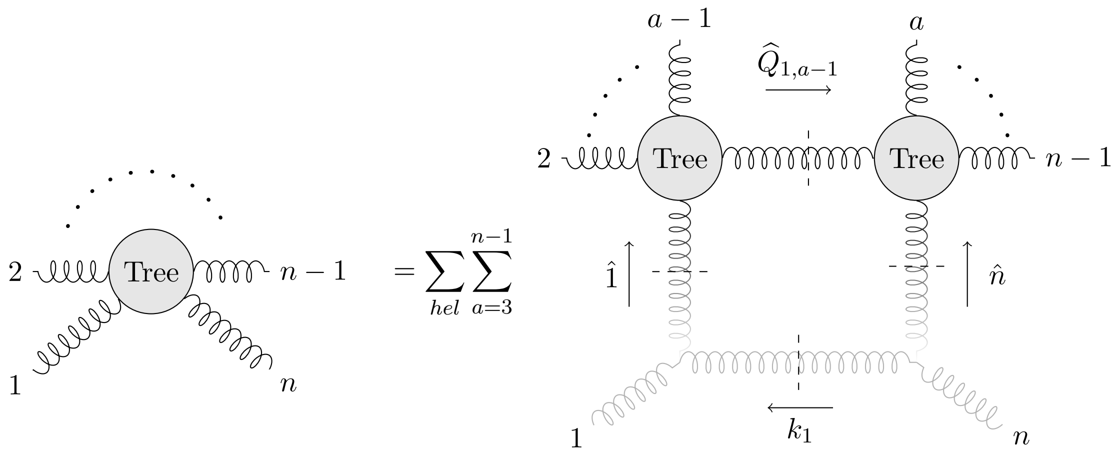
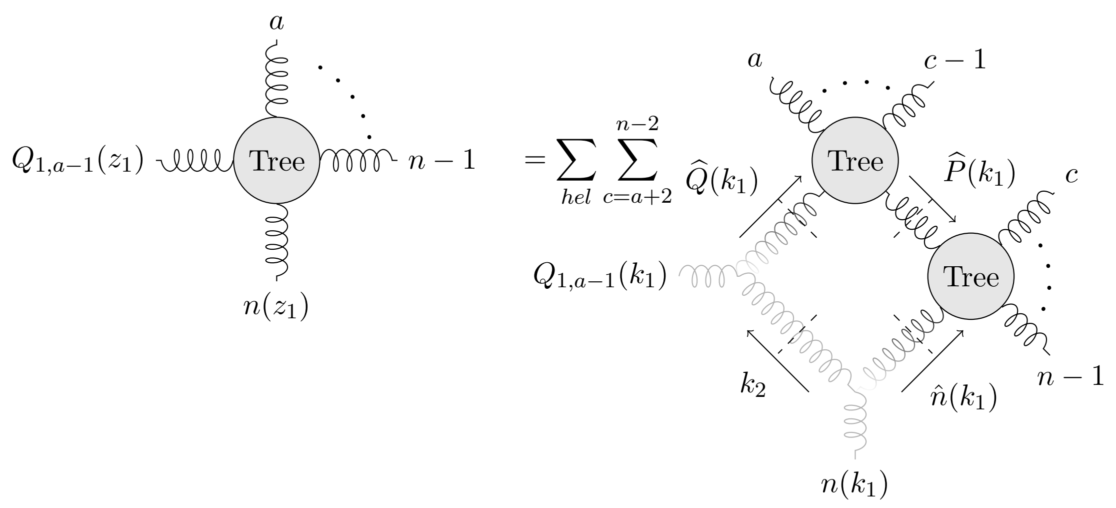


Figure 5.6: Diagrammatic representation of the BCFW recursion in terms of the quadruple-cut of one-loop diagrams.

The sum over *hel* is intended over the two possible helicities of the intermediate state $\widehat{Q}_{1,a-1}$. The sub-amplitudes sitting at the upper-corners of the one-loop diagram are on-shell amplitudes, to which the relation of figure 5.6 can be applied in a recursive manner. We adopt the convention of labelling by k_i the loop momentum appearing at the i-th step of the recursion. This momentum will be carried by the propagator connecting the two legs which were to be shifted in canonical BCFW. For example consider the upper-right corner of one of the addenda in figure 5.6, the successive step of the recursion is given by:



where:

$$\begin{aligned}\widehat{Q}(k_1) &= Q_{1,a-1}(k_1) + k_2\\ \widehat{P}(k_1) &= Q_{1,c-1}(k_1) + k_2\\ \hat{n}(k_1) &= p_n(k_1) + k_2\end{aligned} \tag{5.37}$$

Some of the external legs now depend on k_1, which is the value of the loop momentum computed at the preceding step of the recurrence. k_2 is the new loop momentum. The four cut-conditions, define a system of equations whose solution fixes k_2 and through momentum conservation all the other internal-state momenta. The system to be solved is given by:

$$\begin{cases} k_2^2 = 0\\ \widehat{Q}_{1,a-1}(k_1)^2 = (Q_{1,a-1}(k_1) + k_2)^2 = 0\\ \widehat{Q}_{1,c-1}(k_1)^2 = (Q_{1,c-1}(k_1) + k_2)^2 = 0\\ \hat{p}_n(k_1)^2 = (p_n(k_1) - k_2)^2 = 0\end{cases} \tag{5.38}$$

Using the same strategy as for the solution of eq. (5.27), one writes k_2 in the basis

$$k_2^\mu = x_1 Q_{1,a-1}^\mu(k_1) + x_2 p_n^\mu(k_1) + x_3 \frac{\langle Q_{1,a-1}(k_1)|\sigma^\mu|n(k_1)]}{2} + x_4 \frac{\langle n(k_1)|\sigma^\mu|Q_{1,a-1}(k_1)]}{2} \tag{5.39}$$

Then on-shellness of $\widehat{Q}_{1,a-1}(k_1)$ and $\hat{p}_n(k_1)$ lead to $x_1 = 0 = x_2$, solving $k_2^2 = 0$ gives $x_3 x_4 = 0$ and finally the last condition yields the two possible solutions:

$$k_2 = -\frac{Q_{1,c-1}(k_1)^2}{\langle Q_{1,a-1}(k_1)|Q_{1,c-1}(k_1)|n(k_1)]}\langle Q_{1,a-1}(k_1)|\sigma^\mu|n(k_1)] \tag{5.40}$$

and

$$k_2 = -\frac{Q_{1,c-1}(k_1)^2}{\langle n(k_1)|Q_{1,c-1}(k_1)|Q_{1,a-1}(k_1)]}\langle n(k_1)|\sigma^\mu|Q_{1,a-1}(k_1)] \tag{5.41}$$

These solutions can be obtained directly using the BCFW formalism of section 5.2.1 as $k_2 = l_1(z_1^*)$ and $k_2 = l_2(z_2^*)$.

The n-point amplitude, after the second step of the recursion can be written as:

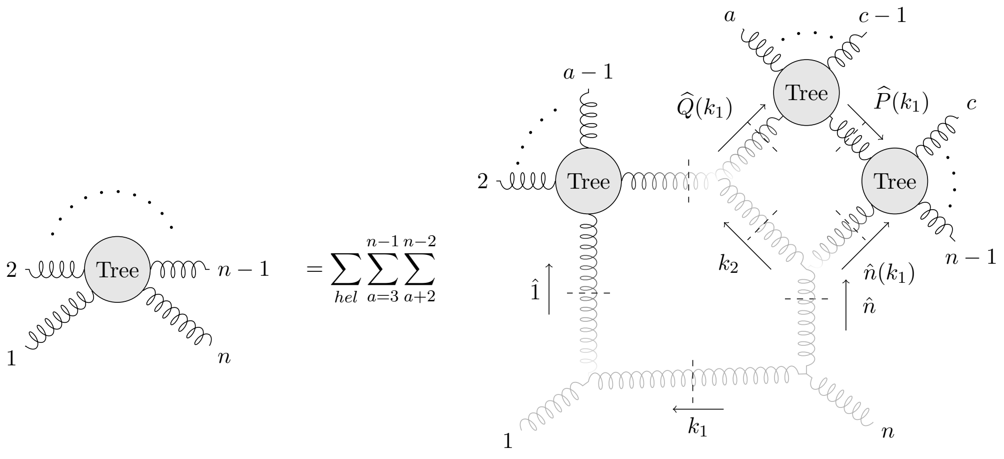


Figure 5.7: Second step of BCFW recursion.

The recursion goes on until only three-point amplitudes appear.

Once a specific complex-momentum was injected into the external particles, i.e. a loop-momentum was computed and fixed, some particles carry a dependence on this complex (loop) momentum. These dependences get more involved the higher the number of external legs of the initial amplitude, since proportionally more steps in the recursion are needed for a complete factorization. We will display explicitly the dependence on previously computed loop-momenta when needed.

Shift of non-adjacent legs

The correspondence of figure 5.4 could be established since we decided to shift the momenta of two adjacent legs, in particular we always stick to the convention of shifting the first and last particles' momenta. However even choosing a different convention and shifting random legs, since one considers color-ordered diagrams, the relation presented stays true modulo reordering of the legs.

5.3 Four-,five-,six-point examples

5.3.1 Conventions

From here on, in order for the figures to be of simpler interpretation, we will adopt the following conventions:

1. we will use straight lines instead of coils for the gluon lines. This leads to no ambiguity since we are considering gluons only.

2. The three-point amplitudes which contribute to the parent amplitude are marked with a black dot.

3. Successive steps in the recursion can be distinguished:

    - in the canonical BCFW factorization by a number above the dashed lines which indicates at which step of the recursion that propagator was put on-shell

- in the multi-loop representation by a number inside the loops which indicates at which step of the recursion that loop appeared

4. All along the recursion one has to sum over the helicity states of internal particles, this sum will be understood since it is always present.
5. In multi-loop representation all the internal propagators are cut, but the cuts are understood and not explicitly depicted. In other words all propagators represent on-shell particles.
6. The dependence of internal momenta on a previously computed loop-momenta will be represented as $p(i, \cdots, j)$, meaning that the momentum p depends on the loop momenta $\{k_i, \cdots, k_j\}$.
7. Finally all external momenta are considered as incoming.

5.3.2 Four-point amplitude

Considering a four point amplitude the BCFW recursion leads to an immediate one-step factorization, which is represented in figure 5.8.

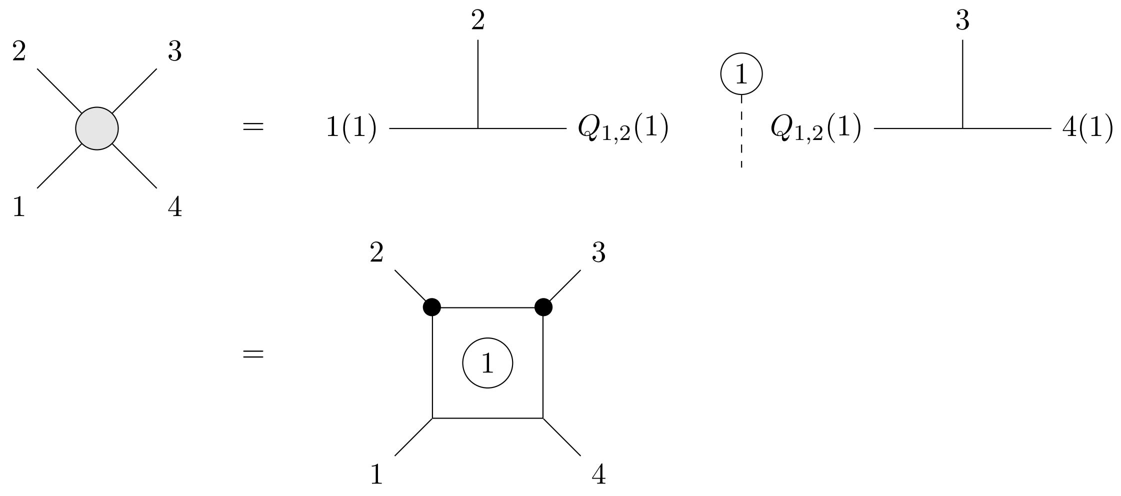


Figure 5.8: Four-point amplitude factorization.

5.3.3 Five-point amplitude

In this section we are going to treat in detail the case of a five-point amplitude. First we will present the factorization in both the canonical BCFW form and via multi-loop diagrams, then we will consider one of the diagrams resulting from the factorization and explicitly solve the two-loop maximum cut system, showing that the constrains as well as solutions it provides are the same as those given by the BCFW recursion.

Factorization

The complete factorization requires two iterations of the recursive relation. These are displayed in figure 5.9 and figure 5.10 respectively.

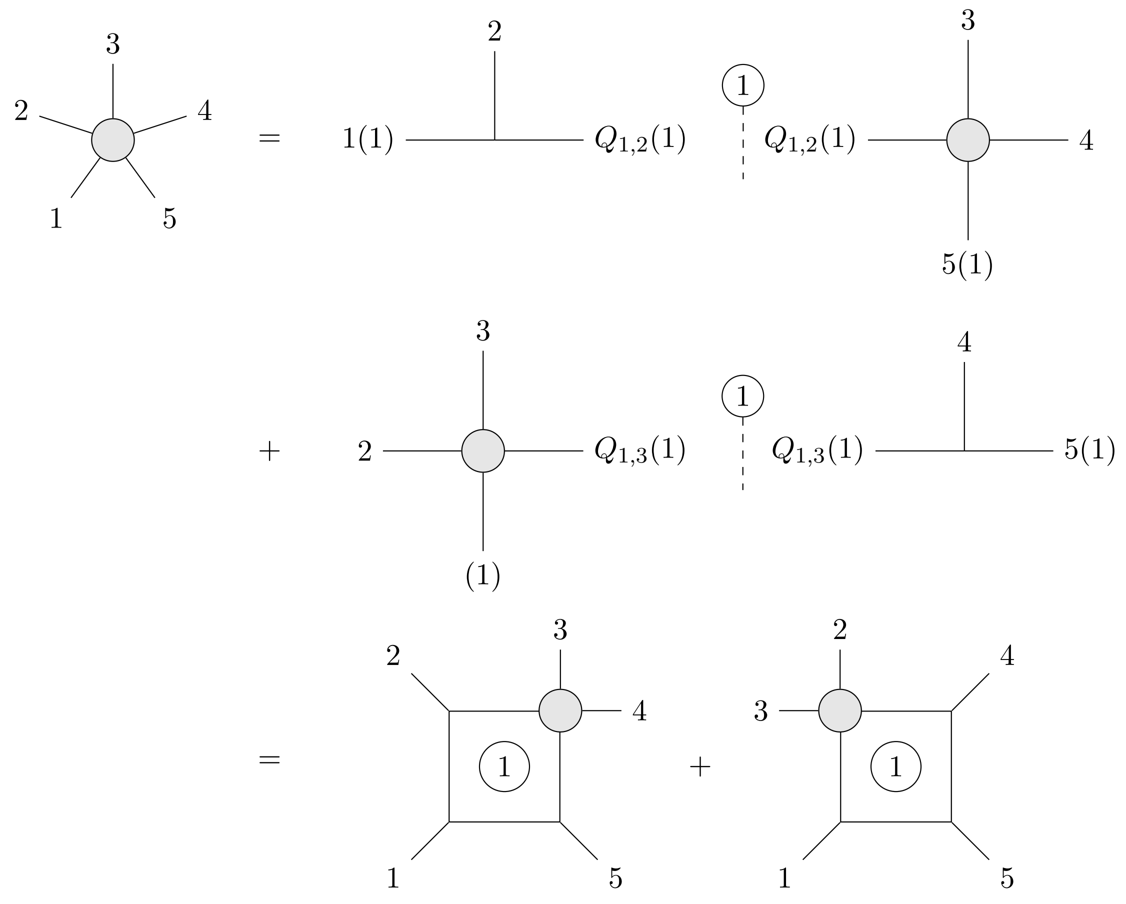


Figure 5.9: First step of BCFW recursion for a five point amplitude.

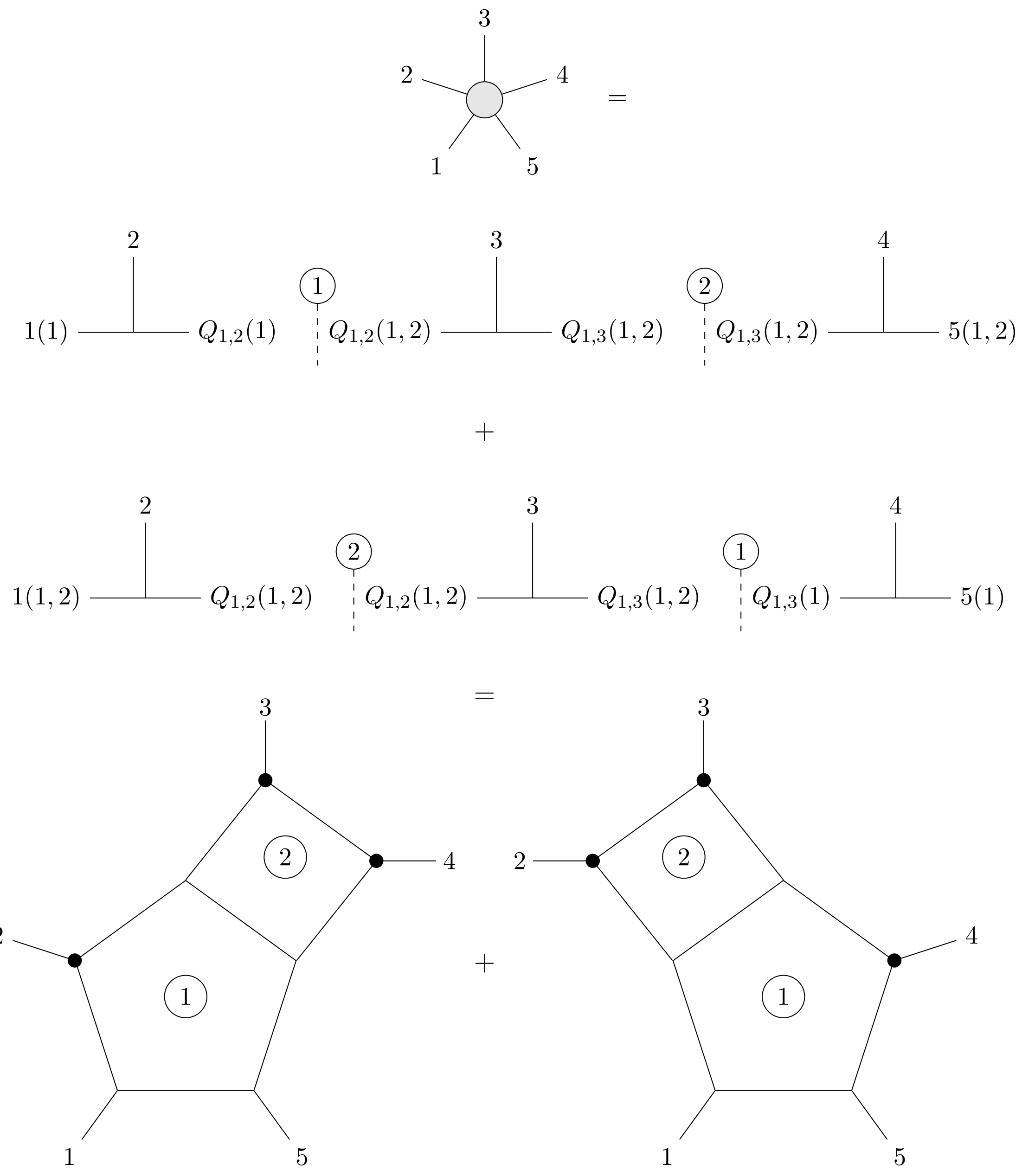


Figure 5.10: Second step of the BCFW factorization for a five point amplitude.

Analytic solution of a two-loop maximum cut system

Let us consider the first diagram resulting from the complete factorization of the five-point amplitude, which is represented in figure 5.11.

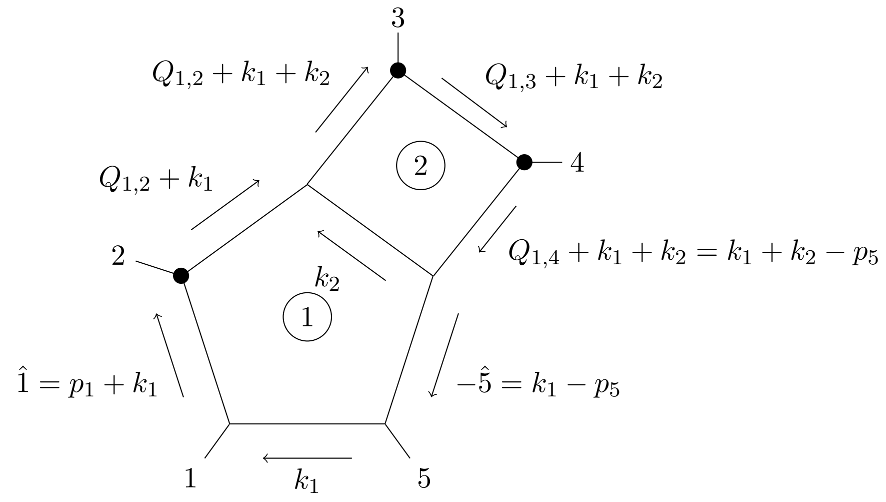


Figure 5.11: One of the terms appearing in the final factorized form of the five-point amplitude.

The 8-ple cut of this diagram corresponds to the following system of eight equations:

$$\begin{cases} k_1^2=0 \\ (k_1+p_1)^2=0 \\ (k_1+p_1+p_2)^2=0 \\ (k_1+k_2+p_1+p_2)^2=0 \\ (k_1+k_2-p_4-p_5)^2=0 \\ (k_1+k_2-p_5)^2=0 \\ k_2^2=0 \\ (k-p_5)^2=0 \end{cases} \tag{5.42}$$

We will proceed in two steps, solving the system first in k_1 and then in k_2. Writing k_1 in the light-cone basis $\{p_1,p_5,\epsilon_{15},\epsilon_{51}\}$ we get

$$k_1^\mu = y_1p_1+y_2p_5+y_3\epsilon_{15}+y_4\epsilon_{51} \tag{5.43}$$

$$\begin{cases} 0=(k_1+p_1)^2=2y_2p_1\cdot p_5 \;\rightarrow\; y_2=0 \\ 0=(k_1-p_5)^2=-2y_1p_1\cdot p_5 \;\rightarrow\; y_1=0 \\ 0=k_1^2=2p_1\cdot p_5(y_1y_2-y_3y_4) \;\rightarrow\; y_3=0\vee y_4=0 \end{cases} \tag{5.44}$$

Using then the condition $0=(k_1+p_1+p_2)^2=2k_1\cdot p_2+2p_1\cdot p_2$ we get

$$\begin{cases} y_1=0 \\ y_2=0 \\ y_3=0\wedge y_4=-\dfrac{p_1\cdot p_2}{p_2\cdot\epsilon_{51}} \end{cases} \tag{5.45}$$

or

$$\begin{cases} y_1=0 \\ y_2=0 \\ y_3=-\dfrac{p_1\cdot p_2}{p_2\cdot\epsilon_{15}}\wedge y_4=0 \end{cases} \tag{5.46}$$

which completely define k_1. Now we can proceed similarly with the momentum k_2. Defining $P \equiv k_1 + p_1 + p_2$ and $\bar{P} \equiv p_5 - k_1$ and

$$k_2^\mu = x_1 P + x_2 \bar{P} + x_3 \epsilon_{P\bar{P}} + x_4 \epsilon_{\bar{P}P} \tag{5.47}$$

$$\begin{cases} 0 = (k_2 + P)^2 = 2x_2 P \cdot \bar{P} \ \rightarrow \ x_2 = 0 \\ 0 = (k_2 - p_4)^2 = -2x_1 P \cdot \bar{P} \ \rightarrow \ x_1 = 0 \\ 0 = k_2^2 = 2P \cdot \bar{P}(x_1 x_2 - x_3 x_4) \ \rightarrow \ x_3 = 0 \vee x_4 = 0 \end{cases} \tag{5.48}$$

And finally using $(k_2 - p_4 - \bar{P})^2 = 0$ we get x_3 and x_4 as functions of P and $\bar{P}$ so implicitly also of k_1:

$$\begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = 0 \wedge x_4 = \dfrac{p_4 \cdot \bar{P}}{p_4 \cdot \epsilon_{\bar{P}P}} \end{cases} \tag{5.49}$$

or

$$\begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = \dfrac{p_4 \cdot \bar{P}}{p_4 \cdot \epsilon_{P\bar{P}}} \wedge x_4 = 0 \end{cases} \tag{5.50}$$

Substituting the two possible values for k_1 computed before inside P and $\tilde{P}$ we get the four possible solutions to the 8-ple cut. We present the solutions writing k_1 explicitly for an easier comparison with the BCFW results which will be given below.

$$\text{For } k_1 = -\frac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{51}} \epsilon_{51}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\dfrac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{51}} \\ 0 \\ 0 \\ 0 \\ \dfrac{2p_4 \cdot (p_5 - k_1)}{\langle p_5 - k_1 | p_4 | k_1 + p_1 + p_2]} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\dfrac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{51}} \\ 0 \\ 0 \\ \dfrac{2p_4 \cdot (p_5 - k_1)}{\langle k_1 + p_1 + p_2 | p_4 | p_5 - k_1]} \\ 0 \end{pmatrix} \right\} \tag{5.51}$$

$$\text{For } k_1 = -\frac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{15}} \epsilon_{15}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \left\{ \begin{pmatrix} 0 \\ 0 \\ -\dfrac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{15}} \\ 0 \\ 0 \\ 0 \\ 0 \\ \dfrac{2p_4 \cdot (p_5 - k_1)}{\langle p_5 - k_1 | p_4 | k_1 + p_1 + p_2]} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ -\dfrac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{15}} \\ 0 \\ 0 \\ 0 \\ \dfrac{2p_4 \cdot (p_5 - k_1)}{\langle k_1 + p_1 + p_2 | p_4 | p_5 - k_1]} \\ 0 \end{pmatrix} \right\} \tag{5.52}$$

BCFW analytic solution

Now we are going to compute the term arising from BCFW recurrence related to the 2-loop pentabox maximum cut given above.

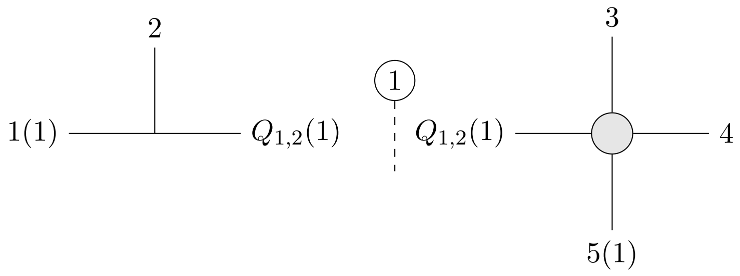


Figure 5.12: First factorization of the term corresponding to figure 5.11.

At the first factorization the term we are interested in has the form shown in figure 5.12. The poles and the associated complex momentum to be injected in the external particles, are computed using the expressions of section 5.2.1. The poles arising from the first factorization are

$$z_{1,1} = -\frac{Q_{12}^2}{\langle 5|Q_{12}|1]} = -\frac{2p_1 \cdot p_2}{\langle 5|p_2|1]} = -\frac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{51}} \tag{5.53}$$

$$z_{1,2} = -\frac{Q_{12}^2}{\langle 1|Q_{12}|5]} = -\frac{2p_1 \cdot p_2}{\langle 1|p_2|5]} = -\frac{p_1 \cdot p_2}{p_2 \cdot \epsilon_{15}} \tag{5.54}$$

(5.55)

where in $z_{i,j}$ the label i refers to the step of the recursion in which the pole is computed and j labels the two distinct possible solutions. The associated complex momenta are

$$l_{1,1} = \frac{z_{1,1}}{2}\langle 5|\sigma|1] \tag{5.56}$$

$$l_{1,2} = \frac{z_{1,2}}{2}\langle 1|\sigma|5] \tag{5.57}$$

leading to

$$p_1(z_1) = p_1 + l_1 \tag{5.58}$$

$$p_5(z_1) = p_5 - l_1 \tag{5.59}$$

$$Q_{12}(z_1) = p_1(z_1) + p_2 = Q_{12} + l_1 \tag{5.60}$$

To compute the second set of poles one has to consider the four-point sub-amplitude factorization represented in figure 5.13.

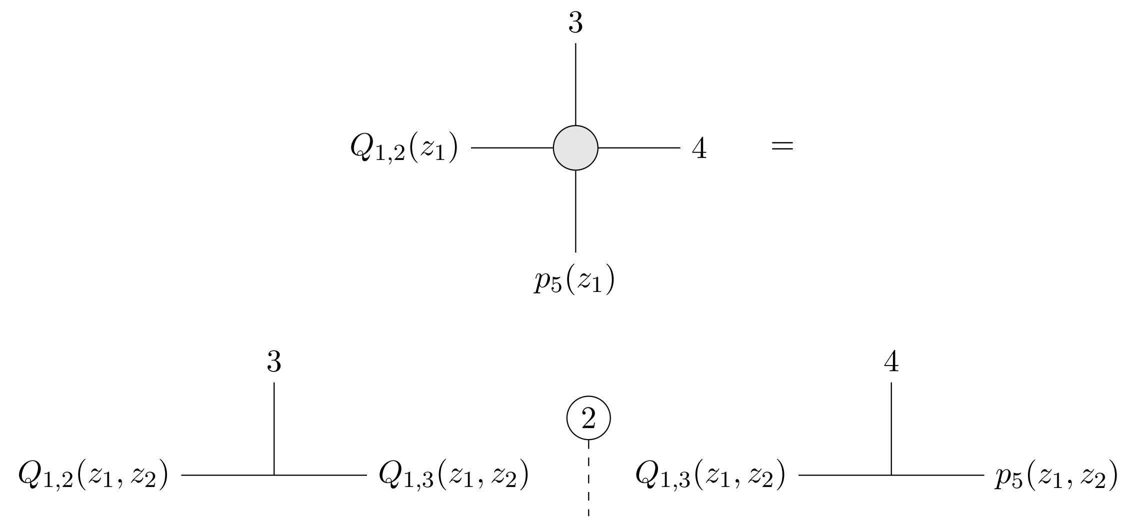


Figure 5.13: Decomposition of the 4-point amplitude resulting from the first factorization shown in figure 5.12.

Notice that the z_2 poles will depend on z_1, so in total there will be four sets of solutions with different poles: $\{\{z_{1,1}, z_{2,1}\}, \{z_{1,1}, z_{2,2}\}, \{z_{1,2}, z_{2,1}\}, \{z_{1,2}, z_{2,2}\}\}$. Applying again the expressions of section 5.2.1 we get:

$$z_{2,1}(z_{1,1}) = -\frac{(Q_{12}(z_{1,1}) + p_3)^2}{\langle Q_{12}(z_{1,1})|(Q_{12}(z_{1,1}) + p_3)|p_5(z_{1,1})]} \tag{5.61}$$

$$= \frac{2p_4 \cdot (p_5 - l_{1,1})}{\langle p_1 + p_2 + l_{1,1}|p_4|p_5 - l_{1,1}]} \tag{5.62}$$

having used momentum conservation $p_1 + p_2 + p_3 = -p_4 - p_5$ and the Dirac equation in momentum space eq. (1.2). And for the second pole

$$z_{2,2}(z_{1,1}) = -\frac{(Q_{12}(z_{1,1}) + p_3)^2}{\langle p_5(z_{1,1})|(Q_{12}(z_{1,1}) + p_3)|Q_{12}(z_{1,1})]} \tag{5.63}$$

$$= \frac{2p_4 \cdot (p_5 - l_{1,1})}{\langle p_5 - l_{1,1}|p_4|p_1 + p_2 + l_{1,1}]} \tag{5.64}$$

Clearly $z_{2,1}(z_{1,2})$ and $z_{2,2}(z_{1,2})$ are given by eq. (5.61) and eq. (5.63) with the substitution $z_{1,1} \to z_{1,2}$. Substituting the explicit form of $l_{1,1}$,$l_{1,2}$ in the BCFW poles and of k_1 in the 8-ple cut solutions we see that they coincide.

5.4 Six-point amplitude

Finally we consider a six-point function, for complete factorization three iterations of the recurrence are needed. The first step of the factorization is depicted in figure 5.14, we get three terms called A,B,C whose decomposition is represented in figure 5.15, figure 5.16 and figure 5.17 respectively. The final result is presented in figure 5.18.

Redundant diagrams

At each step of the recursion one propagator is cut, i.e. the corresponding internal state goes on-shell and gets a complex momentum injection. The value of this complex component depends on the momentum flowing along the propagator, which by momentum conservation depends on the external momenta. The cut propagator produces two new external states for the sub-amplitudes resulting from the factorization, thus successive evaluated cuts depend on previously computed ones. In other words the terms of final factorization may be characterized by the order in which the propagators are cut.

However, the sub-amplitudes produced at each step of the factorization are new on-shell amplitudes which are completely independent. For certain symmetric diagrams then redundant terms may be produced, which arise only because propagators were cut sequentially without considering the independence of the daughter amplitudes. This is the case of the two terms of figure 5.16 of the six-point amplitude. The propagators are cut in the order $\{2, 1, 3\}$ and $\{2, 3, 1\}$, however after the first factorization one gets two independent four-point amplitudes. Cutting first propagator 1 and then 3 or vice versa makes no difference, and the two permutations are equivalent. This can be clearly seen from figure 5.18 where the two equivalent terms have exactly the same form with the loop momenta $k_2 \leftrightarrow k_3$ swapped. The momenta k_i depend only on adjacent loop-momenta, thus $k_2 = k_2(k_1)$ and $k_3 = k_3(k_1)$ and the two terms are the same upon relabelling momenta.

In the next section we will provide some rules which allow to construct the final multi-loop diagrams which contribute to the amplitude directly. One of these will be that diagrams differing only by relabelling of independent loop-momenta are equivalent and contribute only once.

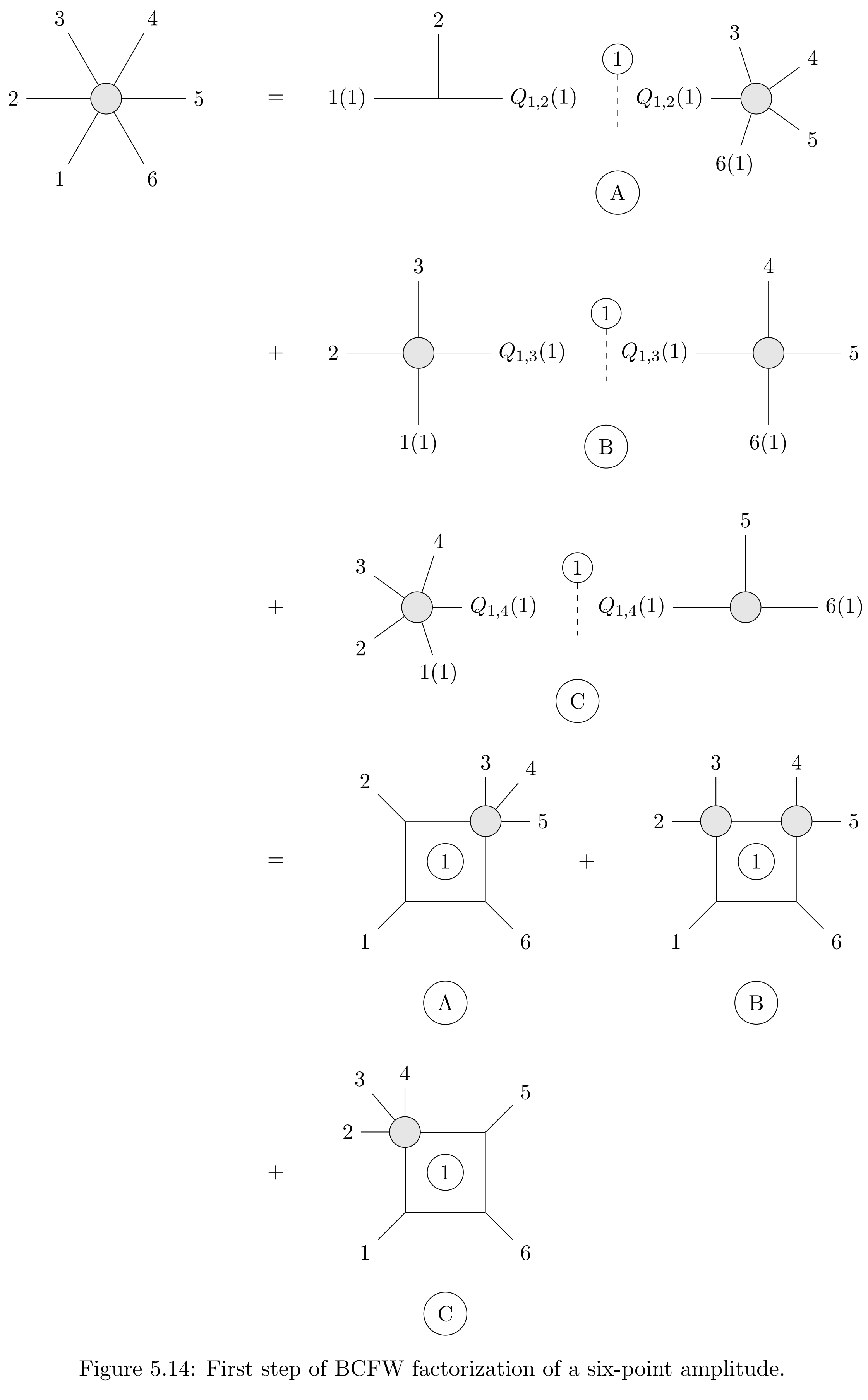


Figure 5.14: First step of BCFW factorization of a six-point amplitude.

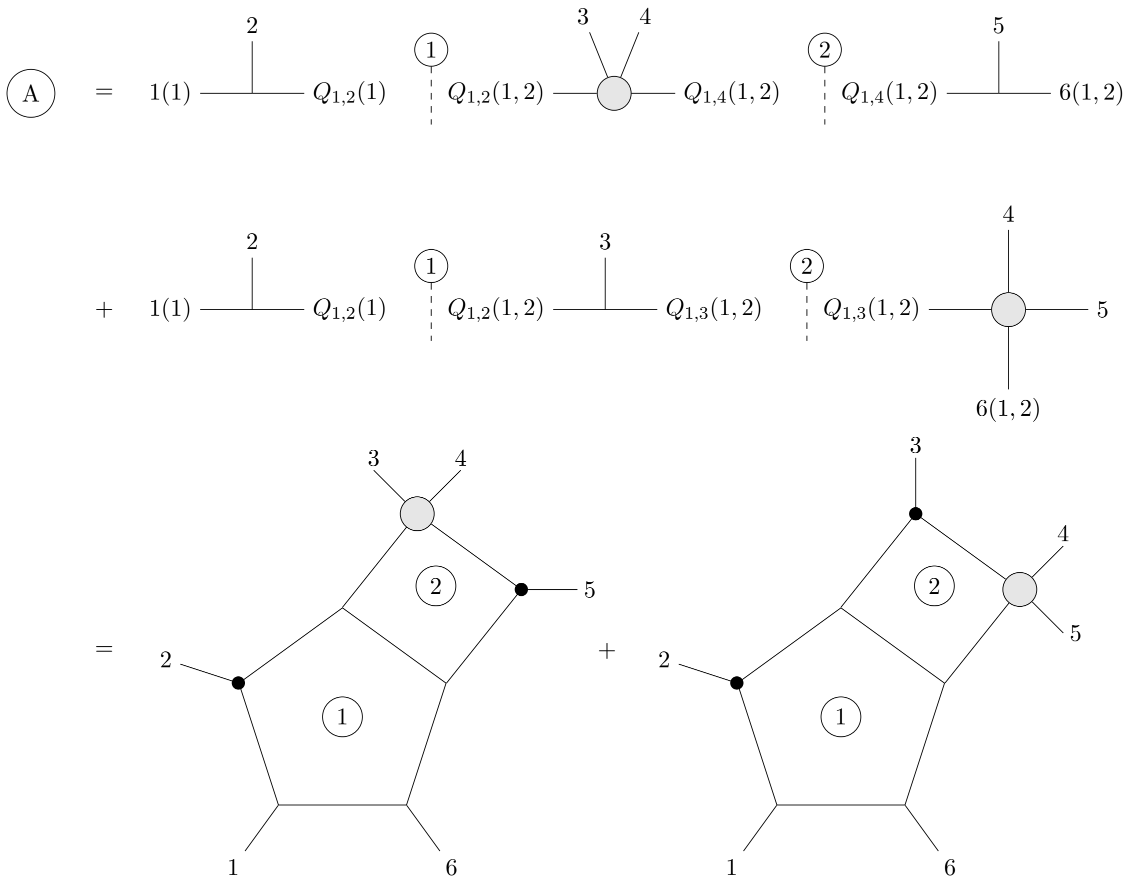


Figure 5.15: Second step of BCFW recursion for the term A resulting from the first factorization.

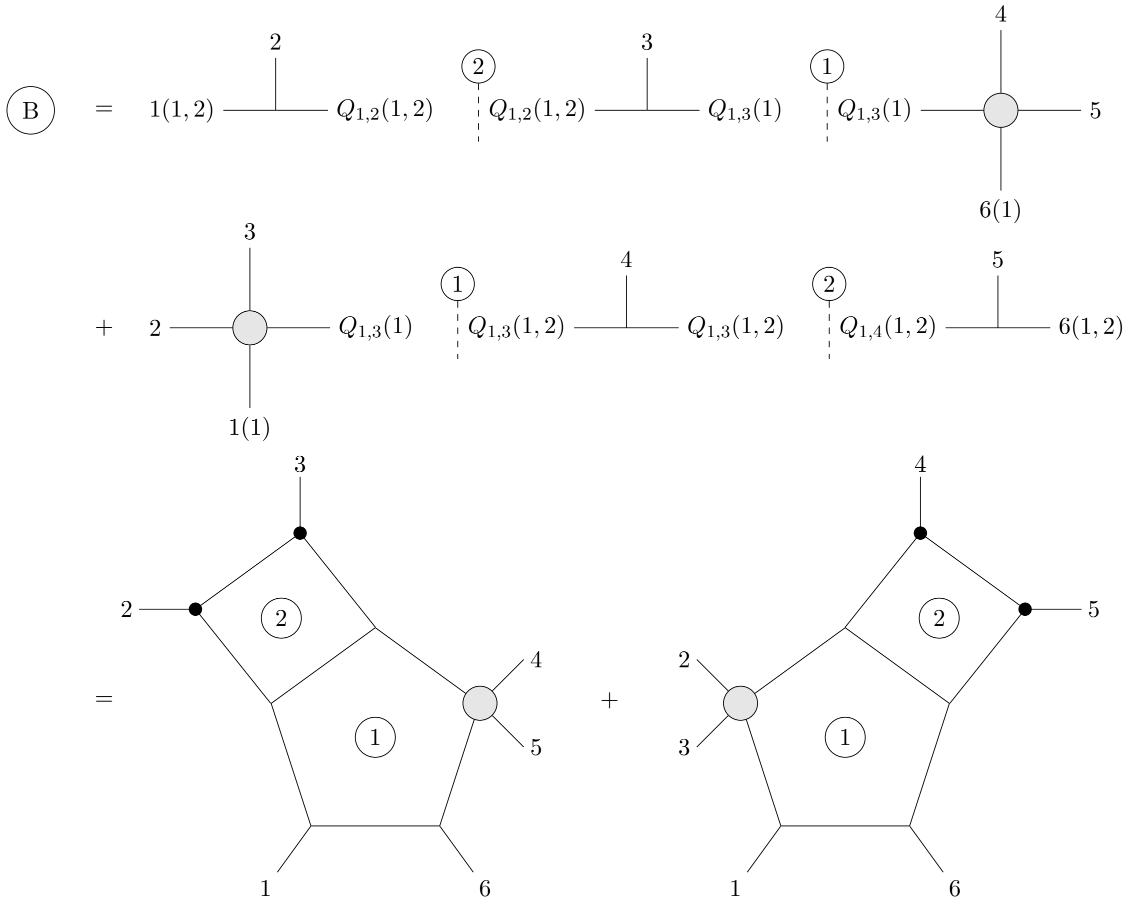


Figure 5.16: Second step of BCFW recursion for the term B resulting from the first factorization.

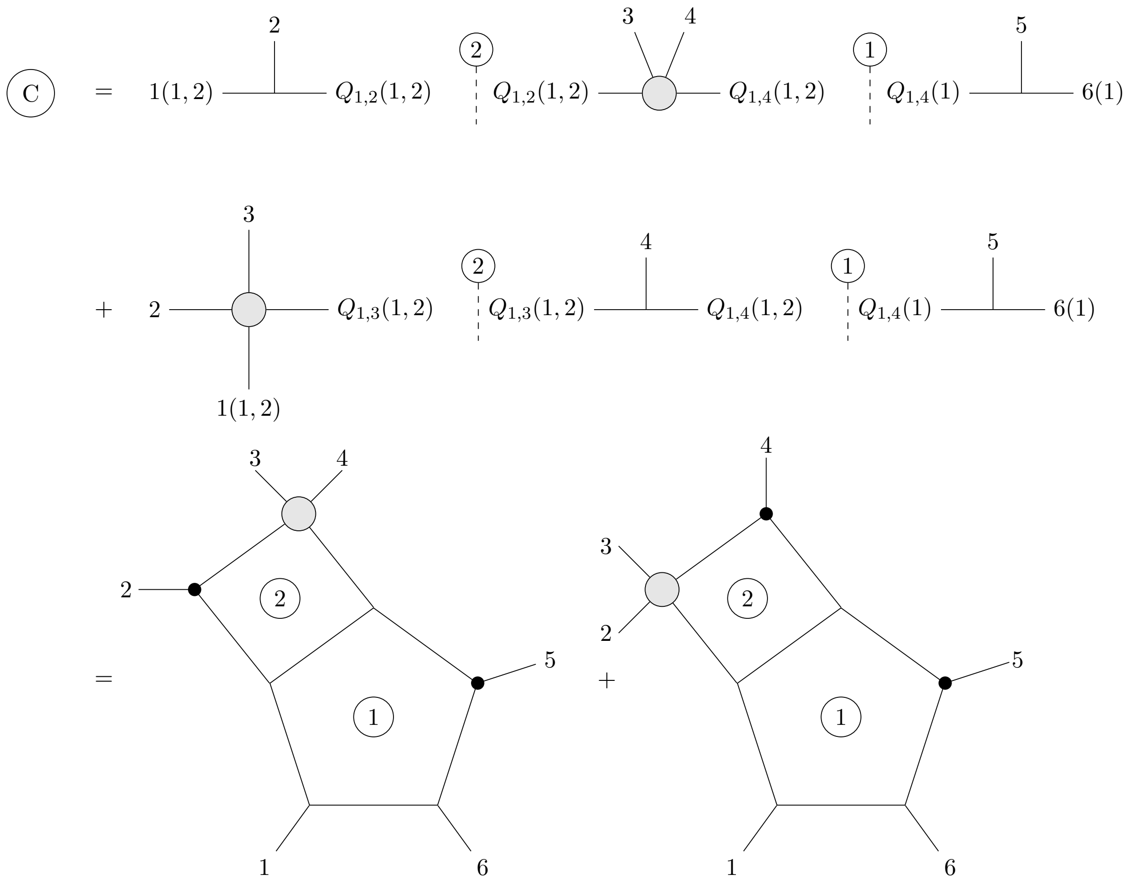


Figure 5.17: Second step of BCFW recursion for the term C resulting from the first factorization.

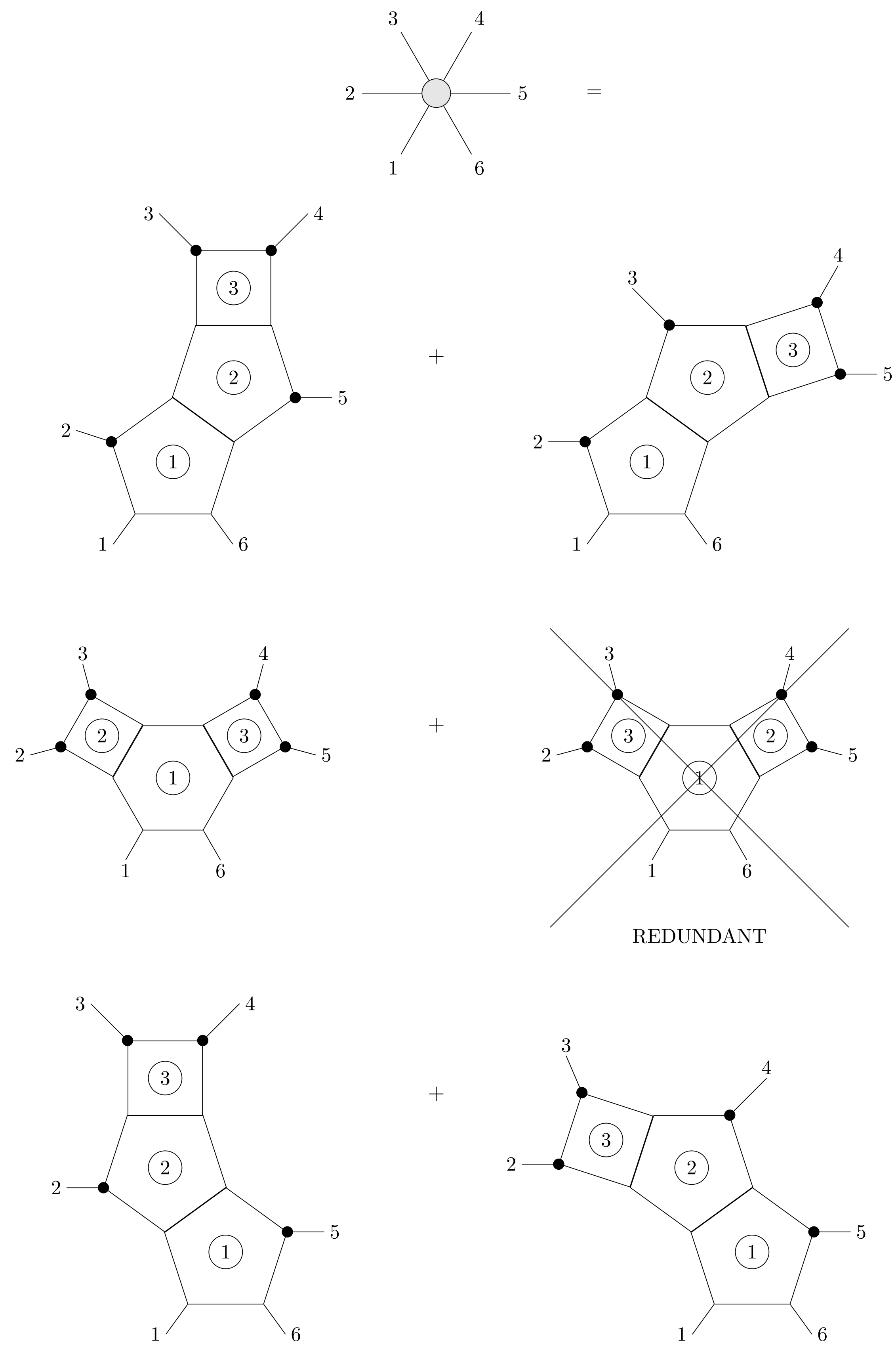


Figure 5.18: Multi-loop representation of the complete factorized six-point amplitude. The crossed term is redundant. It appears only because in this explicit example construction we considered all possible permutations of cut propagators, but only independent permutations contribute.

5.5 Direct construction of the multi-loop representation

The loop diagrams contributing to the BCFW factorization of an n-point amplitude are not all the possible maximum cut graphs with that given number of legs, but a limited subset of these. The following rules allow to build these diagrams in terms of polygons, more in detail squares, pentagons and hexagons, where each edge represents a cut propagator.

The fundamental building block is the square. As seen in section 5.2.3 at each iteration of the BCFW tree-level recursion, the amplitude is substituted by a square with two on-shell sub-amplitude sitting at adjacent corners of it.

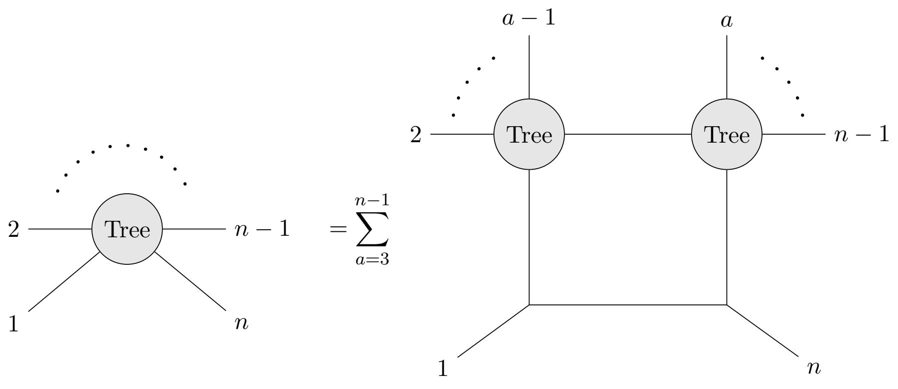


Figure 5.19: BCFW recursion in diagrammatic form.

These sub-amplitudes are then again substituted by new squares, until only three-point functions appear at every corner. Thus at each step of the recursion, four new edges corresponding to the new square appear. This square is nested into an already existing polygon which will thus gain a new edge itself, unless it was the first step where the diagram is the tree-level amplitude. So define

$$I \equiv \text{total iterations needed for complete factorization} \tag{5.65}$$

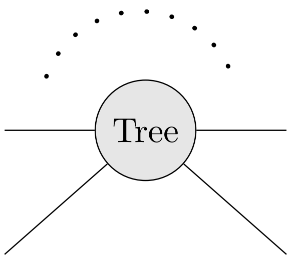
$$N_i \equiv \text{total number of edges in the loop-diagram at } i\text{-th iteration} \tag{5.66}$$

we have that:

$$\begin{aligned} N_I &= 4 + \overbrace{(4+1) + \cdots + (4+1)}^{I-1} \\ &= 4 + 5 \times (I-1) \\ &= 5 \times I - 1 \end{aligned} \tag{5.67}$$

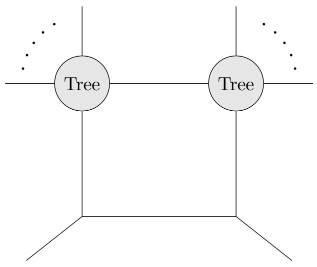
Consider the following as an example, each diagram represented is only one of the possible terms arising from the application of the recursion to the previous step.

Start with the tree-level amplitude:



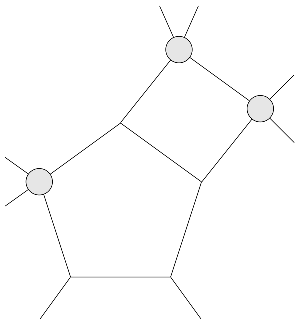
$$\longrightarrow \quad N_0 = 0 \tag{5.68}$$

First application of the recursion in figure 5.19:



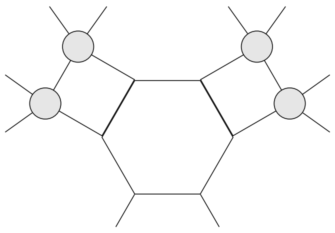
$$\longrightarrow \qquad N_1 = 4 \tag{5.69}$$

Second application:



$$\longrightarrow \qquad N_2 = \underbrace{(4+1)}_{\text{pentagon}} + \overbrace{4}^{\text{new square}} \tag{5.70}$$

third application:



$$\longrightarrow \qquad N_3 = \underbrace{(4+1+1)}_{\text{hexagon}} + \underbrace{4}_{\text{first square}} + \overbrace{4}^{\text{new square}} \tag{5.71}$$

Given an n-point amplitude, the total number of iterations needed for complete factorization is:

$$I = n - 3 \tag{5.72}$$

Clearly since at each iteration a new polygon appears, the total number of them in each diagram must be I.

From the above example it should also be clear that the only figures that may appear are squares, pentagons and hexagons, see figure 5.20.

The building blocks of the multi-loop representation are thus:

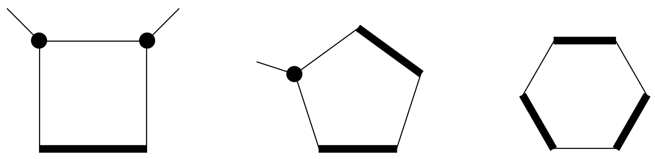
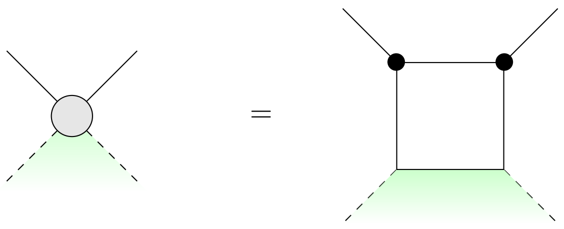
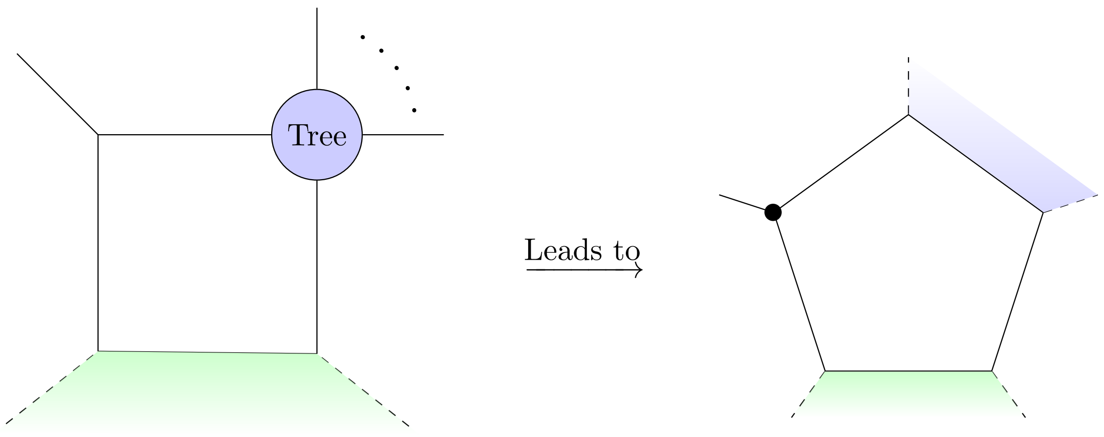


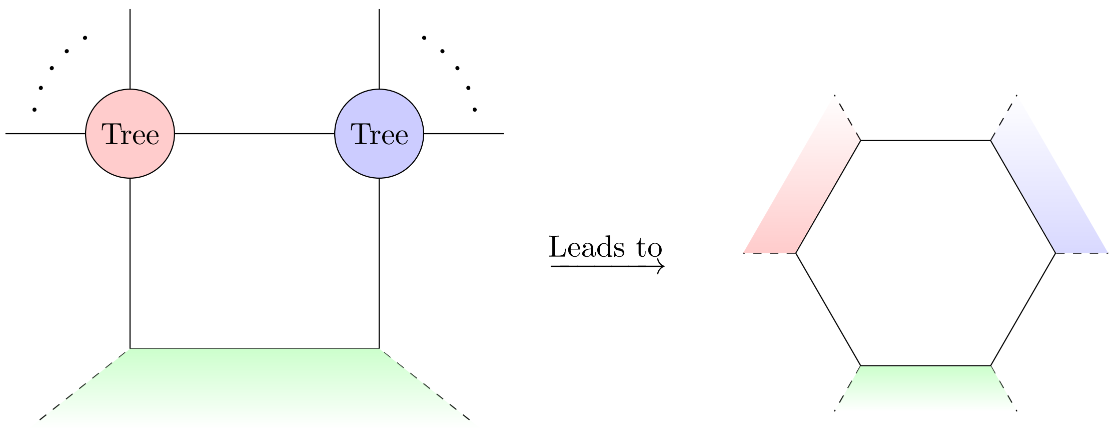
Figure 5.21: Building blocks of the multi-loop representation. The thick edges are shared with other polygons of the diagram.



(a) Square



(b) Pentagon



(c) Hexagon

Figure 5.20: Possible polygons appearing in the multi-loop representation. The shaded areas represent the rest of the diagram to which the given figure is attached.

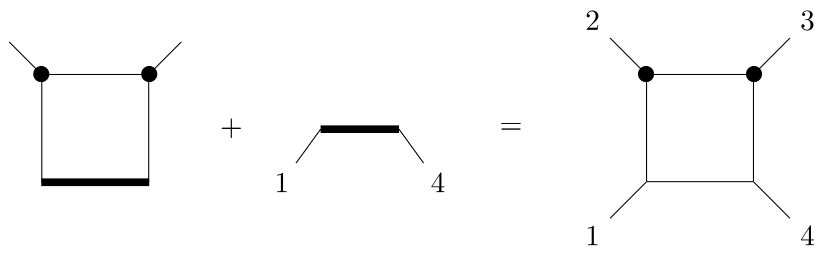


Figure 5.23: The four point amplitude is obtained by all possible combinations (i.e. only one) of one square and the base line.

The thick edges are shared among different polygons. Notice that, by construction, in the final diagrams a square is always attached to two external lines, a pentagon to one and a hexagon to none. There is however one exception: the polygon which originates from the first application of BCFW is attached to the first and last external legs, independently of whether it is a square, a pentagon or a hexagon.[3] We then define an object called "base line" as in figure 5.22, which makes up for this exception. This object saturates one of the shared edges of the polygons.

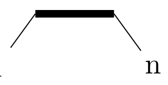


Figure 5.22: "Base line" which saturates one shared edge of a polygon.

For example given a four-point amplitude the only figure that can appear in the multi-loop diagram is one square. This has two external legs attached to it and one edge which should be shared with another polygon. This edge is saturated by the base line. The four-point amplitude is obtained by matching the square with the base line in all possible combinations, as represented in figure 5.23.

Finally notice that any application of the BCFW recursion ends inevitably with the factorization of a four-point amplitude into two three-point sub-amplitudes. This means that each multi-loop diagram must contain at least one square, which corresponds to this factorization.

Drawing recipe

We have that the maximum-cut graphs contributing to an n-point tree-level amplitude can be obtained by the following rules:

1. Compute the number of iterations needed for complete factorization: $I = n - 3$

[3]These two external legs are the only external lines attached to three-point amplitudes which do not contribute to the value of the parent amplitude.

2. Build sets of I polygons among those represented in figure 5.21 that satisfy $N = 5I - 1$, with N sum of all the edges of the figures in the set. Each set must contain at least one square.

3. For each set combine the polygons among them and with the base line in all possible ways, considering that:

   - The polygons are to be connected to one another only on the thick lines of figure 5.21
   - Polygons with the same number of edges are to be considered indistinguishable
   - Only connected diagrams are admitted

The resulting diagrams are all the possible maximum-cut graphs corresponding to the BCFW tree-level factorization.

Example:five-point amplitude

Consider a five-point amplitude. We have that:

$$\begin{aligned} n &= 5 \\ I &= 2 \\ N &= 9 \end{aligned} \tag{5.73}$$

So now we need to build all the possible sets of $I = 2$ polygons among squares, pentagons and hexagons whose total edges sum to $N = 9$. The only possibility is

$$9 = 4 + 5 \tag{5.74}$$

Thus take the set:

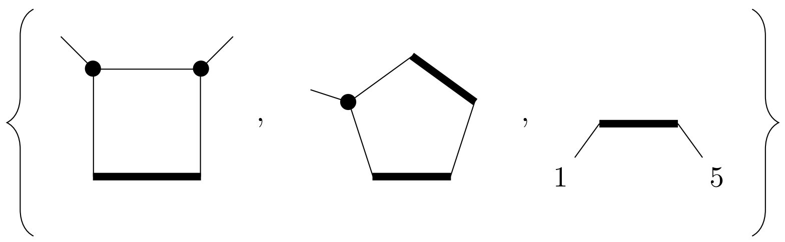


Figure 5.24: Building blocks of the five-point amplitude.

and take all possible combinations. The result is depicted in figure 5.25.

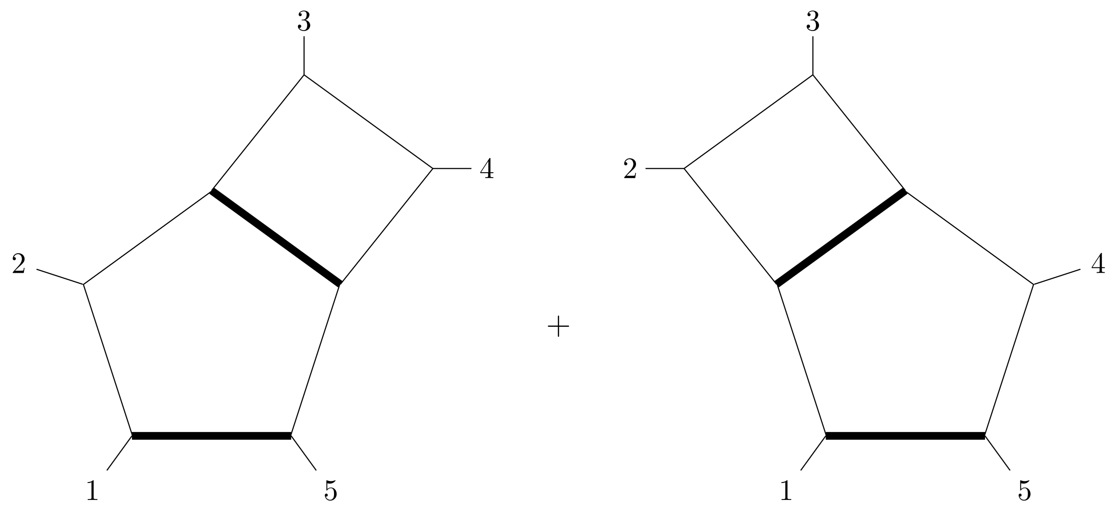


Figure 5.25: Five point amplitude multi-loop representation.

Example: seven-point amplitude

Consider a seven-point amplitude. We have that:

$$\begin{aligned} n &= 7 \\ I &= 4 \\ N &= 19 \end{aligned} \tag{5.75}$$

So the contributing diagrams are given by a combination of four polygons among squares, pentagons and hexagons, with in total 19 edges plus the base line. Since at least one square must be present it is easiest to build the sets in terms of increasing number of squares. We get that the only allowed sets of figures are:

$$\begin{aligned} 19 &= 4 + 5 + 5 + 5 \\ 19 &= 4 + 4 + 5 + 6 \end{aligned} \tag{5.76}$$

The two possible sets are then:

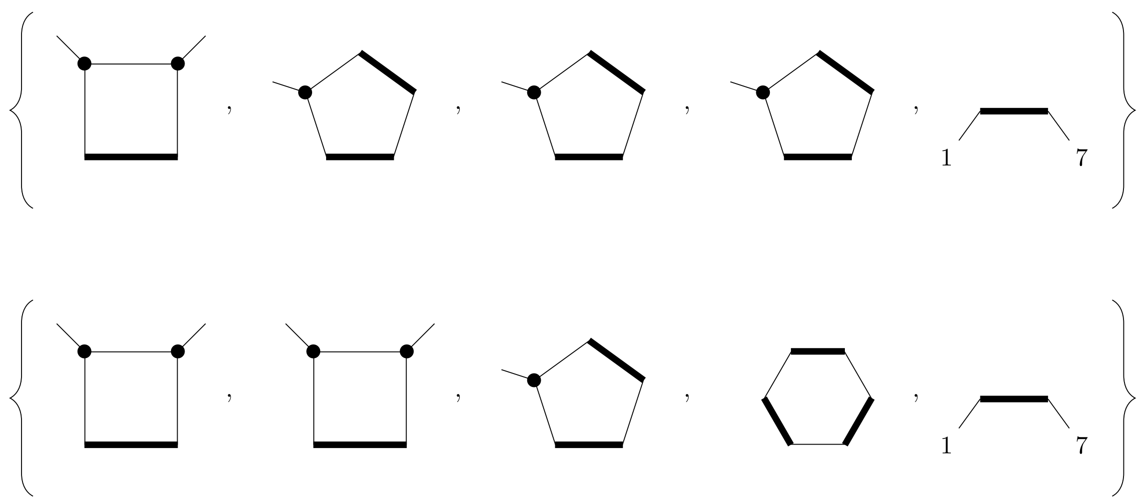


Figure 5.26: The two allowed sets of building blocks for the seven-point amplitude.

These figures will then be combined in all the possible ways among them, obtaining the diagrams shown in figure 5.27 and figure 5.28. The two legs at the bottom of each diagram are the lines 1 and 7 which form the base line.

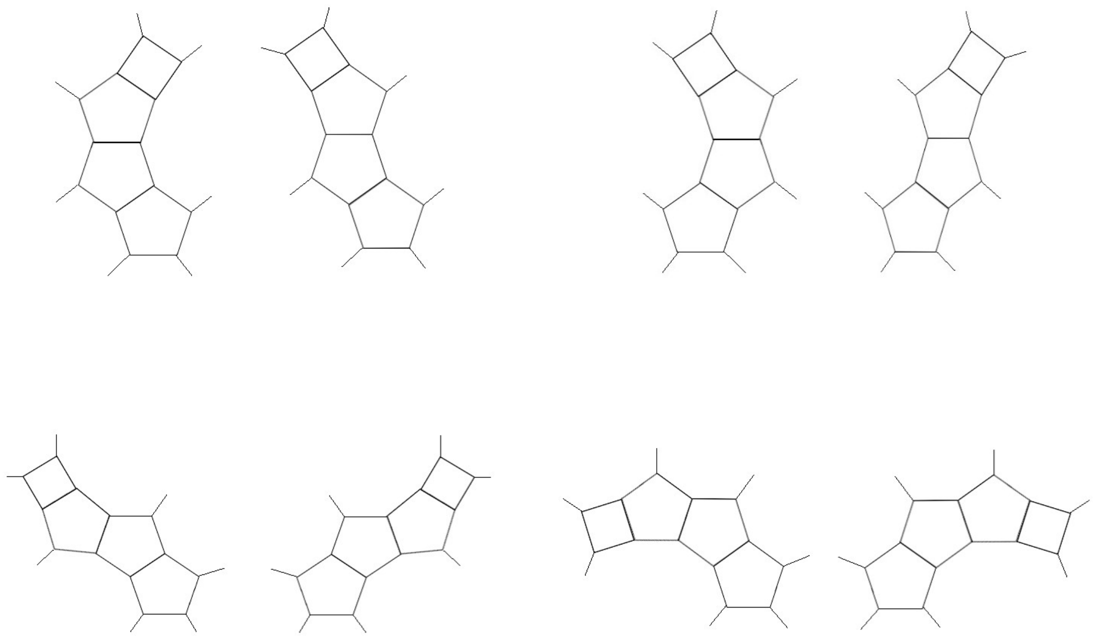


Figure 5.27: Seven-point amplitude complete factorization in multi-loop representation. Possible combinations with one square and three pentagons.

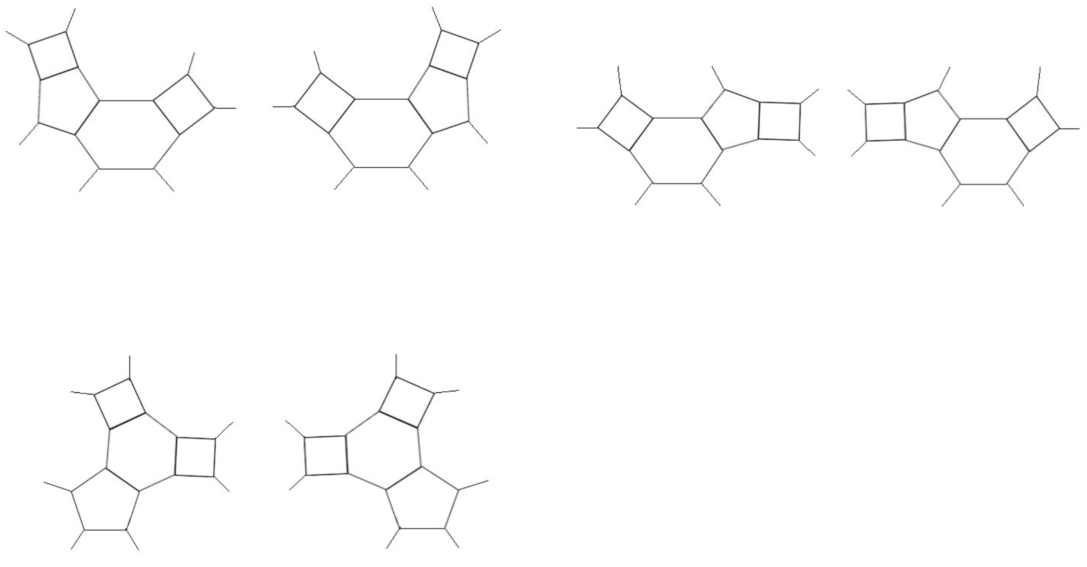


Figure 5.28: Seven-point amplitude complete factorization in multi-loop representation. Possible combinations with two squares, one pentagon and one hexagon.

Conclusions

In this work we presented in detail all the techniques needed to perform the functional reconstruction of univariate rational functions over finite fields, and discussed its application to tree-level scattering amplitudes.

We started by introducing the rational reconstruction techniques on the field $\mathbb{Q}$, which is the most natural setting for them to be applied. Since however it is not always the most convenient one, finite fields were introduced along with their fundamental operations. These were defined mathematically as well as operatively, always presenting an immediate possible code implementation. Particular attention has been given to the problem of defining a multiplicative inverse not only on finite fields but also on rings, which proves essential when applying the Chinese remainder theorem at the last stage of the reconstruction procedure. The task of inverting the mapping from $\mathbb{Q}$ to $\mathbb{Z}_p$ was addressed, showing how it is possible to obtain a unique inverse image under appropriate hypothesis, which involve the size of the order p of the finite field. Then we showed how this hypothesis could be fulfilled using the Chinese remainder theorem, allowing us not to leave the domain of machine-size numbers for the numerical calculations.

Then we discussed numerical evaluation of tree-level amplitudes on finite fields. We showed how the combined use of spinor-helicity formalism and momentum-twistor variables allows to write scattering amplitudes as rational functions. The Berends-Giele and BCFW recursions were presented, along with appropriate factorizations of them in terms of a completely rational part and an overall factor carrying imaginary and irrational dependencies.

The first main result of this work was then obtained, i.e. the rational reconstruction on finite fields of a tree-level scattering amplitude using the BCFW recursion as black-box algorithm, which was presented here for the first time.

Finally some relations among BCFW tree-level recurrence and multi-loop maximum-cut graphs were explored. We presented a novel strategy which allows the direct identification/construction of all the multi-loop graphs related to BCFW recursion, starting only from the number of external legs. The multi-loop representation, a part from providing a possible direct construction of the completely factorized amplitude usually obtained by the application of the BCFW recursion, also allows to recognize otherwise hidden relations among addenda appearing in it. Furthermore its relation with the residue of the maximum-cut appearing in the integrand decomposition method is still to be studied, in particular the possible connection with the constant term of this polynomial. It is our hope that a deeper mathematical structure, apparently hidden behind these considerations, could one day be uncovered.

Appendix A

Groups, rings and fields

We briefly recall some definitions about groups, rings and fields, see for example [16].

Definition A.1 (Group)**.** Given a set G is said to be a group if it is endowed with an operation $\circ$ such that

1. $\circ$ is associative, $a \circ (b \circ c) = (a \circ b) \circ c \;\; \forall\, a, b, c \in G$
2. there is an identity element e such that $a \circ e = e \circ a = a \;\; \forall\, a \in G$
3. $\forall a \in G$ there is an inverse element denoted a^{-1} such that $a \circ a^{-1} = a^{-1} \circ a = e$

A group is said to be *abelian* if $\circ$ is commutative.

Definition A.2 (Ring)**.** A ring $(R, +, \cdot)$ is a set R together with two binary operations $+$ called addition and $\cdot$ called multiplication, such that:

1. R is an abelian group with respect to $+$
2. $\cdot$ is associative, $a \cdot (b \cdot c) = (a \cdot b) \cdot c \;\; \forall\, a, b, c \in R$
3. the distributive law holds: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(b + c) \cdot a = (b \cdot a) + (c \cdot a) \;\; \forall\, a, b, c \in R$

A ring is said to be a ring with identity if $\cdot$ admits an identity element, which we will denote by 1. The identity element of the operation $+$ will be denoted 0. Furthermore a ring is called commutative if $\cdot$ is commutative.

Definition A.3 (Field)**.** A commutative ring whose *non-zero* elements form a group under $\cdot$ is called a field

A field composed of a finite number of elements is said to be finite.

Another definition which we will need later is that of group homomorphism:

Definition A.4 (Group homomorphism)**.** A map $F : G \to H$ is said to be an homomorphism between the groups G and H if it preserves the group operation of G:

$$F(a \circ_G b) = F(a) \circ_H F(b) \qquad \forall\, a, b \in G$$

where $\circ_G$ and $\circ_H$ are the group operations of G and H respectively.

In a similar fashion for rings, and thus fields, we have the extension of the above to

Definition A.5 (Ring homomorphism)**.** A map $F : R_1 \to R_2$ is said to be an homomorphism between the rings (R_1, σ_1, ϕ_1) and (R_2, σ_2, ϕ_2) if

$$F(\sigma_1(a, b)) = \sigma_2(F(a), F(b)) \quad \forall\, a, b \in G$$

and

$$F(\phi_1(a, b)) = \phi_2(F(a), F(b)) \quad \forall\, a, b \in G$$

where σ_i and ϕ_i denote the addition and multiplication operation on R_i.

In other words the ring isomorphism preserves both, addition and multiplication. We will not use different names to refer to these operations on different rings, since this distinction should be clear from the context. Thus if F is an homomorphism we will simply write

$$F(a + b) = F(a) + F(b) \qquad F(a \cdot b) = F(a) \cdot F(b)$$

A bijective homomorphism is called *isomorphism.*

Bibliography

[1] V.E. Asribekov. "Choice of invariant variables and analytical properties of many-point functions". In: *Physics Letters* 2.6 (1962), pp. 284–286. ISSN: 0031-9163. DOI: https://doi.org/10.1016/0031-9163(62)90040-9. URL: http://www.sciencedirect.com/science/article/pii/0031916362900409.

[2] Simon Badger. "Automating QCD amplitudes with on-shell methods". In: *J. Phys. Conf. Ser.* 762.1 (2016), p. 012057. DOI: 10.1088/1742-6596/762/1/012057. arXiv: 1605.02172 [hep-ph].

[3] Simon Badger, Hjalte Frellesvig, and Yang Zhang. "A Two-Loop Five-Gluon Helicity Amplitude in QCD". In: *JHEP* 12 (2013), p. 045. DOI: 10.1007/JHEP12(2013)045. arXiv: 1310.1051 [hep-ph].

[4] Frits A. Berends and W. T. Giele. "Recursive Calculations for Processes with n Gluons". In: *Nucl. Phys.* B306 (1988), pp. 759–808. DOI: 10.1016/0550-3213(88)90442-7.

[5] Zvi Bern, Lance J. Dixon, and David A. Kosower. "On-Shell Methods in Perturbative QCD". In: *Annals Phys.* 322 (2007), pp. 1587–1634. DOI: 10.1016/j.aop.2007.04.014. arXiv: 0704.2798 [hep-ph].

[6] Zvi Bern et al. "One loop n point gauge theory amplitudes, unitarity and collinear limits". In: *Nucl. Phys.* B425 (1994), pp. 217–260. DOI: 10.1016/0550-3213(94)90179-1. arXiv: hep-ph/9403226 [hep-ph].

[7] Ruth Britto, Freddy Cachazo, and Bo Feng. "Generalized unitarity and one-loop amplitudes in N=4 super-Yang-Mills". In: *Nucl. Phys.* B725 (2005), pp. 275–305. DOI: 10.1016/j.nuclphysb.2005.07.014. arXiv: hep-th/0412103 [hep-th].

[8] Ruth Britto, Bo Feng, and Pierpaolo Mastrolia. "The Cut-constructible part of QCD amplitudes". In: *Phys. Rev.* D73 (2006), p. 105004. DOI: 10.1103/PhysRevD.73.105004. arXiv: hep-ph/0602178 [hep-ph].

[9] Ruth Britto et al. "Direct proof of tree-level recursion relation in Yang-Mills theory". In: *Phys. Rev. Lett.* 94 (2005), p. 181602. DOI: 10.1103/PhysRevLett.94.181602. arXiv: hep-th/0501052 [hep-th].

[10] Bruno Buchberger. "An Algorithmic Criterion for the Solvability of an Algebraic System of Equations". In: (Jan. 1998).

[11] Freddy Cachazo, Peter Svrcek, and Edward Witten. "MHV vertices and tree amplitudes in gauge theory". In: *JHEP* 09 (2004), p. 006. DOI: 10.1088/1126-6708/2004/09/006. arXiv: hep-th/0403047 [hep-th].

[12] Annie Cuyt and Wen-shin Lee. "Sparse interpolation of multivariate rational functions". In: *Theoretical Computer Science* 412.16 (2011). Symbolic and Numerical Algorithms, pp. 1445–1456. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.11.050. URL: http://www.sciencedirect.com/science/article/pii/S0304397510006882.

[13] Lance J. Dixon. "Calculating scattering amplitudes efficiently". In: *QCD and beyond. Proceedings, Theoretical Advanced Study Institute in Elementary Particle Physics, TASI-95, Boulder, USA, June 4-30, 1995.* 1996, pp. 539–584. arXiv: hep-ph/9601359 [hep-ph]. URL: http://www-public.slac.stanford.edu/sciDoc/docMeta.aspx?slacPubNumber=SLAC-PUB-7106.

[14] Henriette Elvang and Yu-tin Huang. "Scattering Amplitudes". In: (2013). arXiv: 1308.1697 [hep-th].

[15] Andrew Hodges. "Eliminating spurious poles from gauge-theoretic amplitudes". In: *JHEP* 05 (2013), p. 135. DOI: 10.1007/JHEP05(2013)135. arXiv: 0905.1473 [hep-th].

[16] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications.* Cambridge university press, 1986.

[17] D. Maitre and P. Mastrolia. "S@M, a Mathematica Implementation of the Spinor-Helicity Formalism". In: *Comput. Phys. Commun.* 179 (2008), pp. 501–574. DOI: 10.1016/j.cpc.2008.05.002. arXiv: 0710.5559 [hep-ph].

[18] Pierpaolo Mastrolia and Giovanni Ossola. "On the Integrand-Reduction Method for Two-Loop Scattering Amplitudes". In: *JHEP* 11 (2011), p. 014. DOI: 10.1007/JHEP11(2011)014. arXiv: 1107.6041 [hep-ph].

[19] Pierpaolo Mastrolia, Tiziano Peraro, and Amedeo Primo. "Adaptive Integrand Decomposition in parallel and orthogonal space". In: *JHEP* 08 (2016), p. 164. DOI: 10.1007/JHEP08(2016)164. arXiv: 1605.03157 [hep-ph].

[20] Pierpaolo Mastrolia et al. "Integrand-Reduction for Two-Loop Scattering Amplitudes through Multivariate Polynomial Division". In: *Phys. Rev.* D87.8 (2013), p. 085026. DOI: 10.1103/PhysRevD.87.085026. arXiv: 1209.4319 [hep-ph].

[21] Pierpaolo Mastrolia et al. "Scattering Amplitudes from Multivariate Polynomial Division". In: *Phys. Lett.* B718 (2012), pp. 173–177. DOI: 10.1016/j.physletb.2012.09.053. arXiv: 1205.7087 [hep-ph].

[22] Donal O'Shea, David Cox, and John Little. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* 2nd ed. Springer, 1997.

[23] Giovanni Ossola, Costas G. Papadopoulos, and Roberto Pittau. "Reducing full one-loop amplitudes to scalar integrals at the integrand level". In: *Nucl. Phys.* B763 (2007), pp. 147–169. DOI: 10.1016/j.nuclphysb.2006.11.012. arXiv: hep-ph/0609007 [hep-ph].

[24] Stephen J. Parke and T. R. Taylor. "An Amplitude for n Gluon Scattering". In: *Phys. Rev. Lett.* 56 (1986), p. 2459. DOI: 10.1103/PhysRevLett.56.2459.

[25] Tiziano Peraro. "Scattering amplitudes over finite fields and multivariate functional reconstruction". In: *JHEP* 12 (2016), p. 030. DOI: 10.1007/JHEP12(2016)030. arXiv: 1608.01902 [hep-ph].

[26] Tiziano Peraro. "Unpublished notes". In: (2015).

[27] Michael E. Peskin. “Simplifying Multi-Jet QCD Computation”. In: *Proceedings, 13th Mexican School of Particles and Fields (MSPF 2008): San Carlos, Sonora, Mexico, October 2-11, 2008.* 2011. arXiv: 1101.2414 [hep-ph]. URL: http://www-public.slac.stanford.edu/sciDoc/docMeta.aspx?slacPubNumber=SLAC-PUB-14352.

[28] Kasper Risager. “A Direct proof of the CSW rules”. In: *JHEP* 12 (2005), p. 003. DOI: 10.1088/1126-6708/2005/12/003. arXiv: hep-th/0508206 [hep-th].

[29] Mark Srednicki. *Quantum field theory.* Cambridge university press, 2007.

[30] Paul Wang. “A p-adic algorithm for univariate partial fractions”. In: (Jan. 1981), pp. 212–217.

[31] Paul S. Wang, M. J. T. Guy, and J. H. Davenport. “P-adic Reconstruction of Rational Numbers”. In: *SIGSAM Bull.* 16.2 (May 1982), pp. 2–3. ISSN: 0163-5824. DOI: 10.1145/1089292.1089293. URL: http://doi.acm.org/10.1145/1089292.1089293.

[32] Stefan Weinzierl. “Tales of 1001 Gluons”. In: *Phys. Rept.* 676 (2017), pp. 1–101. DOI: 10.1016/j.physrep.2017.01.004. arXiv: 1610.05318 [hep-th].