

Technische Universität München

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)

Advanced methods for scattering amplitudes in gauge theories

Tiziano Peraro

Vollständiger Abdruck der von der Fakultät für Physik
der Technischen Universität München zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigten Dissertation

Vorsitzender: Univ.-Prof. Dr. S. Schönert
Prüfer der Dissertation: 1. Hon.-Prof. Dr. W. F. L. Hollik
2. Univ.-Prof. Dr. M. Beneke

Die Dissertation wurde am 16. Juni 2014
bei der Technische Universität München eingereicht und
durch die Fakultät für Physik am 24. September 2014 angenommen.

Abstract

In this thesis, we present new techniques for the integrand reduction of scattering amplitudes at one and higher loops and their applications to gauge theories such as QCD and the Standard Model. The goal of integrand-reduction methods is the computation of loop integrals by decomposing the respective integrands as a sum of fundamental, irreducible contributions, yielding an expression for the amplitude as a linear combination of Master Integrals.

We describe a new formulation of this problem, based on simple concepts of algebraic geometry, which allows to easily derive known one-loop results – where all the Master Integrals are known – and extend them to higher loop orders. We show how the integrand decomposition can be found, either by evaluating the integrand on multiple cuts where some loop propagators are put on-shell, or within a purely algebraic approach by recursively applying the multivariate polynomial-division algorithm to integrands with a smaller and smaller number of loop propagators. We show explicit examples of these two approaches at one and two loops.

In the one-loop case, we propose a new algorithm for the reduction, which consists in finding the coefficients of the Master Integrals by computing suitable Laurent series expansions of the integrands. The algorithm has been implemented in the semi-numerical C++ library NINJA, which has been made publicly available. Since the integrands are rational functions of the loop variables, the Laurent expansion can be performed via a partial fraction decomposition which has been implemented via a simplified polynomial-division algorithm. The library has been interfaced to the one-loop package GOSAM and applied to several phenomenological computations, including next-to-leading order corrections to Higgs production in association with a top quark pair and a jet and some phenomenological analysis for Higgs boson production in gluon fusion in association with two and three jets. NINJA proved to have good performance and numerical stability, being suited for applications to complex processes.

In the last part of the thesis, we propose a new method for finding relations between integrals which are independent at the integrand level, by combining integrand-reduction methods and identities between integrals in a different number of dimensions.

Publications

This thesis is based on the following publications:

Articles

- G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella and G. Ossola *et al.*, “GoSam-2.0: a tool for automated one-loop calculations within the Standard Model and beyond,” arXiv:1404.7096 [hep-ph].
- T. Peraro, “Ninja: Automated Integrand Reduction via Laurent Expansion for One-Loop Amplitudes,” arXiv:1403.1229 [hep-ph].
- H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, “Multi-leg One-loop Massive Amplitudes from Integrand Reduction via Laurent Expansion,” JHEP **1403** (2014) 115 [arXiv:1312.6678 [hep-ph]].
- H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, “NLO QCD corrections to Higgs boson production in association with a top quark pair and a jet,” Phys. Rev. Lett. **111**, **171801** (2013) [arXiv:1307.8437 [hep-ph]].
- P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, “Multiloop Integrand Reduction for Dimensionally Regulated Amplitudes,” Phys. Lett. B **727** (2013) 532 [arXiv:1307.5832 [hep-ph]].
- G. Cullen, H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro *et al.*, “NLO QCD corrections to Higgs boson production plus three jets in gluon fusion,” Phys. Rev. Lett. **111** (2013) 131801 [arXiv:1307.4737 [hep-ph]].
- H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro and J. F. von Soden-Fraunhofen *et al.*, “NLO QCD corrections to the production of Higgs plus two jets at the LHC,” Phys. Lett. B **721** (2013) 74 [arXiv:1301.0493 [hep-ph]].

- P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, “Integrand-Reduction for Two-Loop Scattering Amplitudes through Multivariate Polynomial Division,” *Phys. Rev. D* **87** (2013) 085026 [arXiv:1209.4319 [hep-ph]].
- P. Mastrolia, E. Mirabella, G. Ossola and T. Peraro, “Scattering Amplitudes from Multivariate Polynomial Division,” *Phys. Lett. B* **718** (2012) 173 [arXiv:1205.7087 [hep-ph]].
- P. Mastrolia, E. Mirabella and T. Peraro, “Integrand reduction of one-loop scattering amplitudes through Laurent series expansion,” *JHEP* **1206** (2012) 095 [arXiv:1203.0291 [hep-ph]].

Proceedings

- G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella and G. Ossola *et al.*, “NLO QCD Production of Higgs boson plus jets with GoSam,” arXiv:1311.5191 [hep-ph].
- J. Butterworth, G. Dissertori, S. Dittmaier, D. de Florian, N. Glover, K. Hamilton, J. Huston and M. Kado *et al.*, “Les Houches 2013: Physics at TeV Colliders: Standard Model Working Group Report,” arXiv:1405.1067 [hep-ph].
- T. Peraro, H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and U. Schubert, “Integrand reduction at NLO and beyond,” *PoS EPS -HEP2013* (2014) 449.
- G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella and G. Ossola *et al.*, “GoSam @ LHC: algorithms and applications to Higgs production,” *PoS RADCOR 2013* (2014) 029 [arXiv:1312.1761 [hep-ph]].
- T. Peraro, “Integrand-level Reduction at One and Higher Loops,” *Acta Phys. Polon. B* **44** (2013) 2215.
- H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro and U. Schubert, “Multi-loop Integrand Reduction via Multivariate Polynomial Division,” *PoS RADCOR 2013* (2014) 012 [arXiv:1312.1627 [hep-ph]].
- G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella and G. Ossola *et al.*, “GoSam applications for automated NLO calculations,” arXiv:1309.3741 [hep-ph].
- SHeinemeyer *et al.* [LHC Higgs Cross Section Working Group Collaboration], “Handbook of LHC Higgs Cross Sections: 3. Higgs Properties,” arXiv:1307.1347 [hep-ph].

- P. Mastrolia, E. Mirabella, G. Ossola, T. Peraro and H. van Deurzen, “The Integrand Reduction of One- and Two-Loop Scattering Amplitudes,” PoS LL **2012** (2012) 028 [arXiv:1209.5678 [hep-ph]].

Contents

1	Introduction	1
2	Scattering amplitudes in gauge theories	7
2.1	Quantum chromodynamics	7
2.2	The Standard Model	11
2.3	Observables in scattering processes	16
2.4	Tree-level amplitudes	20
2.4.1	Berends-Giele recursion	21
2.4.2	BCFW recursion	23
2.5	Loop amplitudes	25
3	Integrand reduction of loop amplitudes	27
3.1	Integrand reduction via multivariate polynomial division	28
3.2	Two results from algebraic geometry	30
3.3	Reduction techniques	32
3.3.1	Fit-on-the-cut approach	32
3.3.2	Divide-and-conquer approach	34
3.4	An algebraic implementation of the divide-and-conquer approach	35
3.5	The one-loop case: OPP decomposition	36
3.5.1	Fit on the cut at one loop	40
4	One-loop amplitudes via Laurent series expansion with NINJA	45
4.1	The method	46
4.1.1	Higher-rank extension	51
4.2	Implementation in the C++ library NINJA	53
4.2.1	Reduction via polynomial division	53
4.2.2	Master Integrals	55
5	Phenomenological applications of NINJA and GOSAM	57
5.1	Multi-leg massive amplitudes with NINJA and GOSAM	57
5.1.1	The one-loop package GOSAM	58

5.1.2	Interfacing NINJA and GOSAM	59
5.1.3	Precision tests	59
5.1.4	Application to Massive Amplitudes	66
5.2	Higgs boson production plus a top quark pair and a jet	68
5.2.1	Numerical computation and results	69
5.3	Higgs boson production plus two and three jets in gluon fusion	73
5.3.1	The large top-mass limit	75
5.3.2	Numerical computation and results	76
6	Integrand reduction at higher loops	83
6.1	Fit on the cut at two loops	83
6.1.1	Integrands and parametric residues	84
6.1.2	Semi-numerical reduction	91
6.1.3	Analytic computation	94
6.2	Algebraic multi-loop reduction via polynomial division	103
6.2.1	Photon vacuum polarization	103
6.2.2	Diagrams for Higgs production via gluon fusion	106
7	Integrand Reduction and independent Master Integrals	107
7.1	Shouten polynomials, orthogonal vectors and orthogonal tensors	109
7.2	Dimensionally shifted integrals via Schwinger parametrization	110
7.3	One-loop relations	114
7.3.1	Recurrence relations for 1-point integrals	116
7.3.2	Recurrence relations for 2-point integrals	117
7.3.3	Recurrence relations for 3-, 4- and 5-point integrals	119
7.3.4	Examples with QED kinematics	121
7.4	Two-loop relations	122
7.4.1	A simple example	124
8	Conclusions	127
A	Spinor-helicity formalism	129
B	Basic concepts of algebraic geometry	133
B.1	Polynomial ideals	133
B.2	Polynomial division, Gröbner bases and quotient rings	134
B.3	Zero-dimensional ideals	137
C	Usage of the C++ library NINJA	139
C.1	Installation	139
C.2	Basic types and namespaces	142

C.3	Writing the integrand	142
C.3.1	Definition of the input	143
C.3.2	Using NINJANUMGEN	146
C.4	Running the reduction	147
C.4.1	A simple example	147
C.4.2	The <code>Amplitude</code> class	150
C.4.3	Global settings	154
C.4.4	Controlling the settings from GOSAM	157
C.5	Built-in examples	159
C.5.1	Simple Test	160
C.5.2	Four-photon helicity amplitudes	160
C.5.3	Six-photon helicity amplitudes	162
C.5.4	Five-point diagram of $gg \rightarrow Ht\bar{t}$	162
C.5.5	Higher-rank example	163
C.5.6	Usage in multi-threaded applications	164
C.6	The PYTHON package NINJANUMGEN	169
C.7	Interfaces to Integral Libraries	171
C.7.1	Built-in interfaces	172
D	Benchmarks of NINJA and GOSAM	173
D.1	$pp \rightarrow W + 3$ jets	174
D.2	$pp \rightarrow Z + 3$ jets	175
D.3	$pp \rightarrow Z Z Z + 1$ jet	176
D.4	$pp \rightarrow W W Z + 1$ jet	177
D.5	$pp \rightarrow W Z Z + 1$ jet	178
D.6	$pp \rightarrow W W W + 1$ jet	179
D.7	$pp \rightarrow Z Z Z Z$	180
D.8	$pp \rightarrow W W W W$	181
D.9	$pp \rightarrow t\bar{t}b\bar{b}$	182
D.10	$pp \rightarrow t\bar{t} + 2$ jets	183
D.11	$pp \rightarrow Z b\bar{b} + 1$ jet	184
D.12	$pp \rightarrow W b\bar{b} + 1$ jet	185
D.13	$pp \rightarrow W b\bar{b} + 2$ jets	186
D.14	$pp \rightarrow W W b\bar{b}$	188
D.15	$pp \rightarrow W W b\bar{b} + 1$ jet	190
D.16	$pp \rightarrow H + 3$ jets (GF)	191
D.17	$pp \rightarrow Z t\bar{t} + 1$ jet	193
D.18	$pp \rightarrow H t\bar{t} + 1$ jet	194
D.19	$pp \rightarrow H + 3$ jets (VBF)	195

D.20 $pp \rightarrow H + 4$ jets (VBF)	196
D.21 $pp \rightarrow H + 5$ jets (VBF)	197

Chapter 1

Introduction

The study of the most fundamental laws of nature requires to take into account the principles of *Quantum Mechanics* and *Relativity*. This led to the formulation of *Quantum Field Theories* (*QFT*), where one can consistently combine these two theories in a framework which treats particles as excited quantum states of physical fields. Among Quantum Field Theories, *gauge theories* have provided so far the most accurate description of the fundamental laws of nature observed in high-energy interactions, where both quantum and relativistic effects play a very important role. In the theoretical formulation of a gauge theory, the definition of the fields and their interactions is based on a symmetry of nature, which is mathematically described by a *symmetry group*.

The theory which currently provides the best description of the known fundamental particles and their interactions is the so-called *Standard Model* (*SM*). The Standard Model is a gauge theory based on the symmetry group $SU(3)_C \otimes SU(2)_L \otimes U(1)_Y$. The subgroup $SU(3)_C$ defines the theory of *Quantum Chromo-Dynamics* (*QCD*) which describes the so-called *strong interactions*. The subgroup $SU(2)_L \otimes U(1)_Y$ defines instead the *electro-weak interactions* which combine the electromagnetic interaction of *Quantum Electro-Dynamics* (*QED*) and the weak interactions. While the symmetry group of QCD is realized in the SM as an exact symmetry of nature, the $SU(2)_L \otimes U(1)_Y$ symmetry is *broken* via the so-called Higgs mechanism. The Higgs particle is a scalar responsible for the symmetry breaking and for all the masses of the other particles of the SM.

The SM has been validated by an extremely large number of experiments, among which the most accurate measurements ever performed. Among these, there are experiments at *particle colliders*, which study the scattering of particles. Notable colliders which confirmed various predictions of the SM are the *Large Electron-Positron Collider* (*LEP*) at CERN, the *Tevatron* at Fermilab, and more recently the *Large Hadron Collider* (*LHC*) which replaced LEP at CERN and is currently the largest and most powerful particle collider ever built. The main important missing observation predicted by the SM has been for many years the one of the Higgs boson. The search for the Higgs particle (or an alternative mechanism which

would give masses to the particles of the SM) was indeed one of the main reasons which lead to build the LHC. On 4 July 2012, the collaborations of the ATLAS and CMS experiments at LHC reported the observation of a new scalar particle [1,2]. The experimental data which are available at the time of writing are consistent with the assumption that this scalar particle is the Higgs boson predicted by the SM. The study of the properties of this particle is currently one of the main goals of particle physics phenomenology.

Despite its tremendous success, both at the theoretical and at the experimental level, the SM has some important shortcomings which prevent it from being considered the fundamental theory of elementary interactions. Indeed the SM does not provide a description of the gravitational interaction compatible with Quantum Mechanics. Moreover, it fails to explain the existence of dark matter and dark energy deduced from cosmological studies and observations, which together are estimated to constitute about the 95% of the content of the observable universe. Other open problems are phenomena such as neutrino oscillations and the origin of the observed asymmetry between matter and anti-matter. Therefore, in recent years a lot of effort has been put into the formulation of theories which extend the SM as well as in the experimental search of deviations from its predictions which could shed light on the presence of new physics, commonly referred to as physics Beyond the Standard Model (BSM). So far no clear evidence of deviations from the SM predictions has however been observed.

Both the study of the properties of the new observed scalar particle and the search for signals of new physics require to make accurate theoretical predictions to be compared with the experimental data. This need of accurate theoretical predictions has become particularly important in recent years, in order to compare old and new theoretical models with the experimental data coming from the LHC. The latter proved to be an extremely powerful machine, capable of validating the SM in previously unexplored regions of the phase space, as well as making important discoveries, such as the new observed scalar we mentioned above. The high center-of-mass energy in LHC collisions makes it capable of probing the physics at high energy scales and thus at a more fundamental level. However, LHC interactions are also characterized by a large background (mainly coming from QCD) which could hide signals of new or important physics. In order to be able to make the best use of the data from LHC and other high-energy experiments, one therefore needs to understand the physics of the SM as accurately as possible.

Performing accurate predictions in a QFT such as the SM is not a trivial task. Indeed, exact computations of observables in QFT are currently not possible for realistic models and we therefore need to resort to *perturbation theory*. In perturbation theory, the result of a computation is expanded in powers of the coupling constants. The computation of each term in this expansion is, in principle, possible thanks to techniques such as Feynman diagrams and Feynman rules. It is by now well known that leading-order (LO) approximations in perturbation theory are not reliable, because their theoretical uncertainty is too large, often of the same order of magnitude of the observable that is computed. When accuracy is important,

one is therefore required to perform perturbative computations at next-leading-order (NLO) or beyond.

A key ingredient for obtaining theoretical predictions for collider experiments is the computation of *scattering amplitudes*, which are related to the probability of a given interaction between fundamental particles. The amplitudes give information about the process-dependent part of a physical event and thus represent the main point of contact between a theoretical model and the related phenomenology. Because of the high center-of-mass energy available at the LHC, many particles are often produced in the final state of a collision. This, in turn, requires the computation of amplitudes with many external particles (or legs). The latter is a highly non-trivial task, especially at higher orders in perturbation theory, where it requires the computation of complex *loop integrals* which take into account the quantum effects of the theory. For these reasons in recent years, several methods, algorithms and automated tools for the computation of scattering amplitudes have been developed and used for numeric, semi-numeric and analytic computations.

Among these methods, the ones inspired by *unitarity* have been particularly successful. One of the most important consequences of unitarity, which encodes the conservation of probability in Quantum Mechanics, is the *optical theorem* which connects the total cross section to the imaginary part of the forward scattering amplitude. The latter is in turn determined by the discontinuity of the Feynman diagrams across their branch cuts. According to Cutkosky's *cutting rules* [3], this discontinuity is given by a sum over all the possible ways to *cut* – i.e. to put *on-shell* – two internal propagators. On such a cut, an amplitude factorizes as the product of two lower-point amplitudes. More recently, unitarity inspired new approaches for the perturbative computation of scattering amplitudes, with the formulation of the concepts of *generalized unitarity* and *generalized cuts* [4–7]. These techniques showed that Feynman diagrams can be grouped according to their multi-particle factorization channels, by cutting an arbitrary number of propagators which can simultaneously satisfy the on-shell conditions. These methods have been inspired by new insights on the mathematical properties of scattering amplitudes which came from Witten's interpretation of perturbative gauge theory as a string theory in twistor space [8]. This quickly resulted in tremendous progress in the development of mathematical frameworks for their study, within super-symmetric theories (e.g. super-conformal symmetry [9–12], grassmanians [13–15], string-gauge [16] and gravity-gauge [17, 18] duality), and more general on-shell [6, 7, 19–21] and generalized-unitarity [4, 22–37] methods.

In the context of the computation of one-loop amplitudes, one can exploit the knowledge that every one-loop amplitude is a linear combination of known Master Integrals [38] with up to four external legs. Methods for the computation of complex one-loop amplitudes, such as the improved tensor reduction [39] and the already-mentioned unitarity-based techniques, thus aim to extract the coefficients of this linear combination. A new level of understanding of one-loop scattering amplitudes came with the development of the *Integrand Reduction method* [40, 41]. The mathematical structure of the integrated amplitudes has been shown to be related

to a *universal decomposition* valid for the integrands and known as *OPP decomposition*. The coefficients of the Master Integrals can be found, as already suggested by generalized-unitarity methods, by evaluating the integrands on the *multiple cuts*, where a subset of loop denominators go on-shell. The integrand-level reduction, being based on a decomposition of the integrand valid for any QFT, is a completely general method for the computation of one-loop integrals. This allowed to overcome some issues of other unitarity-based methods (whose application has been mostly restricted to amplitudes with massless loop propagators). The automation of the algorithm [42, 43] and its usage within several automated frameworks [44–52] has considerably enhanced our possibility of making phenomenological predictions in QFT for processes characterized by an highly non-trivial complexity.

At two and higher loops, the basis of Master Integrals is instead not known. Hence, the Master Integrals are identified only after the reduction of the amplitude, when they need to be computed. The most successful reduction method for higher-loop amplitudes has been, so far, Integration by Parts (IBP) [53, 54], especially using the so-called Laporta algorithm [55]. The computation of the Master Integrals has been addressed using a wide range of techniques, such as difference [55–57] and differential [58–68] equations, Mellin-Barnes integral representation [69–72], asymptotic expansions [73–75], sector decomposition [76–83], and contour deformation [84]. At two loops, generalized-unitarity techniques have been introduced for supersymmetric theories [85] and later for QCD amplitudes [86]. More recently, the maximal-unitarity approach has been proposed [87–90], aiming to systematically extract the coefficients of the Master Integrals by using suitable integration contours in the complex plane. The first generalization of integrand-reduction methods at higher loops has been made in Ref.s [91, 92] which stressed the importance of finding the generic integrand decomposition for higher-loop topologies – which extends the known one-loop OPP decomposition – and showed the possibility to compute the coefficients of higher-loop Master Integrals by evaluating the integrand on multiple cuts, along the same lines as the one-loop case.

In this thesis we explore and develop new integrand-reduction methods for the computation of scattering amplitudes at the loop level. As we mentioned, these techniques have been originally developed for one-loop diagrams [40, 41] and recently extended to higher loops [91–95]. These methods can be seen as theoretical tools for better understanding the analytic and algebraic structure of the loop integrands of scattering amplitudes, as well as for the development of algorithms for obtaining phenomenological predictions in QFT. The reduction at the integrand level rewrites scattering amplitudes as linear combinations of *Master Integrals*. More in detail, each integrand is a rational function of the integration variables and can be rewritten as a linear combination of *irreducible* terms, which then need to be integrated. These irreducible terms are *universal* and only depend on the topology of the Feynman diagram associated to a given loop integral, while the coefficients of their linear combination are process-dependent. Therefore, the integrand reduction splits the integrands in fundamental and universal building blocks and can be a powerful tool for the evaluation

of the respective integrals. Its most successful application has been so far the computation of one-loop amplitudes, which can always be reduced at the integrand level as a sum of known Master Integrals.

More in detail, in this thesis we deal with the integrand reduction both at one and higher loops. In the former case we mainly focus on the development and implementation of a new improved integrand-reduction method called *integrand reduction via Laurent series expansion* [96], as well as its usage in phenomenological applications. In the latter we develop a framework for the extension of integrand reduction methods at all loop orders.

The *integrand reduction via Laurent series expansion* is a method for the integrand reduction of one-loop scattering amplitudes to a linear combination of known Master Integrals by performing suitable Laurent expansions of the respective integrands. The coefficients of this linear combination can indeed be identified with the ones of properly defined Laurent expansions of the integrand, corrected by some known counter-terms. This yields a reduction algorithm which is lighter and more efficient than the original. We implemented the algorithm in the C++ library NINJA [97], which has been interfaced with the one-loop package GOSAM [49, 98] and used for the computation of several highly non-trivial scattering amplitudes [99] as well as for the calculation of total and differential cross sections at NLO for Higgs boson production in association with a top-quark pair and a jet [100] and more recently for new phenomenological analysis about Higgs boson production in gluon fusion in association with two and three jets [101]. These applications showed that NINJA is faster and numerically more stable than implementations of the original integrand reduction algorithm and it is especially suited for applications to complex one-loop processes.

The extension of integrand reduction methods to higher loops is quite recent and constitutes one of the main results presented in this thesis. It relies on a reformulation of the problem in terms of basic concepts of algebraic geometry such as *multivariate polynomial division* [93, 94]. This allows to find the most general parametric form of the irreducible contributions appearing in the decomposition of loop integrands. The integrand reduction can be performed at any loop, as traditionally done in the one-loop case, by evaluating the integrand on values of the loop momenta which put on-shell internal loop propagators. This is also known as *fit-on-the-cut* approach for the integrand reduction, which has been applied for the reduction of 5-point integrands in $\mathcal{N} = 4$ Super-Yang-Mills (SYM) and $\mathcal{N} = 8$ Super-Gravity (SUGRA) theories, where the simplicity of the integrands make them suited for trying out new methods. This reformulation of the problem also lead us to develop an alternative and purely algebraic reduction strategy, which performs the reduction via recursive polynomial divisions of the integrands, and is a more general approach which can be applied to any integrand [95]. This is known as *divide-and-conquer* approach. Using polynomial division techniques one can also extend the original integrand-reduction algorithm to theories which allow integrands with higher rank. This higher-rank extension has been used in order to compute NLO corrections to Higgs boson production plus two [102] and three [103] jets with

integrand reduction methods.

This thesis is organized as follows. In Chapter 2 we review some basic and well-known concepts about scattering amplitudes in gauge theories, such as QCD and the SM, as well as their relation to physical observables. In Chapter 3 we present a general description of integrand-reduction methods within a coherent framework based on multivariate polynomial division and valid at all loop orders. We also present the main reduction techniques, we derive the known one-loop case as a special case of this framework and we show how to extend it to theories with higher-rank integrands. In Chapter 4 we present the one-loop integrand reduction via Laurent expansion method and its implementation in the public library NINJA. In Chapter 5 we describe some phenomenological applications of NINJA and the one-loop package GOSAM. In Chapter 6 we present some applications of the integrand reduction method at higher loops, using both the *fit-on-the-cut* and the *divide-and-conquer* reduction techniques. In Chapter 7 we present a new method for obtaining additional relations between loop integrals, using a combination of integrand reduction and dimensional shifts, with applications both at one and higher loops. A brief review of the so-called spinor-helicity formalism which is used in several parts of the thesis is given in Appendix A. The basic concepts of algebraic geometry we used are instead reviewed in Appendix B. Appendix C contains more details on the usage of the C++ library NINJA. Appendix D contains instead a list of numerical benchmarks of one-loop scattering amplitudes computed with GOSAM+NINJA, which could be useful as a reference for future implementations or calculations.

Chapter 2

Scattering amplitudes in gauge theories

In this chapter we review some basic concepts about the *perturbative* computation of *scattering amplitudes* in *Quantum Field Theory* (QFT), and in particular in *gauge theories*. We focus on *Quantum Chromodynamics* (QCD) and the *Standard Model* (SM). We will give a brief summary of these theories and the computation of the corresponding scattering amplitudes, as well as their relation to physical observables.

This chapter is meant to be an introduction to the theoretical and phenomenological topics which will be used and developed in the rest of the thesis. We do not attempt to make a comprehensive treatment on these subjects, most of which are well known. The reader is therefore expected to be already familiar with most of the notions presented in the following sections (for which we refer to textbooks and manuals such as Ref.s [104–107]).

2.1 Quantum chromodynamics

Quantum Chromodynamics (QCD) is a *non-abelian gauge theory* with symmetry group $SU(N_c)$, with $N_c = 3$. It describes the so-called *strong interaction* between n_f flavors of *quarks* and the *gluons*. The quarks are spin-1/2 fermions and the gluons are the vector bosons which mediate the interaction. The Lagrangian of QCD is

$$\mathcal{L} = -\frac{1}{4}G_{\mu\nu}^a G_a^{\mu\nu} + \sum_{f=1}^{n_f} \bar{\psi}_f (i\not{D} - m_f)\psi_f + \mathcal{L}_{\text{gf}}, \quad (2.1)$$

where the field tensor $G_a^{\mu\nu}$ can be written in terms of the gluon vector field A_a^μ as

$$G_a^{\mu\nu} = \partial^\mu A_a^\nu - \partial^\nu A_a^\mu + g_s f_{abc} A_b^\mu A_c^\nu. \quad (2.2)$$

The vector field also defines the covariant derivative as

$$D^\mu = \partial^\mu - ig_s t^a A_a^\mu, \quad (2.3)$$

Flavor	up (u)	down (d)	charm (c)	strange (s)	top (t)	bottom (b)
El. charge	2/3	-1/3	2/3	-1/3	2/3	-1/3
Mass	~ 2.3 MeV	~ 4.8 MeV	1.275 GeV	95 MeV	173 GeV	4.18 GeV

Table 2.1: Quark flavors, with their electric charge and approximate mass.

while the ψ_f are the quark fields. In these equations g_s is the coupling constant of the interaction, while f_{abc} and t^a are the structure constants and the generators of the symmetry group $SU(N_c)$ respectively. The contribution \mathcal{L}_{gf} to Eq. (2.1) is a gauge fixing term. Although many *gauge choices* are possible, the most common ones are the so-call R_ξ gauges which have the form

$$\mathcal{L}_{\text{gf}} = -\frac{1}{2\xi} (\partial_\mu A^\mu)^2. \quad (2.4)$$

Typical choices of ξ are $\xi = 1$, known as *Feynman–t Hooft gauge*, and the limit $\xi \rightarrow 0$, known as *Landau gauge*. In this thesis we will mostly use the Feynman–t Hooft gauge.

The “charge” of strongly interacting particles (the QCD analogous of the electric charge in electrodynamics) is called *color*. Each flavor of quark lives in the fundamental representation of the group $SU(N_c)$ and can have (a combination of) $N_c = 3$ different colors. The gluons, which are the gauge bosons and thus live in the adjoint representation of the symmetry group, can have $N_c^2 - 1 = 8$ different colors. In QCD computations, one typically deals with all the different colors simultaneously by means of $SU(N_c)$ algebra, often referred to in this context as *color algebra*. Because of *confinement*, colored particles have never been observed as free states, but only bound into composite objects called *hadrons* (such as protons and neutrons) whose total color charge is zero.

Quantum chromodynamics provides an accurate description of the strong interactions observed in nature. So far, $n_f = 6$ different flavors of quarks have been observed and they are listed in Table 2.1. In computations relevant for high-energy processes at colliders such as LHC, the mass of the lightest flavors, namely all but the top and in some cases the bottom quark, is usually neglected.

For the sake of completeness, we remind that the *quantization* of a non-abelian gauge theory such as QCD requires the introduction of unphysical fields known as *ghosts*, which in the computation of scattering amplitudes and physical observables cancel out the contributions from unphysical polarizations of the gauge bosons.

In Fig. 2.1 we list the Feynman rules for QCD, including the ones involving ghosts. We have 3- and 4-point interactions between gluons, as well as gluon-quark-quark and gluon-ghost-ghost 3-point interactions. Scattering amplitudes can be computed, at any order in perturbation theory, by summing the contributions of the *Feynman diagrams* obtained by combining the Feynman rules. A perturbative computation in QCD is typically organized by

expanding the result in powers of the coupling α_s defined as

$$\alpha_s \equiv \frac{g_s^2}{4\pi}. \quad (2.5)$$

The *renormalization group equation* for α_s is

$$Q^2 \frac{\partial}{\partial Q^2} \alpha_s(Q) = \beta(\alpha_s) \quad (2.6)$$

where Q is the physical scale of the process and the β *function* can be computed perturbatively. The value of α_s is determined by the fits of experimental data and it is typically given at a scale equal to the mass of the Z boson ($\alpha_s(m_Z) \simeq 0.1186$), and evolved to any other scale using Eq. (2.6). In QCD the β function is *negative* and the coupling can be shown to decrease logarithmically with the scale, in first approximation as $1/\ln Q^2$. This phenomenon, known as *asymptotic freedom*, is particularly important because it allows to use perturbation theory at high energy, where the coupling becomes smaller. However, the value of the coupling α_s is still quite large compared to the ones of other interactions (for instance, the electromagnetic coupling $\alpha \simeq 1/137$, while weak interactions are suppressed by the mass of the W and Z gauge bosons). Hence, QCD interactions are the most important ones at colliders, especially at hadron colliders such as LHC. In particular, given the high center-of-mass energy available at LHC, many external particles are produced in the final state. The production of quarks and gluons in the final state of a partonic interaction is signaled by the presence of *jets* in the observed physical final state (see Section 2.3 for more details about jets). This makes the computation of amplitudes with many external legs very important.

$$\begin{aligned}
& \begin{array}{c} \mu_1, a_1 \quad k \quad \mu_2, a_2 \\ \text{oooooo} \end{array} = \frac{-i\delta_{a_1 a_2}}{k^2 + i\epsilon} \left(-g^{\mu_1 \mu_2} + (1 - \xi) \frac{k^{\mu_1} k^{\mu_2}}{k^2 + i\epsilon} \right) \\
& \begin{array}{c} j_1 \quad k \quad j_2 \\ \text{-----} \end{array} = \frac{i\delta_{j_1 j_2} \not{k}}{k^2 - m_f^2 + i\epsilon} \\
& \begin{array}{c} j_1 \quad k \quad j_2 \\ \text{-----} \end{array} = \frac{i\delta_{j_1 j_2}}{k^2 + i\epsilon} \\
& \begin{array}{c} g_1 \\ \text{oooo} \\ \bullet \\ \text{oooo} \quad \text{oooo} \\ g_2 \quad g_3 \end{array} = g_s f^{a_1 a_2 a_3} \left(g^{\mu_1 \mu_2} (k_1 - k_2)^{\mu_3} \right. \\
& \qquad \qquad \qquad \left. + g^{\mu_2 \mu_3} (k_2 - k_3)^{\mu_1} \right. \\
& \qquad \qquad \qquad \left. + g^{\mu_3 \mu_1} (k_1 - k_2)^{\mu_2} \right) \\
& \begin{array}{c} g_1 \quad g_2 \\ \text{oooo} \quad \text{oooo} \\ \bullet \\ \text{oooo} \quad \text{oooo} \\ g_3 \quad g_4 \end{array} = -ig_s^2 \left(f^{a_1 a_2 b} f^{a_3 a_4 b} (g^{\mu_1 \mu_3} g^{\mu_2 \mu_4} - g^{\mu_1 \mu_4} g^{\mu_2 \mu_3}) \right. \\
& \qquad \qquad \qquad \left. + f^{a_1 a_3 b} f^{a_2 a_4 b} (g^{\mu_1 \mu_2} g^{\mu_3 \mu_4} - g^{\mu_1 \mu_4} g^{\mu_2 \mu_3}) \right. \\
& \qquad \qquad \qquad \left. + f^{a_1 a_4 b} f^{a_2 a_3 b} (g^{\mu_1 \mu_2} g^{\mu_3 \mu_4} - g^{\mu_1 \mu_3} g^{\mu_2 \mu_4}) \right) \\
& \begin{array}{c} g_1 \\ \text{oooo} \\ \bullet \\ \text{---} \quad \text{---} \\ q_2 \quad q_3 \end{array} = -ig_s t_{j_1 j_3}^{a_1} \gamma^{\mu_1} \\
& \begin{array}{c} g_1 \\ \text{oooo} \\ \bullet \\ \text{---} \quad \text{---} \end{array} = g_s k_1^{\mu_1}
\end{aligned}$$

Figure 2.1: Feynman rules of QCD in R_ξ gauge. Arrow lines are quarks, curly lines are gluons, and dashed lines are ghost fields. Quark color indices are indicated with j_1, j_2, \dots , gluon color indices with a_1, a_2, \dots , while a sum over the repeated index b is understood. The momenta in 3-point vertices are assumed to be incoming.

2.2 The Standard Model

The *Standard Model* currently provides the best available description of the known fundamental interactions of nature, with the notable exception of gravity. More in detail, it incorporates QCD, briefly reviewed in Section 2.1, and the *electroweak* (*EW*) interaction. The latter describes the electromagnetic and the weak forces between elementary particles. More in detail, the Standard Model is a gauge theory based on the symmetry group $SU(3)_C \otimes SU(2)_L \otimes U(1)_Y$. The symmetry group $SU(3)_C$ is the color symmetry of QCD, which according to the Standard Model is realized as an exact symmetry of nature. The group $SU(2)_L \otimes U(1)_Y$ describes instead the electroweak interactions and is *spontaneously broken* via the Higgs mechanism, which yields three massive vector bosons, respectively known as Z, W^+, W^- . The Higgs field is responsible for the masses of both the electroweak gauge bosons and the fermions of the theory. In this section we will give a brief review of the electroweak sector of the Standard Model.

The symmetry group of the EW interaction is the direct product of the symmetry groups $SU(2)_L$ and $U(1)_Y$. The former defines a non-abelian *chiral* symmetry which only affects the left-handed components of the fermion fields. The latter is instead an abelian symmetry group. The interactions are, as usual, mediated by the gauge bosons, whose Lagrangian reads

$$\mathcal{L}_{SU(2)_L \otimes U(1)_Y} = -\frac{1}{4} W_i^{\mu\nu} W_{\mu\nu}^i - \frac{1}{4} B^{\mu\nu} B_{\mu\nu} \quad (2.7)$$

with the field tensors $W_i^{\mu\nu}$ ($i = 1, 2, 3$) and $B_i^{\mu\nu}$ defined in terms of the gauge vector fields W_i^μ and B_i^μ as

$$\begin{aligned} W_i^{\mu\nu} &= \partial^\mu W_i^\nu - \partial^\nu W_i^\mu + g_W \epsilon_{ijk} W_j^\mu W_k^\nu \\ B^{\mu\nu} &= \partial^\mu B^\nu - \partial^\nu B^\mu, \end{aligned} \quad (2.8)$$

where g_W is the gauge coupling of $SU(2)_L$ and ϵ_{ijk} is the Levi-Civita symbol which represents the structure constants of $SU(2)$. The covariant derivative is defined as the matrix operator

$$D^\mu = \partial^\mu - ig_W t^i W_i^\nu - ig'_W Y B^\mu, \quad (2.9)$$

where the t^i are the generators of $SU(2)$ (i.e. the well-known Pauli matrices) and g'_W is the coupling of the group $U(1)_Y$. In the previous equation we also introduced the *diagonal* matrix Y whose elements are the charges (in units of g'_W) of the particles with respect to the interaction of the symmetry group $U(1)_Y$. For each particle, this quantum number is also known as *hypercharge*. Because $U(1)$ is an abelian group, every particle can have a different hypercharge. Moreover, the left- and the right-handed components of a fermion can also have different hypercharges. One can replace the fields W_1^μ and W_2^μ with the fields W_\pm^μ defined by

$$W_\pm^\mu = \frac{1}{\sqrt{2}} (W_1^\mu \mp iW_2^\mu) \quad (2.10)$$

so that Eq. (2.9) becomes

$$D^\mu = \partial^\mu - ig_W \left(t^3 W_3^\nu + \frac{1}{\sqrt{2}} t^+ W_+^\mu + \frac{1}{\sqrt{2}} t^- W_-^\mu \right) - ig'_W Y B^\mu. \quad (2.11)$$

where $t^\pm \equiv t^1 \pm it^2$ are the *isospin* raising and lowering operators.

According to experimental observations, the vector bosons of the electroweak interactions are massive. However, adding mass terms to the Lagrangian in Eq. (2.7) is well known to yield a non-renormalizable theory, which would negatively affect our capability of making perturbative predictions. The main reason for the introduction of the Higgs field in the Standard Model is the possibility to exploit the Higgs mechanism in order to give masses to the vector bosons. In this case, the mass of a vector boson is not an intrinsic property of the particle, but a dynamic effect which preserves the renormalizability of the theory.

The Higgs field is a doublet of scalar fields $\phi = (\phi^+, \phi^0)$. Its Lagrangian is

$$\mathcal{L}_\phi = (D^\mu \phi)^\dagger (D_\mu \phi) - V(\phi), \quad V(\phi) = \lambda |\phi|^4 - \mu^2 |\phi|^2. \quad (2.12)$$

The *spontaneous symmetry breaking* is due to the negative mass term in the potential $V(\phi)$, which thus has a classical minimum which is not at $\phi = 0$ but at values of ϕ such that

$$|\phi| = \sqrt{\frac{\mu^2}{2\lambda}} \equiv \frac{v}{\sqrt{2}}. \quad (2.13)$$

Therefore, in order to quantize the theory, we can choose a particular direction of the minimum of ϕ , e.g.

$$\langle \phi \rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix} \quad (2.14)$$

and consider fluctuations of ϕ about this minimum. Up to a gauge choice (*unitary gauge*), we can parametrize ϕ as

$$\phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + H \end{pmatrix}. \quad (2.15)$$

where H is the physical Higgs field. By inserting this expression in the Higgs Lagrangian of Eq. (2.12) one obtains a sum of interaction terms, including self-interactions of the Higgs field H as well as interactions between the Higgs and the vector bosons, and quadratic terms in the vector bosons which therefore represent mass terms. Since the propagator for the B^μ and W_3^μ fields is not diagonal, we can diagonalize it by defining the fields A^μ and Z^μ

$$\begin{pmatrix} Z^\mu \\ A^\mu \end{pmatrix} = \begin{pmatrix} \cos \theta_W & \sin \theta_W \\ \sin \theta_W & \cos \theta_W \end{pmatrix} \begin{pmatrix} W_3^\mu \\ B^\mu \end{pmatrix}, \quad (2.16)$$

where the *Weinberg angle* θ_W is given by

$$\sin^2 \theta_W = \frac{g_W'^2}{g_W'^2 + g_W^2}. \quad (2.17)$$

Hence the vector field A is identified with the (massless) *electromagnetic* field, i.e. the *photon* field of *Quantum electrodynamics* (QED), while Z is a massive gauge boson of the electroweak interaction. With an explicit calculation, one can check that in the final Lagrangian the bosons W_{\pm} and Z have respectively acquired masses m_W and m_Z given by

$$m_W = \frac{1}{2}v g_W, \quad m_Z = \frac{1}{2}v \sqrt{g_W^2 + g_W^2} = \frac{m_W}{\cos \theta_W}. \quad (2.18)$$

The Higgs boson also acquires a physical mass ($m_H^2 > 0$) given by

$$m_H = \sqrt{2}\mu = \sqrt{2\lambda}v. \quad (2.19)$$

This completes our brief description of the scalar and vector bosons of the Standard Model. The next step will be the description of the fermions and their couplings.

Experimental observations show that there are two kinds of fermions: *quarks* and *leptons*. Quarks were already introduced in Section 2.1. They are subject to both strong and electroweak interactions. As already stated, six flavors of quarks have been observed (see Table 2.1). The left handed components of the quarks are organized into $SU(2)_L$ doublets

$$P_L \begin{pmatrix} u \\ d \end{pmatrix}, \quad P_L \begin{pmatrix} c \\ s \end{pmatrix}, \quad P_L \begin{pmatrix} t \\ b \end{pmatrix}, \quad (2.20)$$

where the matrix P_L and P_R project out the left- and right-handed components of the fields respectively and they are defined by

$$P_L = \frac{1}{2}(1 - \gamma_5), \quad P_R = \frac{1}{2}(1 + \gamma_5). \quad (2.21)$$

The right-handed components ($P_R u, P_R d, P_R c, P_R s, P_R t, P_R b$) are instead all singlets with respect to $SU(2)_L$. Leptons are instead only subject to the *electroweak* interactions. Similarly to the case of the quarks, we have six different flavors of leptons, whose left-handed components are also organized in $SU(2)_L$ singlets

$$P_L \begin{pmatrix} \nu_e \\ e^- \end{pmatrix}, \quad P_L \begin{pmatrix} \nu_\mu \\ \mu^- \end{pmatrix}, \quad P_L \begin{pmatrix} \nu_\tau \\ \tau^- \end{pmatrix}. \quad (2.22)$$

The right-handed components of the leptons, like the ones of the quarks, are also $SU(2)_L$ singlets. The electron field e^- , the μ^- field and the τ^- field have the same electric charge but very different masses (approximately $m_e \simeq 0.51$ MeV, $m_\mu \simeq 106$ MeV, $m_\tau \simeq 1.78$ GeV). The neutrinos ν_i only interact with the W_{\pm} and Z bosons. They are very light and, although their actual mass is not known, in high-energy computations they can be assumed to be massless. Notice that, with this assumption, the right-handed component of the neutrinos does not take part in any interaction in the SM and therefore it can be omitted altogether from the Lagrangian.

The coupling of the gauge bosons to the fermions is defined via the covariant derivative given in Eq. (2.9), or alternatively in Eq. (2.11). As already observed, in those equations the

Fermion	left-handed			right-handed		
	Q	t^3	Y	Q	t^3	Y
u, c, t	2/3	1/2	1/3	2/3	0	4/3
d, s, b	-1/3	-1/2	1/3	-1/3	0	-2/3
ν_e, ν_μ, ν_τ	0	1/2	-1	-	-	-
e^-, μ^-, τ^-	-1	-1/2	-1	-1	0	-2

Table 2.2: Electroweak quantum numbers of the fermions in the standard model.

operators t^i and t^\pm only act on the left-handed components of the fields, while the hypercharge matrix Y is diagonal and acts differently on left- and right-handed components of the fermions. In order to assign the correct quantum numbers to each fermion field, one can exploit the identification of the vector field A^μ with the electromagnetic field. The relevant part of the Lagrangian is thus identified with the QED Lagrangian, whose interaction term reads

$$\mathcal{L}_{QED} = A_\mu \bar{\Psi} \gamma^\mu \left(g_W \sin \theta_W t^3 + \frac{Y}{2} g'_W \cos \theta_W \right) \Psi, \quad (2.23)$$

where Ψ is a vector of the fermion fields. We thus make the identification

$$e Q = g_W \sin \theta_W t^3 + \frac{Y}{2} g'_W \cos \theta_W, \quad (2.24)$$

where Q is a diagonal matrix with the electric charges of the fields in units of the electron charge e . From the knowledge of the electric charge of the fields and the arbitrary choice $Y = -1$ for left-handed leptons, one gets the relation

$$t^3 + \frac{Y}{2} = Q, \quad (2.25)$$

and the identification

$$g_W \sin \theta_W = g'_W \cos \theta_W = e. \quad (2.26)$$

This allows to assign all the correct quantum numbers, listed in Table 2.2. One can observe that the matter content of the Standard Model essentially consists in three copies of the same theory with two quarks and two leptons. Indeed the other two families of quarks and leptons only differ from the first for their masses.

Finally, the Higgs mechanism is also needed to give masses to quarks and leptons. As we said, the main motivation for the introduction of the Higgs field was the possibility to build a gauge theory with massive gauge bosons. However, in the case of the Standard Model, the Higgs field is also responsible for the masses of the fermions. Indeed a Dirac mass term in the Lagrangian for a fermion ψ would look like, after splitting it into the right- and left-handed components ψ_R and ψ_L , as

$$-m \bar{\psi} \psi = -m(\bar{\psi}_R \psi_L + \bar{\psi}_L \psi_R).$$

This term would clearly violate the chiral symmetry $SU(2)_L$ which only acts on the left-handed components. We can however introduce a new contribution to the Lagrangian which contains Yukawa couplings between the unbroken Higgs field ϕ and the fermions. For one generation of quarks and leptons, one can have a term of the form

$$\mathcal{L}_Y = -\lambda_d(\bar{q}_L \cdot \phi)d_R - \lambda_u(\epsilon^{ab}\bar{q}_{L,a} \cdot \phi_b^\dagger)u_R - \lambda_e(\bar{e}_L \cdot \phi)e_R + \text{c.c.}, \quad (2.27)$$

with

$$q_L = P_L \begin{pmatrix} u \\ d \end{pmatrix}, \quad e_L = P_L \begin{pmatrix} \nu_e \\ e^- \end{pmatrix}.$$

One can check that \mathcal{L}_Y is gauge invariant and can thus be freely added to the Lagrangian of the SM. After symmetry breaking, this will become

$$\mathcal{L}_Y = -m_d\bar{d}d\left(1 + \frac{H}{v}\right) - m_u\bar{u}u\left(1 + \frac{H}{v}\right) - m_e\bar{e}e\left(1 + \frac{H}{v}\right), \quad (2.28)$$

with

$$m_i = \frac{\lambda_i}{\sqrt{2}}v, \quad i = d, u, e, \quad (2.29)$$

generating thus a mass term and a Yukawa interaction between the fermions and the Higgs. If more fermion families are present, we can proceed in the same way by substituting q_L and e_L with vectors of left-handed fermion doublets and u_R, d_R, e_R with vectors of fermion singlets. In the most general case, the couplings λ_i will thus become generic complex matrices. By diagonalizing the mass sector, one can introduce mixing between the quarks (but not the leptons, if the neutrinos are taken as massless) of different generations, proportional to the elements of the so-called *Cabibbo-Kobayashi-Maskawa* (CKM) matrix. With three generations one can show that, with this mechanism, CP violating terms arise. Indeed, a third generation of quarks was introduced in the theory before an experimental observation, in order to give a theoretical explanation for the observed CP violation in electroweak interactions. At very high energies, where the masses of the two lightest generations of quarks can be neglected, one can often assume the CKM matrix to be diagonal and thus no mixing between different generations of quarks is present.

Perturbative computations of scattering amplitudes within the Standard Model can be made, as in QCD, with the usual tools of QFT, namely Feynman diagrams and Feynman rules. A full list of Feynman rules can be found in several textbooks and manuals, e.g. in Ref. [107]. Since the dominant contributions to Standard Model processes in perturbation theory come from QCD interactions, next-to-leading-order (NLO) QCD corrections usually provide an approximation which is good enough for comparisons with experiments. There are cases however where also electroweak corrections can be important, especially when aiming for higher precision, but their computation is significantly more involved due to the presence of many more couplings and interaction terms. The methods presented in this thesis rely on fundamental properties of QFT and gauge theories and can thus be applied to any theoretical

model. However, most of the applications to higher order computations we will describe involve very complex processes and will contain only QCD corrections beyond the leading order approximation.

The Standard Model of elementary particles has been validated by a tremendous amount of experimental data, among which the most accurate experiments ever performed. For a long time, the only fundamental missing piece of evidence required by the predictions of this theory had been the observation of the Higgs boson. The search for the Higgs has also been one of the main motivations for building the Large Hadron Collider (LHC). On 4 July 2012, the ATLAS and CMS collaborations reported the discovery of a new scalar particle compatible with a Standard Model Higgs bosons with mass between 125 and 126 GeV [1, 2]. The analyses performed at the time of writing are in good agreement with the assumption that the new particle is the Higgs boson of the Standard Model, that couples to other SM particles with a strength proportional to their mass [108, 109]. The study of the properties of this newly discovered particle is one of the main goals of particle physics phenomenology in the near future. Accurate predictions are necessary and will play a crucial role for the complete determination of the properties of such boson [110], and in particular to shed light on the structure of its couplings to the other particles. The dominant production mechanism of the Higgs boson at a proton-proton collider such as LHC is the gluon fusion (GF) $gg \rightarrow H$, where the Higgs couples to the gluons via a heavy-quark (a top) loop. Another important channel is vector boson fusion (VBF), $qq \rightarrow Hqq$ via $W^+W^- \rightarrow H$ or $ZZ \rightarrow H$. Other mechanisms are associated production with vector bosons, $q\bar{q}' \rightarrow WH$ or $q\bar{q} \rightarrow ZH$, and associated production with top quarks $gg, q\bar{q} \rightarrow t\bar{t}H$. Among the results presented in this thesis, we will show NLO corrections to phenomenological predictions for Higgs boson production via GF in association with 2 and 3 jets, and associated production with top quarks and a jet. We will also present the computation of NLO scattering amplitudes for Higgs production in VBF with up to 5 additional jets in the final state.

2.3 Observables in scattering processes

The main physical observables computed in high-energy physics are total and differential *cross sections*. In this section we briefly recall some concepts about the relation between scattering amplitudes and physical observables.

In order to make a theoretical prediction for a realistic scattering process, such as the ones measured at colliders, several ingredients are necessary. The first ingredient is the information about the initial state of the process. When this is constituted by elementary particles, its theoretical description is particularly easy, being identified by the incoming momenta (and helicities, if the incoming beam is polarized). If in the initial state we have composite objects, such as hadrons, we need instead information about their structure. The composition of hadrons (such as protons and neutrons) in terms of partons is encoded in

the *parton distribution functions* (PDFs). These cannot be computed perturbatively and need to be measured experimentally. However, since the structure of the hadrons does not depend on the considered process or experiment, the PDFs are *universal*, i.e. PDFs measured using data from a set of processes (or experiments) can be used in order to make predictions for other processes (or experiments). The next ingredient is a description of the fundamental interactions between the elementary particles involved in the process. These *hard* interactions are described by the *scattering amplitudes* which can be computed in perturbation theory and represent the main *process-dependent* part of a process. For this reason they can also be considered the main point of contact between a theory and the related phenomenology. The final ingredient is the knowledge of how the final state of this elementary interaction further evolves into a physical final state which can be measured in a detector. This final state evolution is in turn the combination of several ingredients, such as parton shower (emission of extra radiation), use of jet algorithms (definition of jets, i.e. the final state signatures of quarks and gluons produced in the hard scattering), and hadronization (how final state partons combine together into hadrons). Similarly to the PDFs, also these final-state ingredients are universal and can usually be implemented in process-independent algorithms. In this thesis we will mostly deal with scattering amplitudes and in particular with the development and the implementation of general methods for their computation.

We first consider an n -point process with 2 incoming elementary particles and $n - 2$ outgoing particles, with kinematic $k_1 k_2 \rightarrow k_3 \cdots k_n$. The unitary *scattering matrix* S of the interaction is defined by

$${}_{out}\langle k_2 \cdots k_n | k_1 k_2 \rangle_{in} = \langle k_2 \cdots k_n | S | k_1 k_2 \rangle \quad (2.30)$$

In order to isolate the part of the S matrix which corresponds to interactions, one usually defines a matrix T such that

$$S = \mathbf{1} + iT. \quad (2.31)$$

The *scattering amplitude* \mathcal{M} is thus defined from T by factoring out a δ function related to momentum conservation,

$$\langle k_2 \cdots k_n | iT | k_1 k_2 \rangle = (2\pi)^4 \delta^{(4)}(k_1 + k_2 - \sum_{j=3}^n k_j) i\mathcal{M}. \quad (2.32)$$

The amplitude \mathcal{M} can be computed in perturbation theory using Feynman diagrams and Feynman rules. The cross-section σ can thus be calculated by integrating the squared scattering amplitude over the phase space of the final state, and dividing by the incoming flux,

$$d\sigma = \frac{1}{2E_1 E_2 |v_1 - v_2|} \prod_{j=3}^n \frac{d^3 k_j}{(2\pi)^2 2E_j} \delta^{(4)}(k_1 + k_2 - \sum_{j=3}^n k_j) |\mathcal{M}|^2. \quad (2.33)$$

When dealing with hadrons in the initial state, one should instead compute a *partonic cross section* between the initial partons and the other elementary particles involved, using

the formula above, and thus convolute it with the PDFs. For a process with two hadrons in the initial state $h_1 h_2 \rightarrow X$, with kinematics $k_1 k_2 \rightarrow \sum_{j \in X} k_j$, the total cross section is a sum of all the possible partonic channels $ij \rightarrow X$

$$d\sigma(h_1(k_1)h_2(k_2) \rightarrow X) = \sum_{ij} \int_0^1 dx_1 dx_2 f_{i,1}(x_1, \mu_F^2) f_{j,2}(x_2, \mu_F^2) \times d\sigma_{ij}(i(x_1 k_1)j(x_2 k_2) \rightarrow X, Q^2/\mu_F^2), \quad (2.34)$$

where $f_{i,h}$ denotes the distribution of the parton i in the hadron h , and Q^2 is the scale of the process. The integration variables x_i represent the fraction of the momentum of the hadron h_i carried by the respective parton involved in the interaction. Depending on the process, there might be a sum on the final states X as well (e.g. requiring a final jet could correspond, at the partonic level, to both a quark or a gluon, hence one needs to sum over all the possibilities).

Starting from NLO in perturbation theory, a scattering amplitude \mathcal{M} usually contains divergences arising from integrals in the loop momenta of the form

$$\int_0^\infty \prod_i d^4 q_i$$

associated to diagrams with loops. These divergences need to be regularized and they typically cancel out only at the end of the computation of physical observables. The most common *regularization scheme* is *dimensional regularization*. It consist in performing the integration in a generic number of dimensions d , i.e.

$$\int_0^\infty \prod_i d^4 q_i \rightarrow \int_0^\infty \prod_i d^d \bar{q}_i.$$

Physical observables are finite for $d \rightarrow 4$, but in general the loop integrals appearing in the computation are not. One can distinguish between *ultraviolet* (UV) and *infrared* (IR) divergences, which in dimensional regularization correspond to the presence of *poles* in $d = 4$ in the results. Infrared divergences can in turn be *soft* and *collinear*. We will assume the reader to be familiar with these types of divergences, as well as their *regularization* and *renormalization*.

The presence of these divergencies can be problematic for numerical and semi-numerical computations. In general, ultraviolet divergences are removed from the scattering amplitudes by *UV renormalization*, after a proper choice of *renormalization scheme*. Infrared divergencies instead, only cancel after summing, order by order in perturbation theory, the contributions of the considered process and the one with emission of extra radiation in the final state. As an example, at NLO a cross section of a process with n external particles will be written as

$$\sigma = \sigma_{LO} + \sigma_{NLO}. \quad (2.35)$$

The LO contribution consists in an integration of the form of Eq. (2.33) with the matrix element \mathcal{M} computed at LO, also known as *Born approximation*. We can thus schematically

write

$$\sigma_{LO} = \int_{n-2} d\sigma_B. \quad (2.36)$$

Since this integration is finite, it can be carried out in $d = 4$, either analytically or numerically. The NLO contribution contains two pieces, commonly known as *virtual* and *real*. The former is the integration over the NLO contributions to the squared matrix element $|\mathcal{M}|^2$. The latter involves instead a process with emission of an additional external particle in the final state. Therefore we can write

$$\sigma_{NLO} = \int_{n-2} d\sigma_V + \int_{n-1} d\sigma_R, \quad (2.37)$$

where V and R stand for virtual and real respectively. Since UV renormalization happens at the amplitude level, we can assume to have already removed the UV divergencies in $d\sigma_V$. This two pieces are separately IR divergent and only their sum is finite. However, since the two integrations are performed on different phase spaces (with $n - 2$ and $n - 1$ outgoing particles respectively), a direct application of the formula above in a numerical phase-space integration is not possible. The most common way of dealing with this problem is rewriting this formula as

$$\sigma_{NLO} = \int_{n-2} \left[d\sigma_V + \int_1 d\sigma_A \right] + \int_{n-1} \left[d\sigma_R - d\sigma_A \right], \quad (2.38)$$

where $d\sigma_A$ is an approximation of $d\sigma_R$ which correctly reproduces its singularities in the infrared limits. In this way, the second integral is finite and can be performed numerically. The contribution we subtracted from $d\sigma_R$ is thus added back to the virtual piece, integrated over the extra particle. Several choices of $d\sigma_A$ are possible and they are known as *subtraction schemes*. One of the most common subtraction schemes is the *dipole subtraction* proposed by Catani and Seymour [111]. This exploits the knowledge of the matrix elements in the soft and collinear limits in order to give a recipe for building $d\sigma_A$ in a process-independent way. Moreover, they propose a choice of $d\sigma_A$ where the integration over the extra particle is performed analytically. Other well known NLO subtraction schemes are the antenna subtraction [112, 113] and the so-called FKS subtraction [114, 115].

In the phenomenological applications of this thesis, we will often deal with processes with many particles in the final state. At the level of scattering amplitudes, some of these final states can be partons, i.e. quarks and gluons. However, as we briefly recalled in Section 2.1, partons are never observed as free states in detectors. After the hard interaction described by the scattering amplitude, they fragment into hadrons. A *jet* is a collimated cone of hadrons and can be regarded as the footprint of a parton in the final state of the hard scattering process. A more accurate definition of jets is given by specifying an algorithm for identifying them from the signals of an event. This is known as *jet algorithm*. Several jet algorithms have been proposed. Among the most popular are clustering algorithms, which consist in defining a *distance* d_{ij} between particles i and j , and d_{iB} between a particle i and the beam. One can thus proceed recursively by finding the minimum of these distances and merge particles i and

j if the minimum is d_{ij} , or identifying i as a jet and remove it from the list of particles if the minimum is d_{iB} . The procedure terminates when there are no particles left. A commonly used version of this algorithm is the anti- k_t jet algorithm proposed in Ref. [116], which we will use for the phenomenological applications presented in this thesis.

Finally, we recall that perturbative results in QFT can depend on unphysical scales, such as the *factorization scale* μ_F which appears when partons are present in the initial state after the factorization defined by Eq. (2.34), and the *renormalization scale* μ_R which appears after UV renormalization. The dependence on these scales would cancel out if we were able to sum all orders in perturbation theory and obtain an exact result. In a fixed-order perturbative computation, this cancellation is however spoiled by the neglected higher-order terms. The most important pieces of these contributions typically take the logarithmic form $\ln^k Q_i^2 / \mu_{R,F}^2$, where Q_i^2 are the physical scales of the process. One can try to minimize the effect of these higher-order terms by choosing values of μ_R and μ_F which are close to the physical scales. Although there is no general recipe for obtaining the best choice of this scale, this is usually taken to be a suitable combination of the masses of the produced external particles (if massive) and the transverse momentum of the final jets. Since the unphysical dependence on the renormalization and factorization scales is an effect of the neglected higher-order terms, it can also be exploited in order to assess the theoretical uncertainty of perturbative results, which can be obtained by varying μ_R and μ_F on a given range. Common practice is choosing a central value $\mu_R = \mu_F = \mu_0$ and then vary them from $\mu_0/2$ to $2\mu_0$.

2.4 Tree-level amplitudes

In this section we review some techniques for the computation of tree-level amplitudes. *Tree-level amplitudes* are scattering amplitudes whose Feynman diagrams contain no loop. Their computation is relatively simple, because it only involves algebraic operations, namely substitution of Feynman rules, followed by *color*, *Lorentz* and γ *algebra*. This kind of calculation can also be efficiently performed by numerical and semi-numerical algorithms. Despite their simplicity, compared e.g. to loop computations, some interesting results on the computation of tree level amplitudes have been obtained in recent years, motivated by a better understanding of the analytic and algebraic structure of amplitudes, as well as the research of efficient methods for their evaluation at high multiplicities or as building blocks for loop amplitudes.

We will give a brief review of two well known recursive techniques, namely *Berends-Giele recursion* [117] and *BCFW recursion* [7,19]. Even though we will not make explicit use of these techniques in the computations presented in this thesis, they are nevertheless very interesting. Indeed they shed light on how one can exploit the knowledge of the analytic and algebraic structure of amplitudes, especially their factorization properties at their singularities. This has been particularly important for the development of more advanced techniques for the computation of loop amplitudes, such as the ones presented in this thesis. Since, for our

purposes, we are not interested here in a systematic study of these techniques but only in the main ideas behind their formulation, we will describe them for a $SU(N_c)$ gauge theory with only gluons (i.e. $n_f = 0$). A more systematic and pedagogical review of these techniques can be found in Ref.s [118, 119], which we will partially follow here.

Both Berends-Giele and BCFW recursion are more conveniently applied to *color-ordered* amplitudes. In order to introduce them, we define the $SU(N_c)$ generators T^a which can be obtained from the standard ones t^a by a change of normalization

$$T^a \equiv \frac{1}{\sqrt{2}} t^a, \quad (2.39)$$

so that

$$[t^a, t^b] = i f^{abc} t^c, \quad [T^a, T^b] = i \sqrt{2} f^{abc} T^c. \quad (2.40)$$

With the new normalization, the additional factors $\sqrt{2}$ that appear in the intermediate steps and the Feynman rules cancel out with the ones which would appear in the total results for the color-ordered amplitudes if we used the standard normalization instead. One can also get rid of the structure constants f^{abc} in the Feynman rules by replacing them with combinations of T^a , namely

$$f^{abc} = -\frac{i}{\sqrt{2}} \left(\text{Tr}(T^a T^b T^c) - \text{Tr}(T^b T^a T^c) \right). \quad (2.41)$$

In this way, the total result can be written as a linear combination of purely kinematic factors multiplied by color factors. More specifically, in a pure Yang Mills theory (i.e. with only gluons), any n -point tree-level amplitude \mathcal{M}_n can be written in this way as

$$\mathcal{M}_n(k_1^{h_1}, \dots, k_n^{h_n}) = g_s^{n-2} \sum_{\sigma \in S_n/Z_n} \text{Tr}(T^{a_{\sigma(1)}} \dots T^{a_{\sigma(n)}}) \mathcal{A}_n(\sigma(k_1^{h_1}), \dots, \sigma(k_n^{h_n})) \quad (2.42)$$

where $k_i^{h_i}$ represents a gluon with momentum k_i and helicity h_i . The kinematic factors \mathcal{A}_n are known as color-ordered amplitudes. They receive contributions from Feynman diagrams whose external particles are in a given order, namely the one specified by the permutation σ , which in the sum runs over all the non-cyclic permutations of the external particles. Color-ordered amplitudes can be computed with diagrammatic techniques, by considering diagrams where the external particles are ordered and using the so-called *color-ordered Feynman rules* (they are listed in the references we mentioned above), which can be obtained from the traditional Feynman rules with the help of Eq. (2.41).

In the following subsections, we will briefly show how to recursively compute the color ordered amplitudes \mathcal{A}_n , using and Berends-Giele and BCFW techniques.

2.4.1 Berends-Giele recursion

Berends-Giele recursion is based on the computation of quantities called *off-shell* currents. An off-shell current $J^\mu(k_1^{h_1}, \dots, k_n^{h_n})$ can be seen as a $(n+1)$ -point scattering amplitude

where one of the external legs, namely the one identified by the Lorenz index μ , is off-shell, while all the others, identified by $k_1^{h_1}, \dots, k_n^{h_n}$, are on-shell. Off-shell currents can be easily be built recursively, using lower-point ones as building blocks for higher-point ones. Indeed the external leg μ can either be attached to a 3-gluon or a 4-gluon vertex. The other legs of the vertex can in turn be identified with off-shell currents with a smaller number of gluons. Putting everything together, we have

$$\begin{aligned}
J^\mu(k_1^{h_1}, \dots, k_n^{h_n}) = & -\frac{i}{p_{i,n}^2} \left(\sum_{i=1}^{n-1} \mathcal{V}_{\mu\nu\rho}^{(ggg)}(p_{1,i}, p_{i+1,n}) J^\nu(k_1^{h_1}, \dots, k_i^{h_i}) J^\rho(k_{i+1}^{h_{i+1}}, \dots, k_n^{h_n}) \right. \\
& \left. + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \mathcal{V}_{\mu\nu\rho\sigma}^{(gggg)} J^\nu(k_1^{h_1}, \dots, k_i^{h_i}) J^\rho(k_1^{h_1}, \dots, k_i^{h_i}) J^\sigma(k_1^{h_1}, \dots, k_i^{h_i}) \right),
\end{aligned} \tag{2.43}$$

where we defined

$$p_{i,j} = \sum_{l=i}^j k_l \tag{2.44}$$

and the expressions for $\mathcal{V}_{\mu\nu\rho}^{(ggg)}$ and $\mathcal{V}_{\mu\nu\rho\sigma}^{(gggg)}$ can be easily worked out from the 3- and 4-gluon vertexes respectively and read

$$\begin{aligned}
\mathcal{V}_{\mu\nu\rho}^{(ggg)}(k_1, k_2) &= \frac{i}{\sqrt{2}} \left(g^{\nu\rho}(k_1^\mu - k_2^\mu) + 2g^{\mu\rho}k_2^\nu - g^{\mu\nu}k_1^\rho \right) \\
\mathcal{V}_{\mu\nu\rho\sigma}^{(gggg)} &= \frac{i}{\sqrt{2}} \left(2g^{\mu\rho}g^{\nu\sigma} - g^{\mu\nu}g^{\rho\sigma} - g^{\mu\sigma}g^{\nu\rho} \right).
\end{aligned} \tag{2.45}$$

For the special case $n = 1$ we define the current as the polarization vector with the corresponding helicity, i.e.

$$J^\mu(k^h) \equiv \epsilon_h^\mu(k, r), \tag{2.46}$$

where r is an arbitrary reference vector defining the direction of quantization of the spin. The n -gluon color-ordered amplitude $\mathcal{A}_n(k_1^{h_1}, \dots, k_n^{h_n})$ can be found by contracting the $(n-1)$ -point off-shell current $\mathcal{J}_n^\mu(k_1^{h_1}, \dots, k_{n-1}^{h_{n-1}})$ with the last polarization vector $\epsilon_{h_n}^\mu(k_n, r_n)$, i.e.

$$\mathcal{A}_n(k_1^{h_1}, \dots, k_n^{h_n}) = \lim_{p_{1,n-1}^2 \rightarrow 0} ip_{1,n-1}^2 \mathcal{J}_n(k_1^{h_1}, \dots, k_{n-1}^{h_{n-1}}) \cdot \epsilon_{h_n}(k_n, r_n). \tag{2.47}$$

Notice that, especially in numerical applications, instead of multiplying by $ip_{1,n-1}^2$ and taking the on-shell limit $p_{1,n-1} \rightarrow 0$ one can simply omit the multiplication by the propagator $-i/p_{1,n-1}^2$ in the computation of the last current.

Even though it is not very practical for analytic calculations (compared e.g. to BCFW), the Berends-Giele recursion is very efficient for numerical computations, especially if one builds the currents from lower to higher points and caches them for later use. It should also be clear that the method is very general and can be extended to any gauge theory with any kind of massless or massive particle (in this case, the free index of a current can also be a spinor index), by defining for each vertex a suitable contraction of the lower-point currents from the Feynman rules.

2.4.2 BCFW recursion

While Berends-Giele recursion builds the amplitude from lower points off-shell objects, BCFW recursion achieves a similar effect using only on-shell quantities. Even though it is not numerically as efficient as the former for high multiplicities, BCFW is significantly easier to apply to analytic calculations. Moreover, it is also particularly interesting from a theoretical point of view. Indeed it shows that a tree-level amplitude, being an analytic function of the kinematic invariants, can be determined by its singularity structure on the complex plane. This can in turn be accessed by putting internal propagators on-shell.

Here we start from a slightly more general case. We consider a color-ordered amplitude $\mathcal{A}_n(k_1, \dots, k_n)$, where the external particles can either be gluons, quarks or scalars, with no assumption about their mass. We pick two external particles i and j and we build a null vector η^μ such that

$$k_i \cdot \eta = k_j \cdot \eta = \eta^2 = 0. \quad (2.48)$$

In the general case, we have two linearly independent choices for η . Using the spinor-helicity formalism (see Appendix A) we can write them as

$$\eta^\mu = \frac{\langle e_1 \gamma^\mu e_2 \rangle}{2}, \quad \eta^\mu = \frac{\langle e_2 \gamma^\mu e_1 \rangle}{2}, \quad (2.49)$$

where e_1 and e_2 are massless linear combinations of k_i and k_j . Notice that η^μ is complex. We thus define two new *shifted* momenta \hat{k}_i and \hat{k}_j ,

$$\hat{k}_i^\mu \equiv k_i^\mu + z \eta^\mu, \quad \hat{k}_j^\mu \equiv k_j^\mu - z \eta^\mu, \quad (2.50)$$

as functions of the complex variable z . These new vectors satisfy the same on-shell constraints as the original two, as well as momentum conservation,

$$\hat{k}_i^2 = k_i^2, \quad \hat{k}_j^2 = k_j^2, \quad \hat{k}_i + \hat{k}_j = k_i + k_j. \quad (2.51)$$

This implies that we can consistently define the on-shell amplitude $\mathcal{A}(z)$ as

$$\mathcal{A}(z) = \mathcal{A}(k_1, \dots, \hat{k}_i, \dots, \hat{k}_j, \dots, k_n), \quad (2.52)$$

from the analytic continuation of $\mathcal{A}(k_1, \dots, k_n)$ taken as an analytic function of the invariants built from its momenta. More in detail, $\mathcal{A}(z)$ is thus a rational function of z and for the special value $z = 0$ we recover the original amplitude. If spinors and polarization vectors are present, we should be slightly more careful on how we define the shifted momenta in order to avoid branch cuts in z , but we will deal with this when considering the special case of an all-gluon amplitude.

Now we consider the integral

$$\oint \frac{dz}{2\pi i z} i\mathcal{A}(z) \quad (2.53)$$

taken around a large circle in the complex plane. If $\mathcal{A}(z) \rightarrow 0$ as $z \rightarrow \infty$ (the validity of this assumption will be briefly discussed later), then the integral vanishes. By rewriting it as the sum of its residues at the corresponding poles we have

$$\sum \text{Res} \left(\frac{\mathcal{A}(z)}{z} \right) = 0. \quad (2.54)$$

There is a pole in $z = 0$ whose residue is simply the original amplitude $i\mathcal{A} = i\mathcal{A}(z = 0)$. Further poles appear because of the dependence on z of the denominators of the propagators. The only propagators depending on z are the ones which factorize the amplitude $\mathcal{A}(z)$ by separating the particles i and j . Any propagator of this form splits the external particles in two groups

- $(k_a, k_{a+1}, \dots, \hat{k}_i, \dots, k_b)$ on its left
- $(k_{b+1}, \dots, \hat{k}_j, \dots, k_{a-1})$ on its right.

The momentum flowing in the propagator before the shift is $p_{a,b}^\mu$, defined according to Eq. (2.44). After the z -dependent shift, this becomes equal to $\hat{p}_{a,b}^\mu$ given by

$$\hat{p}_{a,b}^\mu = p_{a,b}^\mu + z\eta^\mu. \quad (2.55)$$

The shifted amplitude $\mathcal{A}(z)$ has a pole for the value of $z = \bar{z}$ which puts $\hat{p}_{a,b}^\mu$ on-shell ($\hat{p}_{a,b}^2 = m^2$), which is

$$z = \bar{z} = -\frac{p_{a,b}^2 - m^2}{2p_{a,b} \cdot \eta} \quad (2.56)$$

where m is the mass of the particle running in the propagator. Since the numerator of the propagator can be written as a sum over polarizations h , the residue of this pole is

$$\begin{aligned} \text{Res} \left(\frac{i\mathcal{A}(z)}{z} \right)_{z=\bar{z}} &= - \sum_h \left(i\mathcal{A}(k_a, \dots, \hat{k}_i, \dots, k_b, -\hat{p}_{a,b}^h) \frac{i}{p_{a,b}^2 - m^2} \right. \\ &\quad \left. \times i\mathcal{A}(\hat{p}_{a,b}^h, k_{b+1}, \dots, \hat{k}_j, \dots, k_{a-1}) \right), \end{aligned} \quad (2.57)$$

where all the z -dependent quantities are evaluated at $z = \bar{z}$.

By repeating the procedure on all the propagators which split the external particles in two partitions, Eq. (2.54) gives thus an expression for the original partial amplitudes $\mathcal{A}(k_1, \dots, k_n)$ in terms of on-shell amplitudes with a lower number of external legs and shifted momenta,

$$\begin{aligned} i\mathcal{A}(k_1, \dots, k_n) &= \sum_{\text{partitions } r} \sum_h \left(i\mathcal{A}(k_{a_r}, \dots, \hat{k}_i, \dots, k_{b_r}, -\hat{p}_{a_r, b_r}^h) \frac{i}{p_{a_r, b_r}^2 - m_r^2} \right. \\ &\quad \left. \times i\mathcal{A}(\hat{p}_{a_r, b_r}^h, k_{b_r+1}, \dots, \hat{k}_j, \dots, k_{a_r-1}) \right). \end{aligned} \quad (2.58)$$

Here the first sum goes over all the partitions r which separate the particles i and j and each addend of this sum has to be evaluated at the respective pole $z = \bar{z}$. The result is quite

remarkable, for several reasons. The formula only involves on-shell scattering amplitudes evaluated at special (and complex) values of the momenta. One can thus start from 3-point amplitudes, which can be easily worked out from 3-point vertexes, and from these build all higher point amplitudes. In particular, in theories like QCD, the presence of 4- or higher-point vertexes is irrelevant and all the information is already contained in the 3-point interactions, provided that $\mathcal{A}(z)$ goes to zero at the boundary conditions.

Now we will give further details about all-gluon amplitudes in QCD. In the presence of gluons (and also quarks) the amplitude should be seen as an analytic function of the spinors $|k_i\rangle$ and $|k_i]$, rather than a function of the components of the external momenta. In this way, one can directly define the shifts on the spinors. With the choice of η

$$\eta^\mu = \frac{\langle k_1 \gamma^\mu k_2 \rangle}{2}, \quad (2.59)$$

we define the shifts

$$\begin{aligned} \hat{k}_i] &= \hat{k}_i] + z k_j], & \hat{k}_i\rangle &= k_i\rangle, \\ \hat{k}_j\rangle &= k_j\rangle - z k_i\rangle, & \hat{k}_j] &= k_j]. \end{aligned} \quad (2.60)$$

One can easily check that this gives the same shifted momenta as Eq. (2.50), while generating an amplitude $\mathcal{A}(z)$ which is a rational function of z . The only remaining issue is thus making sure that $\mathcal{A}(z) \rightarrow 0$ at $z \rightarrow \infty$. One can show that this is always the case for a proper choice of the external particles i and j . In particular, for all-gluon amplitudes, one should avoid a choice of i and j with helicities $(h_i, h_j) = (-, +)$, assuming all the particles incoming (otherwise the helicity should be reversed). One can easily extend everything to the presence of quarks, with the same helicity choice to be avoided and the additional exception that two external quarks i and j cannot be chosen if they are on the same fermion line. The method can be extended to theories with massive fermions, but in this case the choice of helicity and momentum shift are much more complicated.

BCFW recursion is a first example of how one can exploit the analytic properties of scattering amplitudes in order to efficiently compute them. In particular the recursion relies on their evaluation at (complex) values of the external momenta for which an internal propagator goes on-shell. These ingredients will also be important on the methods we present in this thesis for the computation of loop amplitudes, even though they will be used within a different framework.

2.5 Loop amplitudes

Loop amplitudes are sums of Feynman diagrams containing loops. While in a tree-level amplitude the momentum flowing in every propagator is a combination of the external momenta, in a amplitude with ℓ loops there are ℓ momenta which are not fixed by momentum conservation. Loop diagrams correspond to quantum effects where all the values of the loop

momenta interfere and contribute to the final result. The amplitude is therefore the result of an integration over the components of the loop momenta.

As we recalled in Section 2.3, in *dimensional regularization* the integration is performed in d dimensions. Several variants of dimensional regularization exist, which differ for the treatment of the external momenta and internal gluon states propagating in the loop. In this thesis we will use regularization schemes where the external legs are purely four-dimensional. These include *dimensional reduction* scheme and the *t'Hooft–Veltman* scheme. The former treats internal states as four-dimensional, while the latter treats them as d -dimensional keeping thus the dependence on d in the numerator of the integrand. A more detailed comparison of different regularization schemes can be found in Ref. [120] and references therein.

As one can see from the Feynman rules of QFT, the integrand of a loop amplitude can be written as a rational function of the components of the loop momenta. In this thesis we will analyze the analytic and algebraic properties of the integrands in order to develop efficient and general methods for the evaluation of the respective integrals.

Chapter 3

Integrand reduction of loop amplitudes

Integrand reduction methods for the computation of scattering amplitudes have been developed for one-loop diagrams [40, 41] and recently extended to higher loops [91–95]. These methods use the knowledge of the analytic and algebraic structure of loop integrands in order to rewrite scattering amplitudes as linear combinations of Master Integrals. In this chapter we present a general framework for the integrand decomposition of loop amplitudes which is valid at any loop order. It can be applied to any amplitude in Quantum Field Theory, regardless of the number of external legs or the complexity of the integrands.

The integrand of a Feynman diagram is a rational function of the components of the loop momenta, namely a polynomial numerator sitting over a set of quadratic loop denominators corresponding to internal propagators. Any integrand of this form can be written as a combination of fundamental, irreducible contributions. These contributions are integrands characterized by a subset of the original loop denominators and an *irreducible polynomial residue* for numerator. The latter is a polynomial whose terms cannot be written as combinations of its loop denominators (hence, it cannot be further simplified via a partial fraction decomposition). The residues have a universal process-independent parametric form in terms of unknown coefficients. The actual value of these coefficients is process-dependent and it is a function of the kinematics.

In Ref.s [93, 94] the determination of the residues of the integrand decomposition has been formulated as a problem of *multivariate polynomial division*, and solved at any loop order using simple techniques of algebraic geometry (the basic concepts of algebraic geometry we will use in this thesis are illustrated in Appendix B). The most general parametric form of a residue is the most general remainder of a polynomial division modulo the ideal generated by the corresponding subset of denominators. This formulation of the problem allows to easily derive well known one-loop results and extend them to any loop order.

After integration, most of the terms appearing in the integrand decomposition vanish

(they are called *spurious*). The non-spurious terms give instead the Master Integrals of the integrand reduction. The amplitude is therefore a linear combination of Master Integrals. The coefficients of this linear combination can be identified with a subset of the coefficients which parametrize the residues of the integrand reduction. The reduction of a scattering amplitude to Master Integrals can therefore be reduced to the problem of performing a polynomial fit of the residues. In Section 3.3 we describe two general approaches for performing this fit. The first is the traditional method used in the one-loop case [40], which consists in evaluating the integrand on values of the loop momentum such that some denominators vanish (also known as *multiple cuts*) and can be referred to as *fit-on-the-cut* approach. The second, which we call *divide-and-conquer approach* [95], is a more general algorithm based on purely algebraic operations such as the multivariate polynomial division.

3.1 Integrand reduction via multivariate polynomial division

An arbitrary ℓ -loop amplitude in $d = 4 - 2\epsilon$ dimensions is a sum of contributions \mathcal{M} of the form

$$\mathcal{M} = \int d^d \bar{q}_1 \cdots d^d \bar{q}_\ell \mathcal{I}_{i_1 \dots i_n}, \quad \mathcal{I}_{i_1 \dots i_n} \equiv \frac{\mathcal{N}_{i_1 \dots i_n}}{D_{i_1} \cdots D_{i_n}}, \quad (3.1)$$

where i_1, \dots, i_n are (not necessarily distinct) indices labeling loop propagators. The numerator \mathcal{N} and the denominators D_i of the integrand $\mathcal{I}_{i_1 \dots i_n}$ are polynomials in the components of the loop momenta \bar{q}_i . The d -dimensional loop momenta can be split into a four-dimensional part q_i and a (-2ϵ) -dimensional part $\vec{\mu}_i$,

$$\bar{q}_i = q_i + \vec{\mu}_i, \quad \bar{q}_i \cdot \bar{q}_j = q_i \cdot q_j - \mu_{ij} \quad (3.2)$$

where μ_{ij} is the euclidean scalar product of the extra-dimensional components, i.e. $\mu_{ij} \equiv \vec{\mu}_i \cdot \vec{\mu}_j$. Therefore, the integrand will be a rational function of

$$n_v = 4l + \frac{l(l+1)}{2} \quad (3.3)$$

loop coordinates, namely the components of the four-dimensional vectors q_i in terms of a basis and the extra-dimensional coordinates μ_{ij} . In the following we generically refer to these variables as $\mathbf{z} = \{z_1, \dots, z_{n_v}\}$. The loop denominators are quadratic polynomials and their general form is

$$D_i = \left(\sum_j \alpha_j \bar{q}_j + p_i \right)^2 - m_i^2, \quad \text{with } \alpha_j = 0, \pm 1, \quad (3.4)$$

where p_i is a linear combination of external momenta and m_i are the masses of the particles running in the loop (in general m_i can be complex, if the particle has a non-zero width).

Let $P[\mathbf{z}]$ be the ring of all polynomials in the coordinates \mathbf{z} . Every set of indices $\{i_1, \dots, i_n\}$ defines the ideal

$$\mathcal{J}_{i_1 i_2 \dots i_n} \equiv \langle D_{i_1}, \dots, D_{i_n} \rangle = \left\{ \sum_{k=1}^n h_k(\mathbf{z}) D_{i_k}(\mathbf{z}) : h_k(\mathbf{z}) \in P[\mathbf{z}] \right\}. \quad (3.5)$$

The goal of the integrand reduction is finding a decomposition of the form

$$\mathcal{I}_{i_1 \dots i_n} \equiv \frac{\mathcal{N}_{i_1 \dots i_n}}{D_{i_1} \dots D_{i_n}} = \sum_{k=0}^n \sum_{\{j_1 \dots j_k\}} \frac{\Delta_{j_1 \dots j_k}}{D_{j_1} \dots D_{j_k}} \quad (3.6)$$

where the *residues* $\Delta_{j_1 \dots j_k}$ are *irreducible* polynomials, i.e. polynomials which contain no contribution belonging in the corresponding ideal $\mathcal{J}_{i_1 \dots i_n}$. The second sum on the r.h.s. of the previous equation runs over all the multisubsets¹ of $\{i_1 \dots i_n\}$. We can associate to any residue $\Delta_{j_1 \dots j_k}$ a *multiple cut*, defined as the system of equations which puts on-shell the corresponding loop propagators, i.e. $D_{j_1} = \dots = D_{j_k} = 0$. Eq. (3.6) can also be cast as a decomposition of the numerator in terms of residues and denominators,

$$\mathcal{N}_{i_1 \dots i_n} = \sum_{k=0}^n \sum_{\{j_1 \dots j_k\}} \Delta_{j_1 \dots j_k} \prod_{h \in \{i_1 \dots i_n\} \setminus \{j_1 \dots j_k\}} D_h. \quad (3.7)$$

The numerator \mathcal{N} of the integrand can be decomposed by performing the *multivariate polynomial division* modulo a Gröbner basis $\mathcal{G}_{i_1 \dots i_n}$ of $\mathcal{J}_{i_1 \dots i_n}$ as

$$\begin{aligned} \mathcal{N}_{i_1 \dots i_n} &= \mathcal{Q}_{i_1 \dots i_n} + \Delta_{i_1 \dots i_n} \\ &= \sum_{k=1}^n \mathcal{N}_{i_1 \dots i_{k-1} i_{k+1} \dots i_n} D_{i_k} + \Delta_{i_1 \dots i_n} \end{aligned} \quad (3.8)$$

in terms of a quotient $\mathcal{Q}_{i_1 \dots i_n}$ and the remainder $\Delta_{i_1 \dots i_n}$. The properties of Gröbner bases ensure that the remainder is irreducible, therefore it is identified with the residue of the multiple cut $D_{i_1} = \dots = D_{i_n} = 0$, as suggested by the notation. The quotient $\mathcal{Q}_{i_1 \dots i_n}$ belongs instead to the ideal $\mathcal{J}_{i_1 \dots i_n}$, hence in the last equality of Eq. (3.8) it has been written as a combination of denominators. Substituting Eq. (3.8) in Eq. (3.1), we obtain the recursive formula [94, 95]

$$\mathcal{I}_{i_1 \dots i_n} = \sum_{k=1}^n \mathcal{I}_{i_1 \dots i_{k-1} i_{k+1} \dots i_n} + \frac{\Delta_{i_1 \dots i_n}}{D_{i_1} \dots D_{i_n}}. \quad (3.9)$$

Eq. (3.9) expresses a given integrand in terms of an irreducible residue sitting over its denominators and a sum of integrands corresponding to sub-diagrams with fewer loop propagators. Hence, the recursive application of this formula ultimately yields the full decomposition of any integrand in terms of irreducible residues and denominators, as in Eq. (3.6).

The existence of such a recursive formula proves that the integrand decomposition, originally proposed for one-loop amplitudes, can be extended at any number of loops and the most general parametrization of a residue $\Delta_{i_1 \dots i_n}$ can be identified with the most general remainder of a polynomial division modulo the Gröbner basis $\mathcal{G}_{i_1 \dots i_n}$.

¹A multiset is a set where members can have multiplicity. As an example, $\{1,1,2,2,2,3\}$ and $\{1,1,2,3\}$ are two different multisets of integer numbers, which differ for the multiplicity of the element 2, and the latter is a multisubset of the former.

As mentioned above, the universal parametric form of a residue $\Delta_{j_1 \dots j_k}$ is independent of the process and only depends on the topology of its sub-diagram, i.e. on the set of denominators D_{j_1}, \dots, D_{j_k} . As we already stated, our derivation of Eq. (3.9) implies that the most general form of $\Delta_{j_1 \dots j_k}$ is the most general remainder of a polynomial division modulo the Gröbner basis of the ideal $\mathcal{J}_{j_1 \dots j_k}$. Using the notion of *quotient rings* (see Appendix B.2), one can equivalently identify the parametric form of $\Delta_{j_1 \dots j_k}$ with the most general polynomial in the quotient ring $P[\mathbf{z}]/\mathcal{J}_{j_1 \dots j_k}$. This can in principle contain an unlimited number of terms, but it can always be constrained to have a finite number of them by fixing a limit to the degree of the monomials of the residues. Once the maximum rank of a residue has been fixed, its most general parametrization can thus easily be found by means of computer algebra systems such as SINGULAR [121] and MACAULAY2 [122].

3.2 Two results from algebraic geometry

The formulation of the problem of the integrand reduction of loop amplitudes in terms of basic concepts of algebraic geometry allows to prove two useful results, which we refer to as the *reducibility criterion* and the *maximum-cut theorem* respectively, which we proved in Ref. [94].

The *reducibility criterion* is a direct consequence of *Hilbert's weak Nullstellensatz* (see Appendix B.1). We first notice from Eq. (3.8) and (3.9) that a numerator $\mathcal{N}_{i_1 \dots i_n}$ is completely reducible when the residue $\Delta_{i_1 \dots i_n}$, i.e. the remainder of the polynomial division modulo the Gröbner basis generated by its denominators, vanish. The properties of Gröbner bases imply that this happens if and only if $\mathcal{N}_{i_1 \dots i_n}$ belongs to the ideal generated by the loop denominators. There are cases where this ideal coincides with the whole polynomial ring $P[\mathbf{z}]$, and therefore any integrand $\mathcal{I}_{i_1 \dots i_n}$ over this set of denominators is reducible, independently of the explicit form of the numerator. Hilbert's weak Nullstellensatz implies that this happens if and only if the system of equations of the corresponding multiple cut $D_{i_1} = \dots = D_{i_n} = 0$ has no solution. We have thus proved the following

Proposition. *Reducibility criterion. If a multiple cut $D_{i_1} = \dots = D_{i_n} = 0$ has no solution, then any integrand $\mathcal{I}_{i_1 \dots i_n}$ over this set of loop denominators is reducible.*

The hypothesis of the reducibility criterion is satisfied when the number of independent loop denominators is higher than the number n_v of loop variables (given in Eq. (3.3) for amplitudes in dimensional regularization). In this case, the multiple cut $D_{i_1} = \dots = D_{i_n} = 0$ is an over-determined system of equations with no solution and the integrand is guaranteed to be reducible. When all the loop denominators are distinct, this typically happens whenever $n > n_v$, with n being the total number of loop propagators. This will have important consequences, especially for the fit-on-the-cut approach we will discuss in Section 3.3.1.

A second important result is the so-called *maximum-cut theorem*. A *maximum-cut* is a

multiple cut where the number of (independent) loop propagators is equal to the number of loop coordinates. In this case, the multiple cut is a system of n_v equations in n_v unknowns and therefore it will have, in non-special cases, a finite number n_s of distinct solutions. With this definition, one can prove the following

Theorem. Maximum-cut. *If a maximum-cut has a finite number n_s of solutions, each with multiplicity one, the corresponding residue is a polynomial parametrized by n_s coefficients. This polynomial admits an univariate representation of degree $n_s - 1$.*

Proof. Let $\mathcal{J} = \mathcal{J}_{j_1 \dots j_{n_v}}$ be the ideal generated by the denominators defining the maximum-cut $D_{j_1} = \dots = D_{j_{n_v}} = 0$, and $\Delta_{j_1 \dots j_{n_v}}$ the corresponding residue. The solutions $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(n_s)}$ have the form

$$\mathbf{z}^{(k)} = (z_1^{(k)}, \dots, z_{n_v}^{(k)}), \quad \text{for } k = 1, \dots, n_s.$$

Since the solutions are all distinct, up to a change of variables we can assume the last coordinate $z_{n_v}^{(k)}$ of all solutions to be different.

Since the ideal \mathcal{J} defined by the maximum cut is zero-dimensional (i.e. it has a finite number of solutions) we can use the results collected in Appendix B.3. In particular, because the multiplicity of each multiple-cut solution is 1, the Proposition reviewed in Appendix B.3 implies that the ideal \mathcal{J} is radical. Therefore, if we use the *lexicographical order* with $z_1 > \dots > z_{n_v}$, the hypotheses of the *Shape lemma* are satisfied. This implies that the dimension of the quotient ring $P[\mathbf{z}]/\mathcal{J}$ is finite and the monomials $1, z_{n_v}, z_{n_v}^2, \dots, z_{n_v}^{n_s-1}$ form a basis. Since the most general parametrization of $\Delta_{j_1 \dots j_k}$ coincides with the most general polynomial in the quotient ring $P[\mathbf{z}]/\mathcal{J}$, this completes the proof. \square

The *Maximum-cut Theorem* will be particularly important for the *fit-on-the-cut approach* for the reduction, which we will describe in Section 3.3.1. In Fig. 3.1 we show examples of maximum cuts for one-, two- and three-loop topologies, where we explicitly checked that the maximum cut theorem is satisfied. At one loop in $d = 4 - 2\epsilon$ dimensions, a maximum cut has $n_v = 5$ loop propagators. The corresponding cut equations $D_{j_1} = \dots = D_{j_5} = 0$ can be shown to have only one solution. Hence, its residue can be parametrized by one coefficient, i.e. by a constant. In $d = 4$ dimensions instead, we only have four loop coordinates and thus 5-point diagrams are reducible (for the reducibility criterion) and a maximum cut is a 4-ple cut. In four dimensions, quadruple cuts have two solutions and their residue can thus be parametrized by two coefficients. Similarly, at two and three loops, four-dimensional maximum cuts involve eight and twelve loop propagators respectively. The number of solutions, in these cases, depends on the topology of the diagram, but it is always equal to the number of unknown coefficients appearing in the parametrization of the residues. When the *lexicographical monomial order* is used, this parametrization is an univariate monomial, however we will generally use the *degree reverse lexicographical monomial order* (see Appendix B.2), hence these residues will be multivariate but have in turn a lower degree, significantly simplifying our computations.

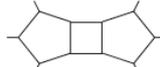
diagram	Δ	n_s	diagram	Δ	n_s
	c_0	1		$c_0 + c_1 z$	2
	$\sum_{i=0}^3 c_i z^i$	4		$\sum_{i=0}^3 c_i z^i$	4
	$\sum_{i=0}^7 c_i z^i$	8		$\sum_{i=0}^7 c_i z^i$	8

Figure 3.1: Examples of maximum cuts. The first on the left is a one-loop maximum cut in $(4 - 2\epsilon)$ dimensions. The others are one-, two- and three-loop maximum cuts in four dimensions. For each cut we show the number of solutions and the parametric residue, assuming a *lexicographical monomial order*.

3.3 Reduction techniques

Within the framework we described in the previous section, the reduction of a scattering amplitude to Master Integrals has been reduced to the problem of performing a polynomial fit of the residues of the integrand decomposition. The traditional way of performing this fit is by evaluating the integrand on *multiple cuts*, i.e. on values of the loop momentum such that some denominators vanish. This is the so-called *fit-on-the-cut* approach, which we will review in Section 3.3.1. One can also perform the full integrand decomposition of any multi-loop amplitude by means of purely algebraic operations, within what we call the *divide-and-conquer approach* [95], which will be illustrated in Section 3.3.2. In this case, the residues are generated as the remainders of multivariate polynomial divisions recursively applied to the numerator of the diagram and the ones of the sub-diagrams defined by the quotients of former divisions. Because of its wider range of applicability we may consider the latter a more general method for the integrand decomposition of loop integrals.

3.3.1 Fit-on-the-cut approach

The fit-on-the-cut approach was proposed in Ref. [40] for one-loop amplitudes, and first applied to the integrand reduction of higher-loop amplitudes in Ref. [91]. It consists in finding the values of the coefficients which parametrize the residues in Eq. (3.6) by evaluating the integrand on values of the loop momenta such that a subset of loop denominators vanish. On such values, the only non-vanishing contributions to the decomposition of the numerator in Eq. (3.7) are the ones coming from the residue identified by those denominators and from the ones which have the corresponding loop propagators as a sub-diagram.

In this subsection we will assume that all the denominators D_j are distinct. This can always be assumed to be the case at one-loop, but not at two or higher loops. The problem of the reduction of diagrams with arbitrary powers of loop denominators can be addressed with *divide-and-conquer* method, which we will describe in the next subsection. An alternative method which can be used in the presence of double powers of loop propagators has been proposed in Ref. [123].

With this hypothesis we can assume $\{i_1, \dots, i_n\} \equiv \{1, \dots, n\}$ and Eq. (3.7) can be rewritten as

$$\mathcal{N}_{1\dots n} = \sum_{m=0}^n \sum_{\{i_1 \dots i_m\}} \Delta_{i_1 \dots i_m} \prod_{h \neq i_1, \dots, i_m} D_h. \quad (3.10)$$

By evaluating Eq. (3.10) on a m -ple cut $D_{j_1} = \dots = D_{j_m} = 0$ one gets

$$\Delta_{j_1 \dots j_m} = \frac{\mathcal{N}_{1\dots n}}{\prod_{h \notin \{j_1 \dots j_m\}} D_h} - \sum_{m'=m+1}^n \sum_{\{i_1 \dots i_{m'}\}} \frac{\Delta_{i_1 \dots i_{m'}}}{\prod_{h \in \{i_1 \dots i_{m'}\} \setminus \{j_1 \dots j_m\}} D_h}, \quad (3.11)$$

where the second sum on the r.h.s. runs over all the partitions which satisfy $\{i_1, \dots, i_{m'}\} \supset \{j_1, \dots, j_m\}$, i.e. those whose denominators are a superset of the cut denominators. In other words, a residue is equal to the numerator evaluated on the corresponding multiple cut and divided by the uncut denominators, after subtracting all the non-vanishing contributions from higher-point residues.

This suggests a *top-down* strategy for the determination of the coefficients of all the residues. The first step consists in using Eq. (3.11) on a *maximum-cut*, i.e. with m equal to the number n_v of loop coordinates. The *reducibility criterion* and the *maximum-cut theorem* (see Section 3.2) play here an important role. The *reducibility criterion* ensures that all the higher-point residues vanish, because the corresponding cut is an over-determined system of equations. Hence Eq. (3.11) on a maximum cut becomes

$$\Delta_{j_1 \dots j_{n_v}} = \frac{\mathcal{N}_{1\dots n}}{\prod_{h \notin \{j_1 \dots j_{n_v}\}} D_h}, \quad (3.12)$$

where on the l.h.s. we can substitute the most general parametric form of the residue, which can be found a priori as explained in Section 3.1. In particular, the *maximum-cut theorem* says that the number of coefficients which parametrize the residue $\Delta_{j_1 \dots j_{n_v}}$ is equal to the number n_s of solutions of the cut. Hence, by evaluating Eq. (3.12) on these solutions, with the parametric form of $\Delta_{j_1 \dots j_{n_v}}$ on the l.h.s., we obtain an $n_s \times n_s$ linear system of equations for the coefficients.

The next step is the determination of residues corresponding to next-to-maximum cuts. As usual, the parametric form of the residue is easily known and therefore one must only find the unknown process-dependent coefficients which appear in such parametrization. In this case the system $D_{j_1} = \dots = D_{j_m} = 0$ is under-determined and thus, in non-special kinematic points, it has an infinite number of solutions. By applying Eq. (3.11), we need to

subtract from the integrand the non-vanishing maximum-cuts computed in the previous step and evaluate it on a subset of the solutions of this cut, obtaining a system of equations for the unknown coefficients.

With this procedure, one can thus recursively apply Eq. (3.12) on m -ple cuts. At each step, one must subtract from the integrand all the non-vanishing contributions from higher-point residues with more than m loop propagators. These are known as *subtractions at the integrand level*. A multiple cut can thus be seen as a *projector* that, once applied to the integrand, isolates the contribution of the corresponding residue. An alternative way of carrying out the reduction on the cuts for the one-loop case is presented in Chapter 4.

The fit-on-the-cut strategy has been very successful for the computation of one-loop amplitudes. However, at two or higher loops it has several limitations. A first limitation is that, as stated before, it cannot be applied to processes where the loop denominators are not all distinct (or equivalently, where higher powers of loop propagators are present). This can be clearly seen from Eq. (3.11) where on the r.h.s. we divide by the uncut denominators, which therefore must not be equal to the (vanishing) cut denominators. Another limitation is that the solutions of multiple cuts, and in particular a parametrization of the algebraic variety $\mathcal{V}(\mathcal{J}_{j_1 \dots j_m})$ defined by such solutions, can be difficult to find for a higher-loop topology, especially for lower-point cuts. Such parametrization is however needed in order to fit *all* the unknown coefficients which parametrize these residues. A third potential limitation is due to the fact that there is no guarantee that, by evaluating the integrand on values of the loop coordinates \mathbf{z} belonging to the variety of solutions $\mathcal{V}(\mathcal{J}_{j_1 \dots j_m})$, one can find the values of all the coefficients of the residue $\Delta_{j_1 \dots j_m}$. Indeed, as a consequence of *Hilbert's Nullstellensatz*, the set of polynomials which vanish on $\mathcal{V}(\mathcal{J}_{j_1 \dots j_m})$ is the *radical* $\sqrt{\mathcal{J}_{j_1 \dots j_m}}$ of the ideal defined by the loop denominators. Since in general $\mathcal{J}_{j_1 \dots j_m} \subseteq \sqrt{\mathcal{J}_{j_1 \dots j_m}}$, by evaluating the integrand on values $\mathbf{z} \in \mathcal{V}(\mathcal{J}_{j_1 \dots j_m})$ we are only guaranteed to be able to fit a polynomial p when $p \in P[\mathbf{z}]/\sqrt{\mathcal{J}_{j_1 \dots j_m}}$. However $\Delta_{j_1 \dots j_m} \in P[\mathbf{z}]/\mathcal{J}_{j_1 \dots j_m}$ which is a larger space of polynomials than $P[\mathbf{z}]/\sqrt{\mathcal{J}_{j_1 \dots j_m}}$. Therefore, the fit-on-the-cut approach will work only if the ideals defined by the loop denominators of the diagrams and their sub-diagrams are radical, i.e. if for every residue $\Delta_{j_1 \dots j_m}$ we have $\sqrt{\mathcal{J}_{j_1 \dots j_m}} = \mathcal{J}_{j_1 \dots j_m}$. However, none of these limitations are present in the *divide-and-conquer* approach that we are going to present in the next subsection.

3.3.2 Divide-and-conquer approach

The direct application of the integrand reduction formula of Eq. (3.9) on the numerator of an ℓ -loop graph allows to perform the integrand decomposition algebraically by successive polynomial divisions, within what we call the *divide-and-conquer* approach, which has been presented in Ref. [95]. At each step, the remainders of the divisions are identified with the residues of the corresponding set of denominators, while the quotients become the numerators of the lower-point integrands appearing on the r.h.s. of the formula, allowing thus to iterate

the procedure. In this way, the decomposition of any integrand is obtained analytically, with a finite number of algebraic operations, without requiring the knowledge of the varieties of solutions of the multiple cuts, nor the one of the parametric form of the residues. Explicit applications of this methods will be discussed in Section 6.2.

Since it is based on the same principles used to constructively prove the existence of the integrand decomposition at all loops, the *divide-and-conquer* approach doesn't have the limitations of other methods, and in particular it is not affected by the presence of massive propagators, non planar diagrams, higher powers of loop denominators or higher-rank contributions in the numerator. Therefore, this can be considered a more general integrand reduction algorithm.

3.4 An algebraic implementation of the divide-and-conquer approach

In this Section we describe a semi-automated implementation of the *divide-and-conquer* approach, which we used for some of the computations presented in this thesis. The implementation is a PYTHON package, which uses FORM [124] and MACAULAY2 [122] for the algebraic operations.

The inputs needed by the algorithm are

- the analytic FORM expression of the numerator
- the expressions of the loop denominators
- the set of (not-necessarily distinct) indexes $\{i_1, \dots, i_n\}$ labeling the denominators
- the (possibly cut-dependent) choice of variables z_i for the parametrization of the 4-dimensional part of the loop momenta
- any additional identity between the variables appearing in the computation to be passed to FORM and/or MACAULAY2.

The PYTHON code thus generates all the independent multisubsets of $\{i_1, \dots, i_k\}$ for the original loop indexes and iterates over them. In each iteration, the following operations are performed:

- the parametrization for the loop momenta is substituted in the formula for the numerator (or sub-numerator) $\mathcal{N}_{i_1 \dots i_k}$, which is parsed by FORM checking for all the monomials that appear in the expression
- the Groöbner basis of $\mathcal{G}_{i_1 \dots i_k}$ of the ideal $\mathcal{J}_{i_1 \dots i_k} = \langle D_{i_1}, \dots, D_{i_k} \rangle$ is computed in MACAULAY

- for each monomial appearing in the numerator expression, a polynomial division is performed, where the quotient is then rewritten as a combination of the original denominators D_{i_1}, \dots, D_{i_k}
- the results of the polynomial division are translated into FORM substitutions, the *remainder* is stored as the residue and the quotients will contribute to lower points numerators.

The recursion terminates after the iteration over multisubsets of k denominators, if all the resulting lower $(k - 1)$ -point numerators obtained as sums of quotients vanish.

The result is thus available as a decomposition of the numerator of the form of Eq. (3.7), where algebraic expressions for all the residues have been computed and can be inserted with a set of automatically-generated substitutions in FORM.

This basic implementation, despite being quite simple and minimal, has already been used for some computations which showed the generality of the method and therefore its potential. It is also a proof of concept of the possibility of automating the algebraic integrand reduction at all loops. Furthermore, the combination of FORM, with its capability of handling large expressions, and the features of other Computer Algebra Systems, which are more focused on typical operations of algebraic geometry, seems to be a reasonable and effective choice.

For a systematic usage in more complex computations, several improvements could be implemented. The set of diagrams contributing to a scattering amplitude could be organized in terms of common topologies and sub-topologies, taking into account symmetries of diagrams as well, in order to minimize the number of Gröbner basis to be computed. The possibility of using SINGULAR as an alternative to MACAULAY could also help in improving the performance. A custom implementation of the computation of Gröbner bases or the polynomial division algorithm, tailored for the kind of problems we are dealing with, could also be worth considering. Finally, a default algorithm for the choice of parametrization of the loop momenta could be implemented. Building a full computational framework for the higher-loop integrand reduction is however a complex task which is beyond the purposes of this thesis. Nevertheless, thanks to the ingredients we presented, the path for the automation of higher-loop integrand reduction appears now more clear.

3.5 The one-loop case: OPP decomposition

Integrand reduction methods were first proposed for one-loop amplitudes [40, 41]. In this section, we will present the one-loop case as a special application of the general framework we described in this chapter.

A generic contribution \mathcal{M} to an n -point one-loop amplitude in dimensional regularization can be written as

$$\mathcal{M} = h(\mu_R^2, d) \int d^d \bar{q} \mathcal{I} = h(\mu_R^2, d) \int d^d \bar{q} \frac{\mathcal{N}(\bar{q})}{D_0 \cdots D_{n-1}}. \quad (3.13)$$

In the previous equation, the integrand \mathcal{I} is a *rational function* of the components of the d -dimensional loop momentum \bar{q} , with $d = 4 - 2\epsilon$. The numerator $\mathcal{N}(\bar{q})$ is a process-dependent polynomial in \bar{q} , while the denominators D_i in the one-loop case have the form,

$$D_i = (\bar{q} + p_i)^2 - m_i^2. \quad (3.14)$$

The function h appearing in Eq. (3.13) is a conventional normalization factor given by [125]

$$h(\mu_R^2, d) = h(\mu_R^2, 4 - 2\epsilon) = \frac{\mu_R^{2\epsilon}}{i\pi^{2-\epsilon}} \frac{\Gamma(1 - 2\epsilon)}{\Gamma^2(1 - \epsilon)\Gamma(1 + \epsilon)}, \quad (3.15)$$

as a function of the renormalization scale μ_R^2 and the dimension $d = 4 - 2\epsilon$. The d -dimensional loop momentum \bar{q} can be split into a four-dimensional part q and a (-2ϵ) -dimensional part $\vec{\mu}$ as

$$\bar{q} = q + \vec{\mu}, \quad \bar{q}^2 = q^2 - \mu^2. \quad (3.16)$$

The numerator \mathcal{N} will therefore be a polynomial in the four components of q and the extra-dimensional variable μ^2 .

Since we have five loop coordinates, the *reducibility criterion* (see Section 3.2) implies that every one-loop integrand in dimensional regularization can be decomposed as sum of integrands having five or less loop denominators

$$\mathcal{I} \equiv \frac{\mathcal{N}}{D_0 \cdots D_{n-1}} = \sum_{k=1}^5 \sum_{\{i_1, \dots, i_k\}} \frac{\Delta_{i_1 \cdots i_k}}{D_{i_1} \cdots D_{i_k}}, \quad (3.17)$$

where the second sum on the r.h.s. runs over all the subsets of the denominator indexes $\{0, \dots, n-1\}$ containing k elements.

In order to write down the most general parametric form of the *irreducible residues* $\Delta_{i_1 \cdots i_k}$, we must first make an explicit choice of the four-dimensional variables. This can be done by choosing, for any set of denominators D_{i_1}, \dots, D_{i_k} with $k \leq 5$, a proper four-dimensional basis. As proposed in Ref.s [30, 40, 43, 126] we build a basis of massless momenta $\mathcal{E}^{(i_1 \cdots i_k)} = \{e_1, e_2, e_3, e_4\}$. The first two elements of the basis are linear combinations of two external momenta K_1, K_2 of the sub-diagram identified by the considered set of loop denominators. More explicitly, we define

$$e_1^\mu = \frac{1}{1 - r_1 r_2} (K_1^\mu - r_1 K_2^\mu), \quad e_2^\mu = \frac{1}{1 - r_1 r_2} (K_2^\mu - r_2 K_1^\mu), \quad (3.18)$$

with

$$K_1^\mu = p_{i_1}^\mu - p_{i_k}^\mu, \quad K_2^\mu = p_{i_2}^\mu - p_{i_1}^\mu, \quad r_1 = \frac{K_1^2}{\gamma}, \quad r_2 = \frac{K_2^2}{\gamma},$$

$$\gamma = (K_1 \cdot K_2) \left(1 + \sqrt{1 - \frac{K_1^2 K_2^2}{(K_1 \cdot K_2)^2}} \right).$$

where the momenta p_i were defined in Eq. (3.14). If the sub-diagram has less than two independent external momenta, the remaining ones are substituted by arbitrary reference vectors in the definition of e_1 and e_2 . The momenta e_3 and e_4 are instead chosen to be orthogonal to the first two and can be defined using the spinor notation (see Appendix A) as

$$e_3^\mu = \frac{\langle e_1 \gamma^\mu e_2 \rangle}{2}, \quad e_4^\mu = \frac{\langle e_2 \gamma^\mu e_1 \rangle}{2}. \quad (3.19)$$

They satisfy the property $(e_3 \cdot e_4) = -(e_1 \cdot e_2)$. For subsets of $k = 4$ denominators, we also define the vectors v and v_\perp

$$v^\mu = (e_4 \cdot K_3) e_3^\mu + (e_3 \cdot K_3) e_4^\mu, \quad v_\perp^\mu = (e_4 \cdot K_3) e_3^\mu - (e_3 \cdot K_3) e_4^\mu, \quad (3.20)$$

with $K_3^\mu = p_{i_3}^\mu - p_{i_2}^\mu$. We observe that the vector v_\perp is orthogonal to all the external legs of the sub-diagram identified by the four denominators.

By expanding the four dimensional part q of the loop momentum \bar{q} in the basis $\mathcal{E}^{(i_1 \dots i_k)}$, the numerator and the denominators can be written as polynomials in the coordinates $\mathbf{z} \equiv (z_1, z_2, z_3, z_4, z_5) = (x_1, x_2, x_3, x_4, \mu^2)$,

$$\mathcal{N}(\bar{q}) = \mathcal{N}(q, \mu^2) = \mathcal{N}(x_1, x_2, x_3, x_4, \mu^2) = \mathcal{N}(\mathbf{z}), \quad (3.21)$$

with

$$q^\nu = -p_{i_1}^\nu + \frac{1}{e_1 \cdot e_2} (x_1 e_1^\nu + x_2 e_2^\nu - x_3 e_3^\nu - x_4 e_4^\nu). \quad (3.22)$$

The coordinates x_i can be written as scalar products

$$x_1 = (l_{i_1} \cdot e_2), \quad x_2 = (l_{i_1} \cdot e_1), \quad x_3 = (l_{i_1} \cdot e_4), \quad x_4 = (l_{i_1} \cdot e_3), \quad (3.23)$$

where $l_{i_1} \equiv (q + p_{i_1})$. For $k = 4$ we also consider the alternative expansion of the loop momentum

$$q^\nu = -p_{i_1}^\nu + \frac{1}{e_1 \cdot e_2} (x_1 e_1^\nu + x_2 e_2^\nu) + \frac{1}{v^2} (x_{3,v} v_3^\nu - x_{4,v} v_4^\nu). \quad (3.24)$$

with

$$x_{3,v} = (l_{i_1} \cdot v), \quad x_{4,v} = (l_{i_1} \cdot v_\perp). \quad (3.25)$$

Following the general discussion of Section 3.1, the universal parametric form of the residues $\Delta_{i_1 \dots i_k}$ in a renormalizable theory can be found as the most general remainder of a numerator of the form

$$\mathcal{N}_{i_1 \dots i_k} \equiv \sum_{j_1, j_2, j_3, j_4, j_5} n_{j_1 j_2 j_3 j_4 j_5} x_1^{j_1} x_2^{j_2} x_3^{j_3} x_4^{j_4} \mu^{2j_5}, \quad \text{with } j_1 + j_2 + j_3 + j_4 + 2j_5 \leq r_{\max}, \quad (3.26)$$

where $r_{\max} = k$. In practice, as suggested in Ref. [127], using a different parametrization of the 5-point residues may be more convenient, namely

$$\Delta_{i_1 i_2 i_3 i_4 i_5} = c_0 \mu^2 \quad \text{instead of} \quad \Delta_{i_1 i_2 i_3 i_4 i_5} = c_0,$$

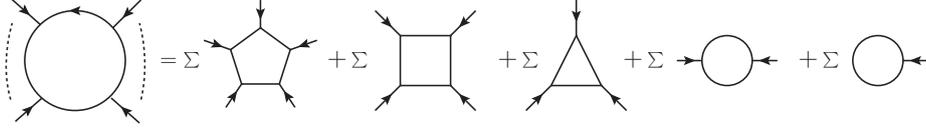


Figure 3.2: Schematic illustration of the one-loop OPP decomposition.

which is allowed because 1 is proportional to μ^2 in the quotient ring $P[\mathbf{z}]/\mathcal{J}_{i_1 i_2 i_3 i_4 i_5}$. The main advantage of this choice is that the 5-point residues will vanish after integration. Besides, with this parametrization, the four-dimensional part of the 4-point residues will coincide with the one which would be obtained with a purely four-dimensional reduction (i.e. with $\bar{q} = q, \mu^2 = 0$). With these choices, the most general parametric form of the residues is [40, 41, 94]

$$\begin{aligned}
\Delta_{i_1 i_2 i_3 i_4 i_5} &= c_0 \mu^2 \\
\Delta_{i_1 i_2 i_3 i_4} &= c_0 + c_1 x_{4,v} + \mu^2 (c_2 + c_3 x_{4,v} + \mu^2 c_4) \\
\Delta_{i_1 i_2 i_3} &= c_0 + c_1 x_4 + c_2 x_4^2 + c_3 x_4^3 + c_4 x_3 + c_5 x_3^2 + c_6 x_3^3 + \mu^2 (c_7 + c_8 x_4 + c_9 x_3) \\
\Delta_{i_1 i_2} &= c_0 + c_1 x_1 + c_2 x_2^2 + c_3 x_4 + c_4 x_4^2 + c_5 x_3 + c_6 x_3^2 + c_7 x_1 x_4 + c_8 x_1 x_3 + c_9 \mu^2 \\
\Delta_{i_1} &= c_0 + c_1 x_2 + c_2 x_1 + c_3 x_4 + c_4 x_3,
\end{aligned} \tag{3.27}$$

where we understand that the unknown coefficients c_j depend on the indexes of the residue (e.g. $c_j = c_j^{(i_1 \dots i_k)}$), while the scalar products x_i and $x_{i,v}$ depend on both the indexes of the residue and the loop momentum q . The decomposition in Eq. (3.17) with parametric residues of Eq. (3.27) is often referred to as the *OPP integrand decomposition*. It is schematically depicted in Fig. 3.2.

The parametrization in Eq. (3.27) can easily be extended to effective and non-renormalizable theories where the rank r of the numerator can be larger than the number n of loop propagators [96]. In the case with $r = n + 1$, such parametrization can be generalized by allowing $r_{\max} = k + 1$ in Eq. (3.26). The result,

$$\begin{aligned}
\Delta_{i_1 i_2 i_3 i_4 i_5}^{(r=n+1)} &= \Delta_{i_1 i_2 i_3 i_4 i_5} \\
\Delta_{i_1 i_2 i_3 i_4}^{(r=n+1)} &= \Delta_{i_1 i_2 i_3 i_4} + c_5 \mu^4 x_{v,4} \\
\Delta_{i_1 i_2 i_3}^{(r=n+1)} &= \Delta_{i_1 i_2 i_3} + \mu^2 (c_{10} x_4^2 + c_{11} x_3^2) + c_{12} x_4^4 + c_{13} x_3^4 + c_{14} \mu^4 \\
\Delta_{i_1 i_2}^{(r=n+1)} &= \Delta_{i_1 i_2} + \mu^2 (c_{10} x_1 + c_{11} x_4 + c_{12} x_3) + c_{13} x_1^3 + c_{14} x_4^3 + c_{15} x_3^3 \\
&\quad + c_{16} x_1^2 x_4 + c_{17} x_1^2 x_3 + c_{18} x_1 x_4^2 + c_{19} x_1 x_3^2 \\
\Delta_{i_1}^{(r=n+1)} &= \Delta_{i_1} + c_5 x_2^2 + c_6 x_1^2 + c_7 x_4^2 + c_8 x_3^2 + c_{10} x_2 x_4 + c_{11} x_2 x_3 \\
&\quad + c_{12} x_1 x_4 + c_{13} x_1 x_3 + c_{14} \mu^2 + c_{15} x_3 x_4,
\end{aligned} \tag{3.28}$$

agrees with the one we first found with a different (and less general) method in Ref. [96].

Most of the terms appearing in Eq. (3.27) are *spurious*, i.e. they vanish after integration and do not contribute to the final result. The amplitude \mathcal{M} can thus be expressed as a linear combination of Master Integrals, corresponding to the non-spurious terms of the integrand decomposition, namely

$$\begin{aligned}
\mathcal{M} = & \sum_{\{i_1, i_2, i_3, i_4\}} \left\{ c_0^{(i_1 i_2 i_3 i_4)} I_{i_1 i_2 i_3 i_4} + c_4^{(i_1 i_2 i_3 i_4)} I_{i_1 i_2 i_3 i_4}[\mu^4] \right\} \\
& + \sum_{\{i_1, i_2, i_3\}} \left\{ c_0^{(i_1 i_2 i_3)} I_{i_1 i_2 i_3} + c_7^{(i_1 i_2 i_3)} I_{i_1 i_2 i_3}[\mu^2] \right\} \\
& + \sum_{\{i_1, i_2\}} \left\{ c_0^{(i_1 i_2)} I_{i_1 i_2} + c_1^{(i_1 i_2)} I_{i_1 i_2}[(q + p_{i_1}) \cdot e_2] \right. \\
& \quad \left. + c_2^{(i_1 i_2)} I_{i_1 i_2}[((q + p_{i_1}) \cdot e_2)^2] + c_9^{(i_1 i_2)} I_{i_1 i_2}[\mu^2] \right\} \\
& + \sum_{i_1} c_0^{(i_1)} I_{i_1}, \tag{3.29}
\end{aligned}$$

where

$$I_{i_1 \dots i_k}[\alpha] \equiv h(\mu_R^2, d) \int d^d \bar{q} \frac{\alpha}{D_{i_1} \dots D_{i_k}}, \quad I_{i_1 \dots i_k} \equiv I_{i_1 \dots i_k}[1]. \tag{3.30}$$

The coefficients of this linear combination can be identified with a subset of the coefficients of the parametric residues in Eq. (3.27). Since all the Master Integrals appearing in Eq. (3.29) are known, the problem of the computation of an arbitrary one-loop amplitude can be reduced to the problem of the determination of the non-spurious coefficients appearing in the parametrization of the residues $\Delta_{i_1 \dots i_k}$.

For the higher-rank case with $r = n + 1$, the integral decomposition is generalized as

$$\begin{aligned}
\mathcal{M}^{(r=n+1)} = & \mathcal{M}^{(r=n)} + \sum_{\{i_1, i_2, i_3\}} c_{14}^{(i_1 i_2 i_3)} I_{i_1 i_2 i_3}[\mu^4] \\
& + \sum_{\{i_1, i_2\}} \left\{ c_{10}^{(i_1 i_2)} I_{i_1 i_2}[\mu^2 (q + p_{i_1}) \cdot e_2] + c_{13}^{(i_1 i_2)} I_{i_1 i_2}[((q + p_{i_1}) \cdot e_2)^3] \right\} \\
& + \sum_{i_1} \left\{ c_{14} I_{i_1}[\mu^2] + c_{15}^{(i_1)} I_{i_1}[((q + p_{i_1}) \cdot e_3)((q + p_{i_1}) \cdot e_4)] \right\}, \tag{3.31}
\end{aligned}$$

where, once again, all the Master Integrals are known [96] (see also Section 4.2.2).

3.5.1 Fit on the cut at one loop

The fit of the unknown coefficients of residues, and thus the Master Integrals, can be efficiently performed using the *fit-on-the-cut approach* we described in Section 3.3.1. This consists in evaluating the numerator of the integrand on *multiple cuts*, i.e. on values of the loop momentum such that a subset of the loop denominators vanish. A residue can be evaluated by putting on-shell the corresponding loop propagators and subtracting from the integrand

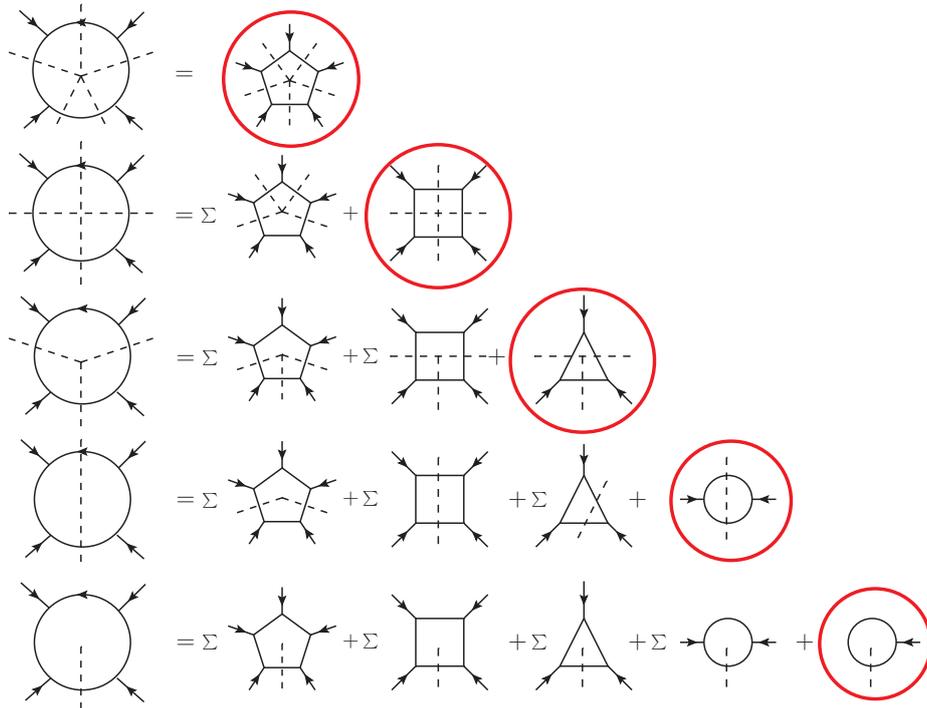


Figure 3.3: Schematic illustration of the fit-on-the-cut reduction technique at one loop.

the non-vanishing contributions coming from higher-point residues. This leads to a top-down algorithm, schematically illustrated in Fig. 3.3, which allows to compute any one-loop amplitude with any number of external legs.

Within semi-numerical computations, the algorithm is usually implemented by sampling the integrand on several solutions of the multiple cuts and subtracting at each step of the reduction all the non-vanishing contributions coming from higher-point residues. This yields a system of equations for the coefficients of each residue. The method is suited for automation and it has been implemented in several codes, some of which are public (e.g. CUTTOOLS [42] and SAMURAI [43]). Its usage within several automated frameworks [44–52] has been particularly successful and produced highly non-trivial phenomenological results.

In this section we give a brief review of this method, as proposed in Ref.s [40, 41, 43], addressing in some detail the computation of 5-, 4-, 3-, 2-, and 1-point residues, also commonly known as *pentagons*, *boxes*, *triangles*, *bubbles* and *tadpoles* respectively. In Chapter 4 we will present an alternative integrand-reduction algorithm, namely the *integrand reduction via Laurent series expansion* [96], which can be numerically more accurate and efficient.

5-point residues A maximum-cut in a $(4 - 2\epsilon)$ -dimensional one-loop amplitude is a 5-ple cut. The corresponding system of equations $D_{i_1} = \dots = D_{i_5} = 0$ has one solution and the residue $\Delta_{i_1 \dots i_5}$, as already observed, can be parametrized by one coefficient, in agreement with the *maximum-cut theorem*. Indeed, using Eq. (3.12) and the parametrization in Eq. (3.27),

on this solution $q = q_s, \mu^2 = \mu_s^2$, the coefficient c_0 of the pentagon can be simply found as

$$c_0 = \frac{1}{\mu_s^2} \frac{\mathcal{N}(q_s, \mu_s^2)}{\prod_{h \neq i_1, \dots, i_5} D_h}. \quad (3.32)$$

4-point residues The most general parametrization of a 4-point residue $\Delta_{i_1 \dots i_4}$ was given in Eq. (3.27), and its coefficients can be found by evaluating the integrand on the solutions of the corresponding quadruple cut [5], i.e. $D_{i_1} = \dots = D_{i_4} = 0$. More in detail, the two coefficients c_0 and c_1 can be easily found on the solutions of a four-dimensional quadruple cut, i.e. with $\bar{q} = q, \mu^2 = 0$. Indeed, in four dimensions we only have four loop coordinates and thus a quadruple cut is a maximum cut. This has two solutions q_+ and q_- , which can be used to fit the two four-dimensional coefficients c_0 and c_1 . In particular, the coefficient c_0 can be shown to be equal to the average of the integrand evaluated on the two solutions, i.e.

$$c_0 = \frac{1}{2} \left(\frac{\mathcal{N}(q)}{\prod_{j \neq i_1, i_2, i_3, i_4} D_j} \Big|_{q=q_+} + \frac{\mathcal{N}(q)}{\prod_{j \neq i_1, i_2, i_3, i_4} D_j} \Big|_{q=q_-} \right), \quad (3.33)$$

while c_1 is proportional to the difference. The computation of the other coefficients requires instead to evaluate the integrand on $(4 - 2\epsilon)$ -dimensional quadruple cuts, which admit an infinite number of solutions $\{(q_s, \mu_s^2)\}$. By sampling the integrand on a subset of these solutions and subtracting the contributions coming from 5-point residues, one obtains a system of equations for the unknown coefficients c_2, c_3 and c_4 , using the decomposition

$$\Delta_{i_1 i_2 i_3 i_4} = \frac{\mathcal{N}(q_s, \mu_s^2)}{\prod_{j \neq i_1, i_2, i_3, i_4} D_j} - \sum_k \frac{\Delta_{i_1 i_2 i_3 i_4 k}}{D_k}, \quad (3.34)$$

valid on a generic quadruple cut.

3-point residues The coefficients of the residues of a generic triangle contribution $\Delta_{i_1 i_2 i_3}$ can be determined by evaluating the integrand on the solutions of the corresponding d -dimensional triple cut [30], i.e. $D_{i_1} = D_{i_2} = D_{i_3} = 0$. On the solutions $\{(q_s, \mu_s^2)\}$ the integrand decomposition reads

$$\Delta_{i_1 i_2 i_3} = \frac{\mathcal{N}(q_s, \mu_s^2)}{\prod_{j \neq i_1, i_2, i_3} D_j} - \sum_{i_4} \frac{\Delta_{i_1 i_2 i_3 i_4}}{D_{i_4}} - \sum_{i_4, i_5} \frac{\Delta_{i_1 i_2 i_3 i_4 i_5}}{D_{i_4} D_{i_5}}, \quad (3.35)$$

hence one can compute the coefficients of the residue by sampling the integrand on a finite subset of these solutions and subtracting the non-vanishing contributions coming from boxes and pentagons.

2-point residues In order to determine the coefficients of the bubble residues $\Delta_{i_1 i_2}$ one can evaluate the integrand on the solutions $\{(q_s, \mu_s^2)\}$ of the corresponding d -dimensional double

cut $D_{i_1} = D_{i_2} = 0$. In this case the integrand decomposition reads

$$\Delta_{i_1 i_2} = \frac{\mathcal{N}(q_s, \mu_s^2)}{\prod_{j \neq i_1, i_2} D_j} - \sum_{i_3} \frac{\Delta_{i_1 i_2 i_3}}{D_{i_3}} - \sum_{i_3, i_4} \frac{\Delta_{i_1 i_2 i_3 i_4}}{D_{i_3} D_{i_4}} - \sum_{i_3, i_4, i_5} \frac{\Delta_{i_1 i_2 i_3 i_4 i_5}}{D_{i_3} D_{i_4} D_{i_5}}, \quad (3.36)$$

and one needs to subtract higher-point contributions from triangles, boxes and pentagons.

1-point residues Finally, the coefficients of the tadpole residues Δ_{i_1} can be determined by evaluating the integrand on the corresponding single cut $D_{i_1} = 0$, where the integrand decomposition becomes

$$\Delta_{i_1} = \frac{\mathcal{N}(q_s, \mu_s^2)}{\prod_{j \neq i_1} D_j} - \sum_{i_2} \frac{\Delta_{i_1 i_2}}{D_{i_2}} - \sum_{i_2, i_3} \frac{\Delta_{i_1 i_2 i_3}}{D_{i_2} D_{i_3}} - \sum_{i_2, i_3, i_4} \frac{\Delta_{i_1 i_2 i_3 i_4}}{D_{i_2} D_{i_3} D_{i_4}} - \sum_{i_2, i_3, i_4, i_5} \frac{\Delta_{i_1 i_2 i_3 i_4 i_5}}{D_{i_2} D_{i_3} D_{i_4} D_{i_5}}, \quad (3.37)$$

and one needs to subtract higher-point contributions from bubbles, triangles, boxes and pentagons.

Sampling the integrand Since, with the exception of 5-point cuts and four-dimensional 4-point cuts, every multiple cut has an infinite number of solutions, there is also an infinite number of choices on how to sample the numerator in order to obtain a system of equations for the coefficients of the residues. A choice proposed in Ref. [128], and then implemented in the public code SAMURAI [43], has been the usage of values of the loop coordinates corresponding to a *Discrete Fourier Transform* which can be used to project out the values of the coefficients of the polynomial residues. More details on this approach can be found in the two references we mentioned. In Chapter 4, we present instead the *integrand reduction via Laurent expansion* method, where no choice of sampling is needed and the systems of equations for the coefficients are automatically diagonal.

Chapter 4

One-loop amplitudes via Laurent series expansion with NINJA

In this Section we will describe the *Integrand Reduction via Laurent Series Expansion* method, proposed in Ref. [96] and implemented in the public C++ library NINJA [97], which is an alternative one-loop integrand reduction algorithm to the traditional one we reviewed in Section 3.5.1. This new algorithm is based on the systematic application of the Laurent series expansion to an integrand on the multiple cuts. After performing a suitable Laurent expansion on a multiple cut, in the asymptotic limit both the integrand and the subtraction terms exhibit the same polynomial behavior as the residue. This allows one to directly identify the coefficients of the residues (and thus the ones of the Master Integrals) with the ones of the Laurent expansion of the integrand, corrected by subtraction terms which can be computed once and for all as functions of a subset of the higher-point coefficients. This leads to a diagonal system of equations for the coefficients of each residue and to a significant reduction of the number of subtraction terms which affect the computation of lower-point contributions.

NINJA is a public C++ library which provides a semi-numerical implementation of the integrand reduction via Laurent expansion. Since the integrand of a one-loop amplitude is a rational function of the loop momentum, a Laurent expansion can be performed via a partial fraction decomposition. NINJA implements it semi-numerically via a simplified polynomial division algorithm between the expansions of the numerator and the ones of the denominators.

The simplified subtractions and the diagonal systems of equations make the algorithm implemented in NINJA significantly simpler and lighter than the traditional one. The library has been interfaced with the one-loop package GOSAM [49, 98] and has been used to compute NLO corrections to Higgs boson production in association with a top-quark pair and a jet [100] and several six-, seven- and eight-point amplitudes involving both massive and massless particles as external states or propagating in the loop [99]. More recently, it has been used for producing new phenomenological analysis on Higgs boson production in gluon fusion in association with two and three jets [101]. These applications showed that NINJA

has better performance and numerical stability than implementations of traditional integrand reduction algorithms.

In this chapter we will focus on the description of the algorithm and its implementation, while in Chapter 5 we will present several phenomenological results obtained using NINJA. In Section 4.1 we will review the method, pointing out its differences with respect to the traditional one. In Section 4.2 we will give more details about the implementation in the C++ library NINJA, whose basic usage is described in Appendix C.

4.1 The method

As we showed in Chapter 3, the coefficients appearing in the integrand decomposition can be computed by evaluating the integrand on *multiple cuts*, i.e. on values of the loop momentum \bar{q} such that a subset of loop denominators vanish [42]. More in detail, the coefficients of a k -point residue $\Delta_{i_1 \dots i_k}$ can be determined by evaluating the integrand on the corresponding k -ple cut $D_{i_1} = \dots = D_{i_k} = 0$. For these values of the loop momentum, the only non-vanishing contributions of the integrand decomposition are the ones coming from the residue in consideration and from all the higher-point residues which have $\{D_{i_1}, \dots, D_{i_k}\}$ as a subset of their loop denominators.

Within the original integrand reduction algorithm [42, 43, 128], the coefficients are computed by sampling the numerator of the integrand on a finite subset of the on-shell solutions, subtracting all the non-vanishing contributions coming from higher-point residues, and finally solving the resulting linear system of equations. This is therefore a top-down approach, where higher-point residues are computed first, starting from $k = 5$, and systematically subtracted from the integrand for the evaluation of lower-point contributions. These are referred to as subtractions at the integrand level.

The integrand reduction via Laurent expansion method, presented in Ref. [96], improves this reduction strategy by elaborating on techniques proposed in [30, 37]. Whenever the analytic dependence of the integrand on the loop momentum is known, this approach allows to compute the coefficients of a residue $\Delta_{i_1 \dots i_k}$ by performing a Laurent expansion of the integrand with respect to one of the components of the loop momentum which are not fixed by the on-shell conditions of the corresponding multiple cut $D_{i_1} = \dots = D_{i_k} = 0$. In the asymptotic limit defined by this Laurent expansion, both the integrand and the higher-point subtractions exhibit the same polynomial behavior as the residue. Therefore one can directly identify the unknown coefficients with the ones of the Laurent expansion of the integrand, corrected by the contributions coming from higher-point residues.

Hence, by choosing a suitable Laurent expansion, one obtains a diagonal system of equations for the coefficients of the residues, while the subtractions of higher-point contributions can be implemented as *corrections at the coefficient level* which replace the subtractions at the integrand level of the original algorithm. Since the polynomial structure of the residues is

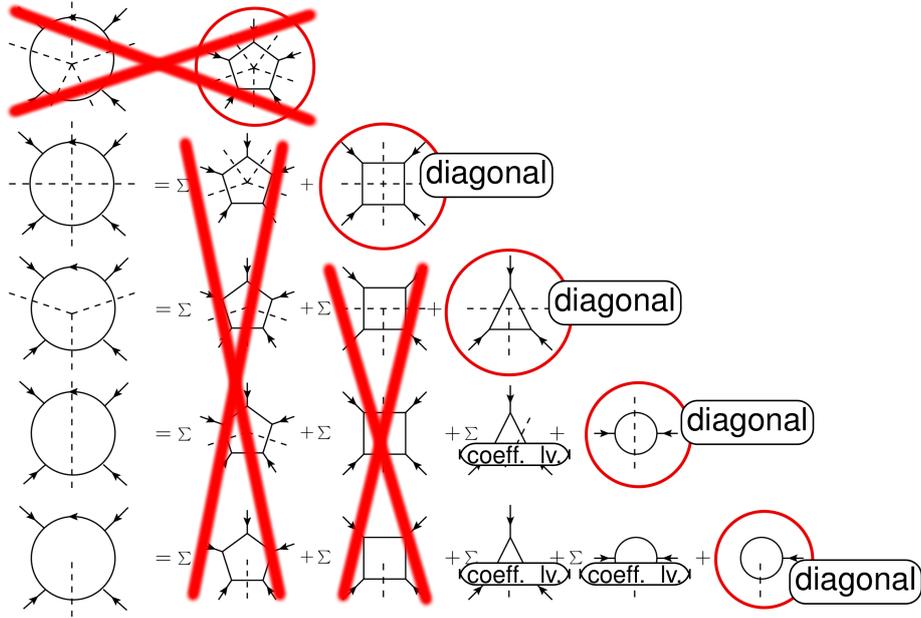


Figure 4.1: Schematic illustration of the simplifications involved in the Laurent expansion method, with respect to the traditional method depicted in Fig. 3.3.

universal and does not depend on the process, the parametric form of the coefficient-level corrections can be computed once and for all, in terms of a subset of the higher-point coefficients. More in detail, the corrections at the coefficient level are known functions of a subset of the coefficients of 3- and 2-point residues. In particular, no subtraction term coming from 4- and 5-point contributions is ever needed. This allows to skip the computation of the (spurious) 5-point contributions entirely, and to completely disentangle the determination of 4-point residues from the one of lower point contributions. A pictorial view of the simplifications obtained with the Laurent expansion method is given in Fig. 4.1.

In the following, we address more in detail the computation of 5-, 4-, 3-, 2-, and 1-point residues, also commonly known as *pentagons*, *boxes*, *triangles*, *bubbles* and *tadpoles* respectively. For simplicity, we first focus on renormalizable theories, where (up to a suitable choice of gauge) the maximum allowed rank of the integrand is equal to the number of loop denominators and the most general parametrization of the residues is the one given in Eq. (3.27). NINJA can also be used for the computation of integrals whose rank exceeds the number of denominators by one. The extension of the method to the higher-rank case is discussed in Subsection 4.1.1.

5-point residues As mentioned above, pentagon contributions are spurious. Within the original integrand reduction algorithm, their computation is needed because they appear in the subtractions at the integrand level required for the evaluation of lower-point contribu-

tions. A 5-point residue only has one coefficient, which can easily be computed by evaluating the numerator of the integrand on the corresponding 5-ple cut. Within the Laurent expansion approach, the subtraction terms coming from five-point residues always vanish in the asymptotic limits we consider, therefore their computation can be skipped. For this reason, in the library NINJA the computation of pentagons is disabled by default (even though it can be enabled for debugging purposes).

4-point residues The coefficient c_0 of a box contribution $\Delta_{i_1 i_2 i_3 i_4}$ can be determined via four-dimensional quadruple cuts, as shown in Eq. (3.33). Given the simplicity of that formula, this is the only coefficient which NINJA computes in the same way as the traditional algorithm. The coefficient c_4 can instead be determined by evaluating the integrand on d -dimensional quadruple cuts in the asymptotic limit of large μ^2 [37]. A d -dimensional quadruple cut has an infinite number of solutions which can be parametrized by the (-2ϵ) -dimensional variable μ^2 . These solutions become simpler in the considered limit, namely

$$q_{\pm}^{\nu} = -p_{i_1}^{\nu} + a^{\nu} \pm \sqrt{\mu^2 + \beta} v_{\perp}^{\nu} \stackrel{\mu^2 \rightarrow \infty}{=} \pm \sqrt{\mu^2} v_{\perp}^{\nu} + \mathcal{O}(1), \quad (4.1)$$

where the vector a^{ν} and the constant β are fixed by the cut conditions. The coefficient c_4 is non-vanishing only if the rank of the numerator is greater or equal to the number of loop denominators. In a renormalizable theory, it can be found in the $\mu^2 \rightarrow \infty$ asymptotic limit as the leading term of the Laurent expansion of the integrand

$$\left. \frac{\mathcal{N}(q, \mu^2)}{\prod_{j \neq i_1, i_2, i_3, i_4} D_j} \right|_{q = \sqrt{\mu^2} v_{\perp} + \mathcal{O}(1)} = c_4 \mu^4 + \mathcal{O}(\mu^3). \quad (4.2)$$

The other coefficients of the boxes are spurious and, since they neither contribute to the final result nor to the subtraction terms, their computation can be skipped.

3-point residues The coefficients of the residues of a generic triangle contribution $\Delta_{i_1 i_2 i_3}$ can be determined by evaluating the integrand on the solutions of the corresponding d -dimensional triple cut [30]. These can be parametrized by the variable μ^2 and a free parameter t ,

$$q_{+}^{\nu} = -p_{i_1}^{\nu} + a^{\nu} + t e_3^{\nu} + \frac{\beta + \mu^2}{2t(e_3 \cdot e_4)} e_4^{\nu}, \quad q_{-}^{\nu} = -p_{i_1}^{\nu} + a^{\nu} + \frac{\beta + \mu^2}{2t(e_3 \cdot e_4)} e_3^{\nu} + t e_4^{\nu}, \quad (4.3)$$

where the vector a^{ν} and the constant β are fixed by the cut conditions $D_{i_1} = D_{i_2} = D_{i_3} = 0$. The momentum a^{ν} is a linear combination of e_1 and e_2 and is therefore orthogonal to e_3 and e_4 . On these solutions, the non-vanishing contributions to the integrand decomposition are the ones of the residue $\Delta_{i_1 i_2 i_3}$, as well as the ones of the boxes and pentagons which share the three cut denominators. However, after performing a Laurent expansion for large t and dropping the terms which vanish in this limit, the pentagon contributions vanish, while the

box contributions are constant in t but they also vanish when taking the average between the parametrizations q_+ and q_- of Eq. (4.3). More explicitly,

$$\begin{aligned} \frac{\mathcal{N}(q_{\pm}, \mu^2)}{\prod_{j \neq i_1, i_2, i_3} D_j} &= \Delta_{i_1 i_2 i_3} + \sum_j \frac{\Delta_{i_1 i_2 i_3 j}}{D_j} + \sum_{jk} \frac{\Delta_{i_1 i_2 i_3 j k}}{D_j D_k} \\ &= \Delta_{i_1 i_2 i_3} + d_1^{\pm} + d_2^{\pm} \mu^2 + \mathcal{O}(1/t), \quad \text{with } d_i^+ + d_i^- = 0. \end{aligned} \quad (4.4)$$

Moreover, the expansion of the integrand is given by

$$\begin{aligned} \frac{\mathcal{N}(q_+, \mu^2)}{\prod_{j \neq i_1, i_2, i_3} D_j} &= n_0^+ + n_7^+ \mu^2 + (n_4 + n_9 \mu^2) t + n_5 t^2 + n_6 t^3 + \mathcal{O}(1/t), \\ \frac{\mathcal{N}(q_-, \mu^2)}{\prod_{j \neq i_1, i_2, i_3} D_j} &= n_0^- + n_7^- \mu^2 + (n_1 + n_8 \mu^2) t + n_2 t^2 + n_3 t^3 + \mathcal{O}(1/t) \end{aligned} \quad (4.5)$$

and it has the same polynomial behavior as the expansion of the residue $\Delta_{i_1 i_2 i_3}$,

$$\begin{aligned} \Delta_{i_1 i_2 i_3}(q_+, \mu^2) &= c_0 + c_7 \mu^2 + (c_4 + c_9 \mu^2) (e_3 \cdot e_4) t + c_5 (e_3 \cdot e_4)^2 t^2 + c_6 (e_3 \cdot e_4)^3 t^3 + \mathcal{O}(1/t), \\ \Delta_{i_1 i_2 i_3}(q_-, \mu^2) &= c_0 + c_7 \mu^2 + (c_1 + c_8 \mu^2) (e_3 \cdot e_4) t + c_2 (e_3 \cdot e_4)^2 t^2 + c_3 (e_3 \cdot e_4)^3 t^3 + \mathcal{O}(1/t). \end{aligned} \quad (4.6)$$

By comparison of Eq. (4.4), (4.5) and (4.6) one can directly identify the ten triangle coefficients as the corresponding terms of the expansion of the integrand,

$$c_{0,7} = \frac{1}{2}(n_{0,7}^+ + n_{0,7}^-), \quad c_{1,4,8,9} = \frac{n_{1,4,8,9}}{(e_3 \cdot e_4)}, \quad c_{2,5} = \frac{n_{2,5}}{(e_3 \cdot e_4)^2}, \quad c_{3,6} = \frac{n_{3,6}}{(e_3 \cdot e_4)^3}. \quad (4.7)$$

Hence, with the Laurent expansion method, the determination of the 3-point residues does not require any subtraction of higher-point terms.

2-point residues The coefficients of a generic 2-point residue $\Delta_{i_1 i_2}$ can be evaluated on the on-shell solutions of the corresponding double cut $D_{i_1} = D_{i_2} = 0$, which can be parametrized as

$$\begin{aligned} q_+^{\nu} &= -p_{i_1}^{\nu} + x e_1^{\nu} + (\alpha_0 + x \alpha_1) e_2^{\nu} + t e_3^{\nu} + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2t(e_3 \cdot e_4)} e_4^{\nu}, \\ q_-^{\nu} &= -p_{i_1}^{\nu} + x e_1^{\nu} + (\alpha_0 + x \alpha_1) e_2^{\nu} + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2t(e_3 \cdot e_4)} e_3^{\nu} + t e_4^{\nu}, \end{aligned} \quad (4.8)$$

in terms of the three free parameters x , t and μ^2 , while the constants α_i and β_i are fixed by the on-shell conditions. After evaluating the integrand on these solutions and performing a Laurent expansion for $t \rightarrow \infty$, the only non-vanishing subtraction terms come from the triangles,

$$\begin{aligned} \frac{\mathcal{N}(q_{\pm}, \mu^2)}{\prod_{j \neq i_1, i_2} D_j} &= \Delta_{i_1 i_2} + \sum_j \frac{\Delta_{i_1 i_2 j}}{D_j} + \sum_{jk} \frac{\Delta_{i_1 i_2 j k}}{D_j D_k} + \sum_{jkl} \frac{\Delta_{i_1 i_2 j k l}}{D_j D_k D_l} \\ &= \Delta_{i_1 i_2} + \sum_j \frac{\Delta_{i_1 i_2 j}}{D_j} + \mathcal{O}(1/t). \end{aligned} \quad (4.9)$$

Even though the integrand and the subtraction terms are rational functions, in the asymptotic limit they both have the same polynomial behavior as the residue, namely

$$\begin{aligned}
\frac{\mathcal{N}(q_+, \mu^2)}{\prod_{j \neq i_1, i_2} D_j} &= n_0 + n_9 \mu^2 + n_1 x + n_2 x^2 - (n_5 + n_8 x)t + n_6 t^2 + \mathcal{O}(1/t) \\
\frac{\mathcal{N}(q_-, \mu^2)}{\prod_{j \neq i_1, i_2} D_j} &= n_0 + n_9 \mu^2 + n_1 x + n_2 x^2 - (n_3 + n_7 x)t + n_4 t^2 + \mathcal{O}(1/t) \quad (4.10) \\
\frac{\Delta_{i_1 i_2 j}(q_+, \mu^2)}{D_j} &= c_{s_3,0}^{(j)} + c_{s_3,9}^{(j)} \mu^2 + c_{s_3,1}^{(j)} x + c_{s_3,2}^{(j)} x^2 - \left(c_{s_3,5}^{(j)} + c_{s_3,8}^{(j)} x \right) t + c_{s_3,6}^{(j)} t^2 + \mathcal{O}(1/t) \\
\frac{\Delta_{i_1 i_2 j}(q_-, \mu^2)}{D_j} &= c_{s_3,0}^{(j)} + c_{s_3,9}^{(j)} \mu^2 + c_{s_3,1}^{(j)} x + c_{s_3,2}^{(j)} x^2 - \left(c_{s_3,3}^{(j)} + c_{s_3,7}^{(j)} x \right) t + c_{s_3,4}^{(j)} t^2 + \mathcal{O}(1/t) \quad (4.11)
\end{aligned}$$

$$\begin{aligned}
\Delta_{i_1 i_2}(q_+, \mu^2) &= c_0 + c_9 \mu^2 + c_1 (e_1 \cdot e_2) x + c_2 (e_1 \cdot e_2)^2 x^2 \\
&\quad + \left(c_5 + c_8 (e_1 \cdot e_2) x \right) (e_3 \cdot e_4) t + c_6 (e_3 \cdot e_4)^2 t^2 + \mathcal{O}(1/t) \\
\Delta_{i_1 i_2}(q_-, \mu^2) &= c_0 + c_9 \mu^2 + c_1 (e_1 \cdot e_2) x + c_2 (e_1 \cdot e_2)^2 x^2 \\
&\quad + \left(c_3 + c_7 (e_1 \cdot e_2) x \right) (e_3 \cdot e_4) t + c_4 (e_3 \cdot e_4)^2 t^2 + \mathcal{O}(1/t). \quad (4.12)
\end{aligned}$$

The coefficients $c_{s_3,i}^{(j)}$ of the expansion of the subtractions terms in Eq.s (4.11) are known parametric functions of the triangle coefficients. Hence, the subtraction of the triangle contributions can be implemented by applying coefficient-level corrections to the terms appearing in the expansion of the integrand. More explicitly, by inserting Eq.s (4.10), (4.11) and (4.12) in Eq. (4.9) one gets

$$\begin{aligned}
c_{0,9} &= n_{0,9} - \sum_j c_{s_3,0,9}^{(j)}, \\
c_{1,3,5} &= \frac{1}{(e_1 \cdot e_2)} \left(n_{1,3,5} - \sum_j c_{s_3,1,3,5}^{(j)} \right), \\
c_{2,4,6,7,8} &= \frac{1}{(e_1 \cdot e_2)^2} \left(n_{2,4,6,7,8} - \sum_j c_{s_3,2,4,6,7,8}^{(j)} \right). \quad (4.13)
\end{aligned}$$

1-point residues The only non-spurious coefficient c_0 of a tadpole residue Δ_{i_1} can be computed by evaluating the integrand on solutions of the single cut $D_{i_1} = 0$. For this purpose, one can consider 4-dimensional solutions of the form

$$q_+^\nu = -p_{i_1}^\nu + t e_3^\nu + \frac{m_{i_1}^2}{2t (e_3 \cdot e_4)} e_4^\nu, \quad (4.14)$$

parametrized by the free variable t . In the asymptotic limit $t \rightarrow \infty$, only bubble and triangle subtraction terms are non-vanishing,

$$\begin{aligned} \frac{\mathcal{N}(q_+)}{\prod_{j \neq i_1} D_j} &= \Delta_{i_1} + \sum_j \frac{\Delta_{i_1 j}}{D_j} + \sum_{jk} \frac{\Delta_{i_1 jk}}{D_j D_k} + \sum_{jkl} \frac{\Delta_{i_1 jkl}}{D_j D_k D_l} \\ &= \Delta_{i_1} + \sum_j \frac{\Delta_{i_1 j}}{D_j} + \sum_{jk} \frac{\Delta_{i_1 jk}}{D_j D_k} + \mathcal{O}(1/t). \end{aligned} \quad (4.15)$$

Similarly to the case of the 2-point residues, in this limit the integrand and the subtraction terms exhibit the same polynomial behavior as the residue, i.e.

$$\frac{\mathcal{N}(q_+)}{\prod_{j \neq i_1} D_j} = n_0 + n_4 (e_3 \cdot e_4) t + \mathcal{O}(1/t) \quad (4.16)$$

$$\frac{\Delta_{i_1 j}(q_+)}{D_j} = c_{s_2,0}^{(j)} + c_{s_2,4}^{(j)} (e_3 \cdot e_4) t + \mathcal{O}(1/t) \quad (4.17)$$

$$\frac{\Delta_{i_1 jk}(q_+)}{D_j D_k} = c_{s_3,0}^{(jk)} + c_{s_3,4}^{(jk)} (e_3 \cdot e_4) t + \mathcal{O}(1/t) \quad (4.18)$$

$$\Delta_{i_1}(q_+) = c_0^{(i_1)} + c_4^{(i_1)} t + \mathcal{O}(1/t). \quad (4.19)$$

Putting everything together, the coefficient of the tadpole integral can be identified with the corresponding one in the expansion of the integrand, corrected by coefficient-level subtractions from bubbles and triangles

$$c_0 = n_0 - \sum_j c_{s_2,0}^{(j)} - \sum_{jk} c_{s_3,0}^{(jk)}. \quad (4.20)$$

The subtraction terms $c_{s_2,0}^{(j)}$ and $c_{s_3,0}^{(jk)}$, coming from 2-point and 3-point contributions respectively, are known parametric functions of the coefficients of the corresponding higher-point residues.

4.1.1 Higher-rank extension

As we pointed out in Ref. [96], the Laurent expansion method can be generalized to non-renormalizable and effective theories with higher-rank numerators. In a renormalizable theory, with a proper choice of gauge the rank r can not be greater than the number n of loop propagators. NINJA can also be used for the computation of integrals with $r = n + 1$. Here we describe the generalization of the method to the higher-rank case we presented in [97], underlining the points where it differs from the renormalizable case.

While the extension of the Laurent expansion method for the computation of higher-rank 3-point and 2-point residues is straightforward, for 4-point and 1-point residues some further observations are in order. The generalization of the Laurent expansion method described here allows to efficiently compute the non-spurious coefficients of 4- and 1-point residues without spoiling the nice features of the algorithm, such as the simplified subtractions of higher-point contributions and the diagonal systems of equations.

4-point residues The coefficient c_0 can be computed exactly as in the renormalizable case. For the coefficient c_4 , one needs instead to keep also the next-to-leading term in the μ^2 expansion described before, so that the d -dimensional solutions of a quadruple cut, given in Eq. (4.1), in the asymptotic limit become

$$q_{\pm}^{\nu} = -p_{i_1}^{\nu} + a^{\nu} \pm \sqrt{\mu^2 + \beta} v_{\perp}^{\nu} \stackrel{\mu^2 \rightarrow \infty}{=} -p_{i_1}^{\nu} + a^{\nu} \pm \sqrt{\mu^2} v_{\perp}^{\nu} + \mathcal{O}\left(\frac{1}{\sqrt{\mu^2}}\right), \quad (4.21)$$

where it is worth noticing that a^{ν} can be obtained as the average of the two solutions of the corresponding four-dimensional quadruple cut. In this limit, the expansion of the integrand reads

$$\left. \frac{\mathcal{N}(q, \mu^2)}{\prod_{j \neq i_1, i_2, i_3, i_4} D_j} \right|_{q = \sqrt{\mu^2} v_{\perp} + a + \mathcal{O}(\mu^{-1})} = c_5 v_{\perp}^2 \mu^5 + c_4 \mu^4 + \mathcal{O}(\mu^3), \quad (4.22)$$

hence the leading term is now the spurious coefficient c_5 , but c_4 can still be obtained as the next-to-leading term. This can be implemented semi-numerically, by keeping the two leading terms of the expansion of the numerator and performing a polynomial division with respect to the two leading terms in the expansion of the uncut denominators which have the form

$$D_{h \neq i_1, i_2, i_3, i_4} \Big|_{q=q_{\pm}} = d_{h,0} \sqrt{\mu^2} + d_{h,1} + \mathcal{O}\left(\frac{1}{\sqrt{\mu^2}}\right). \quad (4.23)$$

Given the very limited number of terms involved, the division can be implemented very efficiently in a small number of operations (more details are given in Section 4.2.1). We observe that the computation and the subtraction of pentagons is not needed in the higher-rank case either.

1-point residues On higher-rank 1-point residues Δ_{i_1} we consider d -dimensional solutions of the corresponding single cut of the form

$$q_{+}^{\nu} = -p_{i_1}^{\nu} + t e_1^{\nu} + \frac{m_{i_1}^2 + \mu^2}{2t(e_1 \cdot e_2)} e_2^{\nu}, \quad q_{-}^{\nu} = -p_{i_1}^{\nu} + t e_3^{\nu} + \frac{m_{i_1}^2 + \mu^2}{2t(e_3 \cdot e_4)} e_4^{\nu}, \quad (4.24)$$

in terms of the free variables t and μ^2 . By taking the $t \rightarrow \infty$ limit of the integrand and the subtraction terms evaluated on these solutions, we obtain an asymptotic polynomial expansion of the form

$$\frac{\mathcal{N}(q_{\pm}, \mu^2)}{\prod_{j \neq i_1} D_j} = n_0^{\pm} + n_1^{\pm} t + n_2^{\pm} t^2 + n_3^{\pm} \mu^2 + \mathcal{O}(1/t) \quad (4.25)$$

$$\frac{\Delta_{i_1 j}(q_{\pm}, \mu^2)}{D_j} = c_{s_2,0}^{\pm(j)} + c_{s_2,1}^{\pm(j)} t + c_{s_2,2}^{\pm(j)} t^2 + c_{s_2,3}^{\pm(j)} \mu^2 + \mathcal{O}(1/t) \quad (4.26)$$

$$\frac{\Delta_{i_1 j k}(q_{\pm}, \mu^2)}{D_j D_k} = c_{s_3,0}^{\pm(jk)} + c_{s_3,1}^{\pm(jk)} t + c_{s_3,2}^{\pm(jk)} t^2 + c_{s_3,3}^{\pm(jk)} \mu^2 + \mathcal{O}(1/t). \quad (4.27)$$

One can check that the non-spurious coefficients of the tadpole are given in terms of the ones of the expansions above by

$$\begin{aligned}
c_0 &= n_0^+ - \sum_j c_{s_2,0}^{+(j)} - \sum_{jk} c_{s_3,0}^{+(jk)}, \\
c_{14} &= n_3^+ - \sum_j c_{s_2,3}^{+(j)} - \sum_{jk} c_{s_3,3}^{+(jk)}, \\
c_{15} &= \frac{2}{(e_3 \cdot e_4)} \left(n_3^- - \sum_j c_{s_2,3}^{-(j)} - \sum_{jk} c_{s_3,3}^{-(jk)} - c_{14} \right).
\end{aligned} \tag{4.28}$$

4.2 Implementation in the C++ library NINJA

The C++ library NINJA provides a semi-numerical implementation of the Laurent expansion method described in Section 4.1. The Laurent series expansion is typically an analytic operation, but since a one-loop integrand is a rational function of the loop variables, its expansion can be obtained via a partial fraction decomposition between the numerator and the denominators. This is implemented in NINJA via a simplified polynomial-division algorithm, which takes as input the coefficients of a parametric expansion of the numerator \mathcal{N} and computes the leading terms of the quotient of the polynomial division with respect to the uncut denominators. In this section we give further details about the implementation of the reduction. The usage of the library and its input are described in Appendix C.

4.2.1 Reduction via polynomial division

For every phase-space point, NINJA at run-time computes the parametric solutions of the multiple cuts corresponding to each residue. The Laurent expansion of the integrand on these solutions is performed via a simplified polynomial division between the expansion of the numerator and the set of the uncut denominators. The coefficients of this expansion are corrected by the coefficient-level subtractions appearing in Eq. (4.13) and (4.20). The non-spurious coefficients are finally multiplied by the corresponding Master Integrals in order to obtain the integrated result as in Eq. (3.29).

NINJA takes as input the numerator and three parametric expansions of the same, which roughly correspond to the ones described in Eq. (4.1) (or (4.21) for the higher-rank case), (4.3) and (4.8) respectively. No new expansion is needed for the tadpoles, where we can use a special case of the parametric expansion we defined for the triangles. A more detailed definition of the input is given in Appendix C.3.1. The coefficients of the expansions of the numerator are written on a contiguous array. The Laurent expansion of the integrand is thus obtained via a simplified polynomial division between these and the uncut denominators. This division is performed in-place on the same array, keeping only the elements which are needed for the final result. A possible implementation for an univariate expansion, with a

numerator

$$\mathcal{N} = \text{num}[0] t^r + \text{num}[1] t^{r-1} + \dots + \text{num}[\text{nterms}-1] t^{r-\text{nterms}+1} + \mathcal{O}(t^{r-\text{nterms}})$$

and denominator

$$D = \text{d}[0] t + \text{d}[1] + \text{d}[2] \frac{1}{t},$$

would have the form

```
void division(Complex num[], int nterms, Complex den[3])
{
    for (int i=0; i<nterms; ++i) {
        num[i] /= den[0];
        if (i+1<nterms) {
            num[i+1] -= den[1]*num[i];
            if (i+2<nterms)
                num[i+2] -= den[2]*num[i];
        }
    }
}
```

One can check that this routine correctly replaces the first `nterms` elements of the array `num` with the first `nterms` leading elements of the Laurent expansion of \mathcal{N}/D . The actual implementation in NINJA, having to deal with multivariate expansions, is significantly more involved than the `division` procedure presented here. Nevertheless, it qualitatively follows the same algorithm.

In the case of the μ^2 -expansion needed for the higher-rank case (defined from Eq. (4.21)), the implementation can however be even simpler. Indeed every expansion only needs to contain the two leading terms. More in detail, if `num` and `den` are arrays of length two containing the leading and next-to-leading terms in the expansion of the numerator and a denominator respectively, we can perform the division in place with the commands

```
num[0] /= den[0];
num[1] -= den[1]*num[0];
num[1] /= den[0];
```

which will have the effect of replacing the entries of `num` with the ones of the expansion of \mathcal{N}/D .

The coefficients obtained by the division are then corrected by the coefficient-level subtractions and thus identified with the corresponding coefficients of the residues, as explained in Section 4.1. Once the reduction is complete, the non-spurious coefficients are multiplied by the corresponding Master Integrals.

4.2.2 Master Integrals

NINJA calls the routines implementing the Master Integrals through a generic interface which, as in the case of the numerator (see Appendix C), is defined in the C++ code via an abstract class. This allows one to use any integral library which can be specified at run-time. More details on the implementation of this interface are given in Appendix C. The current version of NINJA already implements it for two integral libraries.

The first built-in interface is a C++ wrapper of the routines of the ONELOOP library [45, 129]. This wrapper caches every computed integral allowing constant-time lookup of their values from their arguments. The caching of the integrals can significantly speed up the computation, especially for complex processes. Every instance of the class has an independent cache of Master Integrals (hence, one may safely use it in multi-threaded applications by using one instance of the class per thread).

The second implementation of the interface uses instead the LOOPTOOLS library [44], which already has an internal cache of computed integrals.

By implementing a suitable interface, the user can specify any other library to be used, without the need of compiling NINJA again. This interface should provide implementations of the Master Integrals appearing in Eq. (3.29), except those whose numerator is proportional to μ^2 , which contribute to the *rational part* of the amplitude and are already implemented in NINJA.

Rational integrals The integrals of Eq. (3.29) whose numerator is proportional to powers of μ^2 are finite and contribute to the so-called *rational part* of the amplitude, while the other contributions are known, for historical reasons, with the name of *cut-constructible part*. As we stated, only the integrals contributing to the *cut-constructible part* need to be implemented by the libraries of Master Integrals, while the others are computed by NINJA using the formulas [23]

$$I_{i_1 i_2 i_3 i_4}[\mu^4] = -\frac{1}{6} + \mathcal{O}(\epsilon) \quad (4.29)$$

$$I_{i_1 i_2 i_3}[\mu^2] = \frac{1}{2} + \mathcal{O}(\epsilon) \quad (4.30)$$

$$I_{i_1 i_2}[\mu^2] = -\frac{s_{i_1 i_2}}{6} + \frac{m_{i_1}^2 + m_{i_2}^2}{2} + \mathcal{O}(\epsilon), \quad (4.31)$$

where s_{ij} is defined by

$$s_{ij} \equiv (p_i - p_j)^2 \quad (4.32)$$

in terms of the momenta p_i appearing in the denominators, as in Eq. (3.14). In the higher-rank case, as one can see from Eq. (3.31), three new kinds of rational integrals appear. They

have been all computed in Ref. [96] and they read

$$I_{i_1 i_2 i_3}[\mu^4] = \frac{1}{6} \left(\frac{s_{i_2 i_1} + s_{i_3 i_2} + s_{i_1 i_3}}{4} - m_{i_1}^2 - m_{i_2}^2 - m_{i_3}^2 \right) + \mathcal{O}(\epsilon) \quad (4.33)$$

$$I_{i_1 i_2}[\mu^2 ((q + p_{i_1}) \cdot v)] = \frac{((p_{i_2} - p_{i_1}) \cdot v)}{12} (s_{i_2 i_1} - 2m_{i_1}^2 - 4m_{i_2}^2) + \mathcal{O}(\epsilon) \quad (4.34)$$

$$I_{i_1}[\mu^2] = \frac{m_{i_1}^4}{2} + \mathcal{O}(\epsilon), \quad (4.35)$$

where v is an arbitrary momentum.

Other higher-rank integrals The libraries which implement Master Integrals interfaced with NINJA do not need to provide implementations of the higher-rank integrals appearing in Eq. (3.31), because the library has a default implementation of them in terms of lower point integrals. Specifying an alternative implementation is however possible. The only additional higher-rank integrals which contribute to the cut-constructible part of an amplitude are a bubble integral of rank 3 and a tadpole integral of rank 2. The latter can be written as a function of the scalar tadpole integral I_{i_1} as follows [96]

$$I_{i_1}[(q + p_{i_1}) \cdot e_3] ((q + p_{i_1}) \cdot e_4) = m_{i_1}^2 \frac{(e_3 \cdot e_4)}{4} \left(I_{i_1} + \frac{m_{i_1}^2}{2} \right) + \mathcal{O}(\epsilon). \quad (4.36)$$

As for the former, since the vector e_2 in the bubble integral of rank 3 appearing in Eq. (3.31) is massless, the corresponding integral is simply proportional to the form factor B_{111} ,

$$I_{i_1 i_2}[(q + p_{i_1}) \cdot e_2]^3 = ((p_{i_2} - p_{i_1}) \cdot e_2)^3 B_{111}(s_{i_2 i_1}, m_{i_1}^2, m_{i_2}^2). \quad (4.37)$$

The form factor can be computed using the formulas of Ref. [130], as a function of form factors of scalar integrals B_0 . In the special case with $s_{i_2 i_1} = 0$ we use Eq. (A.6.2) and (A.6.3) of that reference. For the general case $s_{i_2 i_1} \neq 0$ we implement instead the following formula [97],

$$B_{111}(s_{i_2 i_1}, m_{i_1}^2, m_{i_2}^2) = \frac{1}{4 s_{i_2 i_1}^3} \left\{ s_{i_2 i_1} \left(m_{i_1}^2 I_{i_1} + I_{i_1}[\mu^2] - m_{i_2}^2 I_{i_2} - I_{i_2}[\mu^2] \right) \right. \\ \left. - 4 I_{i_1 i_2}[\mu^2 ((q + p_{i_1}) \cdot (p_{i_2} - p_{i_1}))] \right. \\ \left. - 4 m_{i_1}^2 I_{i_1 i_2}[(q + p_{i_1}) \cdot (p_{i_2} - p_{i_1})] \right) \\ \left. + 4 (m_{i_2}^2 - m_{i_1}^2 - s_{i_2 i_1}) I_{i_1 i_2}[(q + p_{i_1}) \cdot (p_{i_2} - p_{i_1})]^2 \right\}. \quad (4.38)$$

Chapter 5

Phenomenological applications of NINJA and GOSAM

In this chapter we describe several results obtained using the one-loop package GOSAM and the NINJA library. NINJA can be interfaced to any one-loop generator capable of providing the input it needs, and in particular to packages which can reconstruct the analytic dependence of the numerators on the loop momentum. This allows to achieve a complete automation of the computation of one-loop amplitudes and to fully exploit the advantages of the algorithm implemented in the library for phenomenological applications. An interface between NINJA and the one-loop package GOSAM [49] is already built in the library. NINJA is indeed the default reduction library used by GOSAM since version 2.0 [98]. An interface with the package FORMCALC [44] is currently under development.

In Section 5.1 we give a general description of the one-loop package GOSAM and its interface with the NINJA library, showing the computation of several six-, seven- and eight-point amplitudes involving massive particles as either external states or internal states of the loop. We give several benchmarks and we also perform an assessment of the numerical stability of the computations. In Section 5.2 and 5.3 we present phenomenological results obtained with NINJA and GOSAM, by using available interfaces between the latter and Monte Carlo generators. These are examples of fully automated predictions at next-to-leading order in perturbation theory for physical observables in high-energy physics.

5.1 Multi-leg massive amplitudes with NINJA and GOSAM

In this section we describe the one-loop package GOSAM and its interface with the library NINJA. We assess the performance and numerical stability of this computational framework (which we will refer to as GOSAM+NINJA) on a selection of challenging calculations of scattering amplitudes with massive bosons and quarks, involving six, seven, and eight external legs.

5.1.1 The one-loop package GOSAM

GOSAM [49] is a PYTHON package for the automated computation of one-loop amplitudes. It takes as an input a process *run card* defining the process to be computed and some options. Alternatively, it can be used as a *One-Loop Provider* (OLP) interfaced to Monte Carlo programs according to the standards defined in [131, 132], for the fully automated computation of physical observables.

The generation of the integrands performed by GOSAM is based on Feynman diagrams, which are generated using QGRAF [133]. Analytic expressions for tree-level amplitudes and one-loop integrands are produced with the computer algebra system FORM [124], using the package SPINNEY [134] for the spinor algebra. The parts of the analytic expressions which do not depend on the loop momentum \bar{q} are factored out and substituted by global abbreviations. This makes the resulting formulas suited for integrand reduction methods, where several evaluations of the integrand at different values of \bar{q} are required. The integrands are thus cast in a suitable form for the reduction (depending on the reduction libraries to be used) and written into FORTRAN90 code optimized for fast numerical evaluation, using either HAGGIES [135] or the recent features of FORM-4 [124, 136, 137].

The FORTRAN code is then compiled and used as input for reduction libraries. GOSAM is currently interfaced to three reduction libraries: SAMURAI [43, 138], GOLEM95 [139–141] and NINJA [97]. Within GOSAM, one can switch between the three reduction libraries at run time. The first version of GOSAM used SAMURAI as default reduction library and GOLEM95 as *rescue system* for points detected as numerically unstable (more details on numerical stability will be given later). The current version of the code, namely GOSAM-2.0, uses instead NINJA as default reduction library.

In Fig. 5.1 we show the schematic workflow of GOSAM, both as a standalone generator of one-loop amplitudes (on the left) and as a OLP in combination with a Monte Carlo program (on the right). As mentioned above, in the standalone case the user fills out a process *run-card* where the process is defined and some options can be tuned. Once the run card is filled, GOSAM can be invoked with the command

```
gosam.py process.in
```

where `process.in` is the name of the run-card file. This creates a directory (whose path can be specified in the card) with all the files needed for the generation of the amplitude, including all the Feynman diagrams of the process obtained with QGRAF. The command

```
make source
```

in this directory computes the analytic expression of the loop integrands and writes the corresponding source code. The latter can then be compiled with

```
make compile
```

and the integrals are computed thanks to interfaces to the reduction libraries NINJA, GOLEM95 and SAMURAI.

Among the options which can be activated during the generation of the amplitudes, there is a new feature of GOSAM-2.0 called `diagsum`, which sums loop diagrams characterized (up to a shift in the integration variables) by the same set of loop momenta. Diagrams being summed could share the same loop propagators but differ by tree parts external to the loop, as shown in the example of Fig. 5.2. Another case where the `diagsum` option takes effect is when different diagrams have the same loop denominators but with different particle content, as shown in the example of Fig. 5.3.

When GOSAM is used as OLP, this mechanism is stirred by the Monte Carlo program in use. In this case the input for GOSAM is an *order file* generated by the Monte Carlo, with the details about the partonic sub-processes whose scattering amplitudes are needed. GOSAM reads this and creates a *contract file*, which contains details about how the OLP will provide the scattering amplitudes asked for in the order file. This allows the Monte Carlo program to communicate at run-time with OLPs such as GOSAM and provide full NLO predictions for physical observables.

5.1.2 Interfacing NINJA and GOSAM

The interface between NINJA and GOSAM was presented in [99]. NINJA, as mentioned in Section 4.2.1, takes as input the numerator and three parametric expansions of the same. The analytic generation of the integrands implemented in GOSAM makes the computation of the expansions needed by NINJA straightforward. Moreover, since the \bar{q} -independent part of the numerator is factored out into abbreviations, the generation of such expansions is very fast and efficient, compared to the time needed for the generation of the integrands themselves. These expansions are described in detail in Appendix C.3.1.

Within NINJA, an integrand is defined by an instance of a C++ class derived from the abstract class `ninja::Numerator`, with methods for the evaluation of the numerator expression and its expansions. In the GOSAM interface built in NINJA, we define a numerator class of this form (called `GoSamNumerator`) as a generic wrapper of subroutines of the same type as the ones generated by GOSAM. We also define routines which can be used by the FORTRAN90 code and wrap the calls of the C++ methods performing the computation of the integrals. The internal options of NINJA can, to some degree, be controlled from the code generated by GOSAM, by importing the FORTRAN90 module `ninjago_module` (details on its usage are given in Appendix C.4.4).

5.1.3 Precision tests

In this subsection, we assess the problem of estimating the numerical accuracy of a result computed within GOSAM using a reduction algorithm such as the one implemented in NINJA.

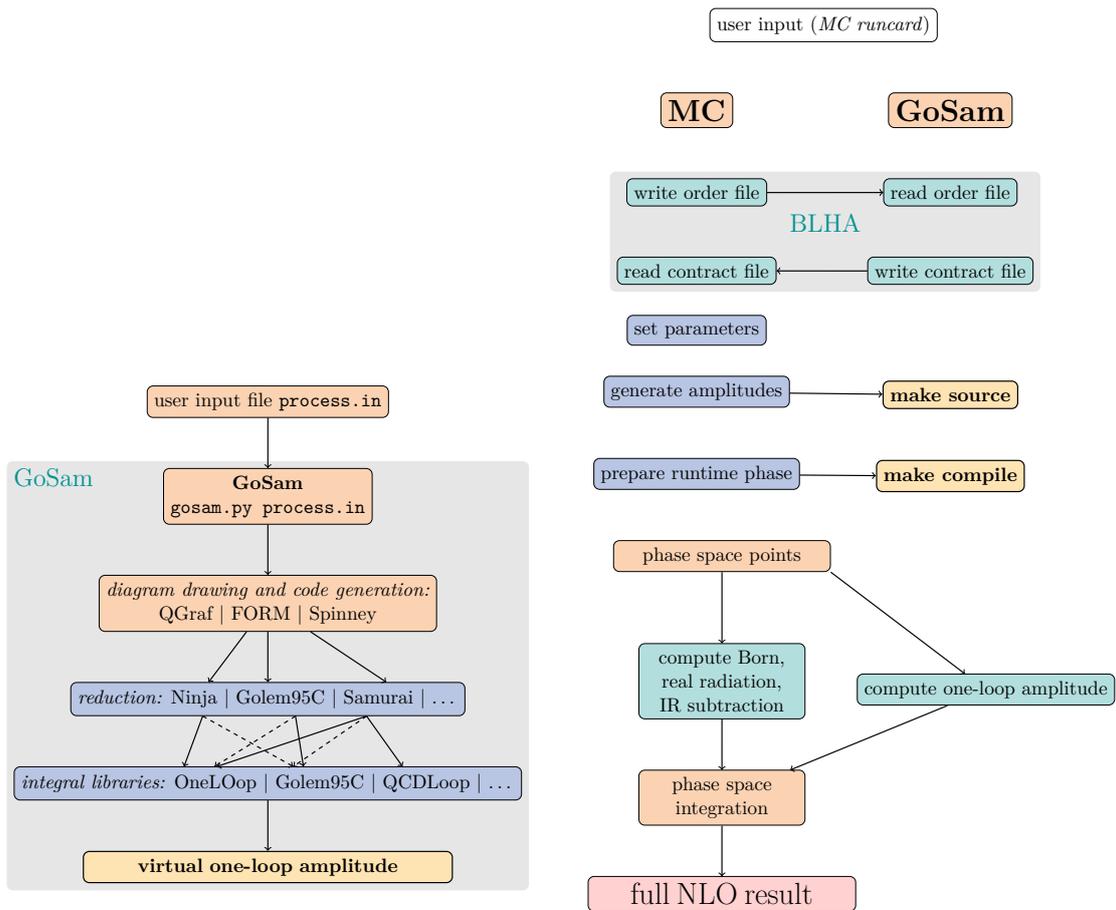


Figure 5.1: On the left, the basic work-flow of GoSAM. On the right, the schematic setup for GoSAM as an OLP in combination with a Monte Carlo program.

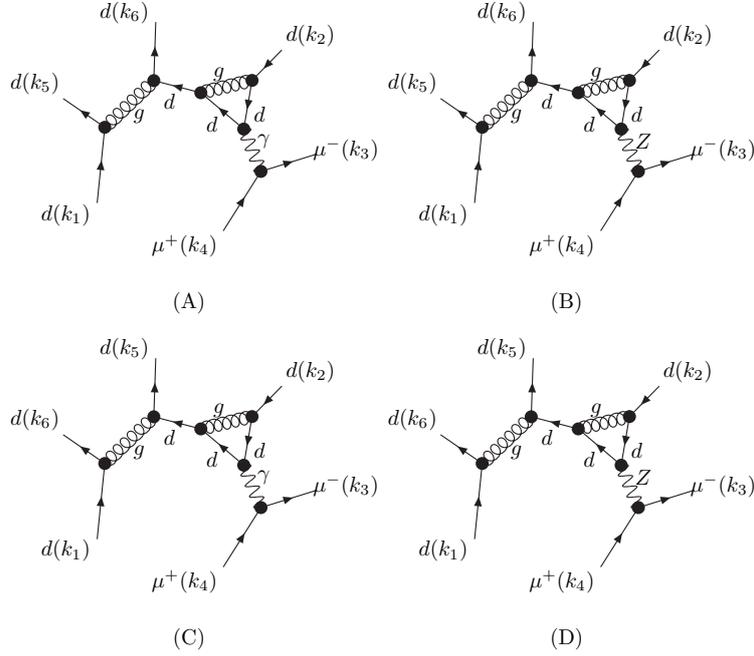


Figure 5.2: Example of diagrams sharing loop propagators but differing for their tree parts. This diagrams are automatically summed by GOSAM when the `diagsum=true`.

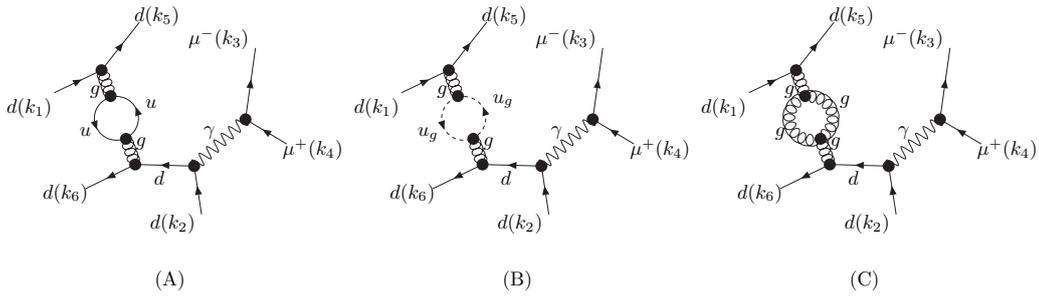


Figure 5.3: Example of diagrams sharing common loop denominators, but with different particle content in the loop. They are automatically summed by GOSAM when the `diagsum=true`.

Having a reliable method for the determination of the precision of the numerical computation is of primary importance. During a phase-space integration, for every phase-space point, if the (estimated) quality of the result falls below a certain threshold of accuracy, either the point is discarded or the evaluation of the amplitude is repeated by means of a numerically more stable, albeit less efficient, procedure. Examples of such methods involve the use of higher precision routines or, in the case of GOSAM, the use of traditional tensor reduction of the amplitude, provided by GOLEM95.

Here we mention three methods for the estimation of the accuracy of the computed amplitude at a given phase-space point. They are commonly known as *pole check*, *scaling test* and *rotation test* respectively.

The *pole check* is a simple and widely adopted strategy, which consists in testing whether the pole of the numerically computed amplitude agrees, after UV renormalization, with the analytic result obtained from the known universal behavior of the infrared singularities. The main drawback of this method is that the number of contributions to the finite part of an amplitude is often higher than the one of the contribution to its poles. Therefore, the pole check very often results in an overestimate of the precision of the computation, since it doesn't take into account all the sources of error.

The *scaling test*, proposed in Ref. [51], is instead a more reliable method, which comes at the price of a double evaluation of the amplitude. It consists in comparing the value of the computed amplitude with the one of an analogous computation where all the momenta and physical scales are multiplied by a constant factor x . As shown in [51], this method provides a very good correlation between the estimated precision, and the actual precision of the finite parts.

In the following, we will focus on the third method we mentioned, i.e. the *rotation test*. Similarly to the scaling test, also the rotation test consists in comparing two different evaluations of the amplitude. As the name suggests, in the second evaluation we rotate the external kinematics along a given axis (typically the z axis). Since a scattering amplitude is invariant under rotation, the obtained result can be directly compared with the one obtained with the original kinematics, and their difference can be used as an estimate of the numerical error. We tested that the choice of the angle of rotation does not affect the estimate, as long as this angle is not too small.

In order to study the correlation of the error estimated by the rotation test and the exact error, we follow the strategy of Ref. [51]. In particular, we generated 10^4 points for the process $u\bar{d} \rightarrow Wb\bar{b}g$ with massive bottom quarks. We define the “exact” amplitude A_{ex} as the one obtained by a computation in quadruple precision for the finite part and from the analytic expression for the poles¹. We simply denote as A the amplitude computed in double

¹The word “exact” is obviously not to be taken literally. The only important property we will use is that, with these definitions, the difference $|A_{ex} - A|$ is, for the purposes of our analysis, a very close approximation of the exact error of the computed amplitude A .

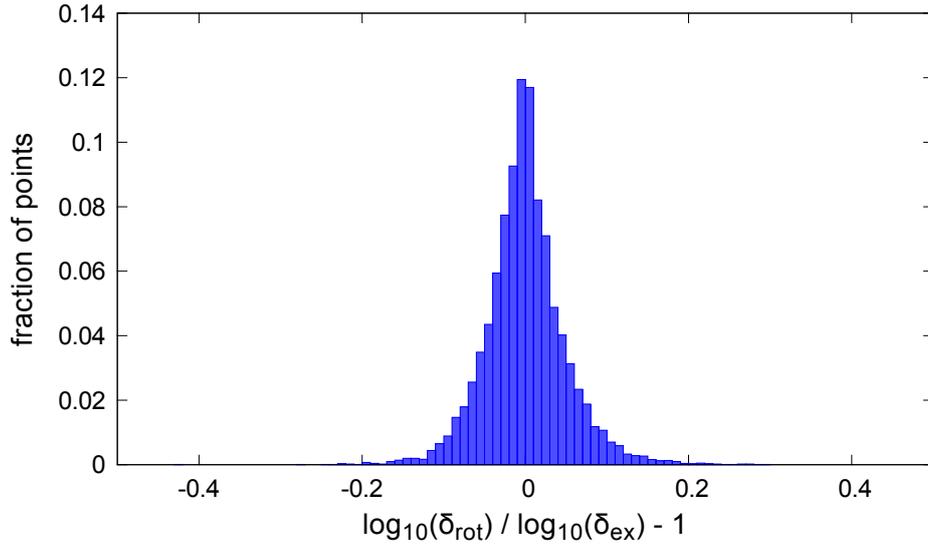


Figure 5.4: Correlation plot based on 10^4 points for the process $u\bar{d} \rightarrow Wb\bar{b}g$ with massive bottom quarks.

precision and A_{rot} the same amplitude computed after a rotation of the external kinematics. The choice of the process is motivated by the fact that, while being non-trivial and containing several mass scales, it is also simple enough to be evaluated on several phase-space points in quadruple precision. Hence, we define the “exact” error δ_{ex} as

$$\delta_{ex} = \left| \frac{A_{ex} - A}{A_{ex}} \right|, \quad (5.1)$$

and the estimated error δ_{rot} as

$$\delta_{rot} = 2 \left| \frac{A_{rot} - A}{A_{rot} + A} \right|. \quad (5.2)$$

In Fig. 5.4, we plot the distribution of the quantity

$$\mathcal{C} = \frac{\log_{10}(\delta_{rot})}{\log_{10}(\delta_{ex})} - 1, \quad (5.3)$$

evaluated for each phase space point. The good correlation between δ_{rot} and δ_{ex} is confirmed by the narrow distribution peaked at $\mathcal{C} = 0$. We observe a similar behavior between the rotation test and the scaling test.

In the following, we will employ the rotation test for the estimation of the precision of the finite part of each renormalized virtual matrix element. If we call δ_0 the error at any given phase space point and calculate it according to Eq. (5.2), we can define the precision of the finite part as $P_0 = \log_{10}(\delta_0)$. Concerning the precision of the double and single poles, $P_{-2} = \log_{10}(\delta_{-2})$ and $P_{-1} = \log_{10}(\delta_{-1})$, we simply use the formula in Eq. (5.1), where the poles of A_{ex} are computed, after renormalization, from the infrared (IR) divergencies, whose expressions are well known [142].

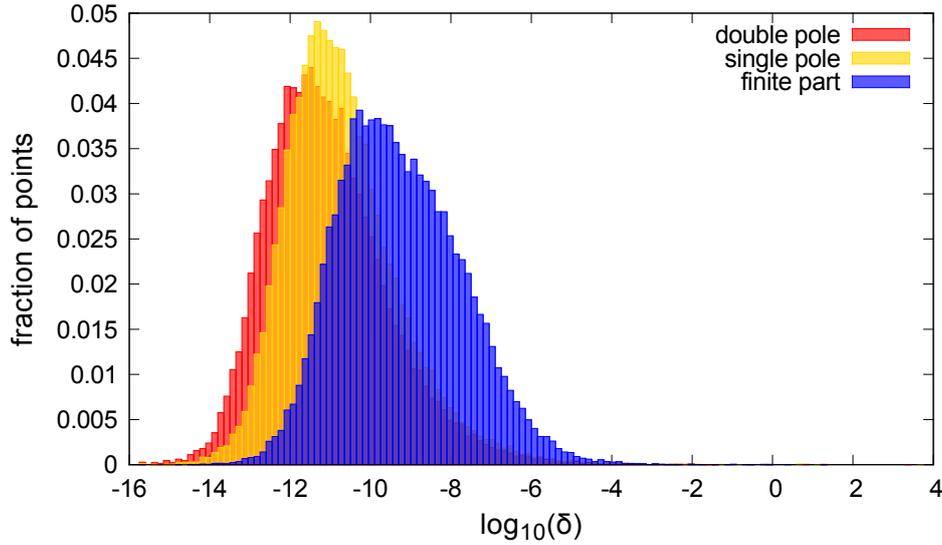


Figure 5.5: Precision Plot for $gg \rightarrow t\bar{t}Hg$: the distributions are obtained using $5 \cdot 10^4$ randomly distributed phase space points.

In order to statistically assess the precision of the results obtained with GOSAM+NINJA, in Fig. 5.5 and 5.6, we plot the distributions of P_{-2} (precision of the double pole), P_{-1} (single pole) and P_0 (finite part) for two challenging virtual amplitudes with massive internal and external particles, namely $gg \rightarrow t\bar{t}Hg$ ($t\bar{t}Hj$) and $u\bar{u} \rightarrow Hw\bar{u}gg$ ($H + 4j$) in VBF. By selecting an upper bound on the value of P_0 , we can set a *rejection criterion* for phase space points in which the quality of the calculated scattering amplitudes falls below the requested precision. This also allows to estimate the percentage of points which would be discarded (or recomputed by the rescue system). This value, as expected by analyzing the shape of the various distributions, is strongly process dependent and should be selected according to the particular phenomenological analysis at hand. As a benchmark value, in Ref. [51] the threshold for rejection was set to $P_0 = -3$. In a similar fashion, in Table 5.1, we provide the percentages of *bad points*, which are points whose precision falls below the threshold, for increasing values of the rejection threshold.

The distributions in the two plots refer to sets of $5 \cdot 10^4$ and $1 \cdot 10^5$ phase space points, for $gg \rightarrow t\bar{t}Hg$ and $u\bar{u} \rightarrow Hw\bar{u}gg$ (VBF) respectively, where the random external kinematic is generated using RAMBO [143]. The use of the algorithm implemented in NINJA yields significant improvements both in the accuracy of results and in reduction of the computational time, due to a more efficient reduction and less frequent calls to the rescue system. These features make GOSAM+NINJA an extremely competitive framework for massive, as well as massless, one-loop calculations.

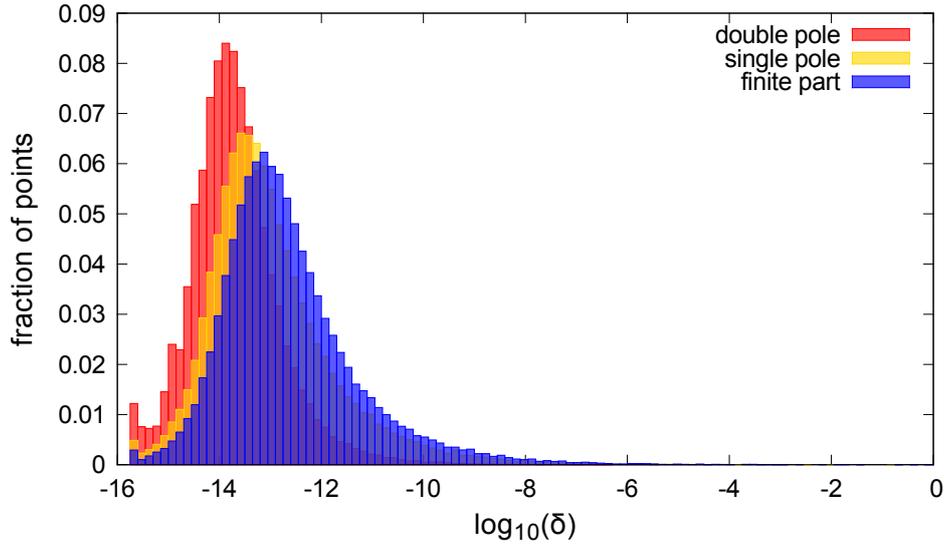


Figure 5.6: Precision plot for $u\bar{u} \rightarrow Hu\bar{u}gg$ in VBF: the distributions are obtained using 10^5 randomly distributed phase space points.

P_0	$u\bar{u} \rightarrow Hu\bar{u}gg$	$gg \rightarrow t\bar{t}Hg$
-3	0.02%	0.06%
-4	0.04%	0.16%
-5	0.08%	0.56%

Table 5.1: Percentage of bad points as a function of the rejection threshold P_0 , for the most complex partonic sub-processes of $pp \rightarrow H + 4j$ in VBF and $pp \rightarrow t\bar{t}H + 1j$ respectively.

Precision tests within GOSAM-2.0

Since version 2.0, GOSAM has an automated built-in implementation of the rotation test we presented above, in addition to the pole check which was already available in the previous version of the code. The new version also allows a more flexible and refined assessment of the accuracy of the results, which in turn determines when the rescue system is called (if present). More in detail, the trigger for the rescue system is a hybrid method, that takes advantage of the computational speed of the pole test, combined with the higher reliability of the rotation test. This hybrid method requires setting three different thresholds, namely P_{high} , P_{set} and P_{low} (with $P_{high} > P_{low}$). The assessment of the precision is done as follows:

- after computing the matrix elements, GOSAM checks the precision $-P_{-1}$ of the single pole with the *pole test*, and if $-P_{-1} > P_{high}$ the point is automatically accepted as stable
- if $-P_{-1} < P_{low}$ the point is either rejected or sent to the rescue system

- if $P_{high} > -P_{-1} > P_{low}$, the computation of the amplitude is repeated with rotated kinematics and the *rotation check* will determine whether the point should be accepted or not, using the threshold P_{set} for the comparison.

The motivation behind this mechanism can be stated as follows. If the pole is extremely accurate, then we can also expect the finite part to be acceptably good. If instead the accuracy of the pole is very low and falls below the lower threshold P_{low} , we cannot even expect the finite part – whose computation has more sources of numerical instabilities – to be acceptable. Finally, if the accuracy of the pole is good but not excellent, we cannot rely on this check alone and the rotation test is triggered, giving the final answer. Also notice that one can disable the rotation check by setting $P_{high} = P_{low}$.

5.1.4 Application to Massive Amplitudes

In Table 5.2 we give a list of processes computed using GOSAM and NINJA, showing the number of diagrams and the timing for the calculation of a full color- and helicity-summed amplitude on one phase-space point. Each of these processes involve six, seven or eight external legs, and at least one massive particle either in the final state or running in the loop. While some of the considered processes had already been studied in the literature, the virtual NLO QCD contributions to $pp \rightarrow Wb\bar{b} + n$ jets ($n = 1, 2$), $pp \rightarrow Zb\bar{b}j$, $pp \rightarrow Zt\bar{t}j$, $pp \rightarrow VVVj$ (with $V = W, Z$), $pp \rightarrow ZZZZ$, and $pp \rightarrow H + n$ jets ($n = 4, 5$) in VBF have been presented in Ref. [99] for the first time. When occurring in the final state, the *bottom* quark is treated as massive. In Appendix D we provide results for a phase space point and a detailed list of the input parameters for each process listed in Table 5.2.

Benchmarks: GoSAM + NINJA			
Process		# NLO diagrams	ms/event
$W + 3j$	$d\bar{u} \rightarrow \bar{\nu}_e e^- ggg$	1 411	226
$Z + 3j$	$d\bar{d} \rightarrow e^+ e^- ggg$	2 928	1 911
$Z Z Z + 1j$	$u\bar{u} \rightarrow Z Z Z g$	915	*12 000
$W W Z + 1j$	$u\bar{u} \rightarrow W^+ W^- Z g$	779	*7 050
$W Z Z + 1j$	$u\bar{d} \rightarrow W^+ Z Z g$	756	*3 300
$W W W + 1j$	$u\bar{d} \rightarrow W^+ W^- W^+ g$	569	*1 800
$Z Z Z Z$	$u\bar{u} \rightarrow Z Z Z Z$	408	*1 070
$W W W W$	$u\bar{u} \rightarrow W^+ W^- W^+ W^-$	496	*1 350
$t\bar{t}b\bar{b} (m_b \neq 0)$	$d\bar{d} \rightarrow t\bar{t}b\bar{b}$	275	178
	$gg \rightarrow t\bar{t}b\bar{b}$	1 530	5 685
$t\bar{t} + 2j$	$gg \rightarrow t\bar{t}gg$	4 700	13 827
$Z b\bar{b} + 1j (m_b \neq 0)$	$dug \rightarrow ue^+ e^- b\bar{b}$	708	*1 070
$W b\bar{b} + 1j (m_b \neq 0)$	$u\bar{d} \rightarrow e^+ \nu_e b\bar{b}g$	312	67
$W b\bar{b} + 2j (m_b \neq 0)$	$u\bar{d} \rightarrow e^+ \nu_e b\bar{b}s\bar{s}$	648	181
	$u\bar{d} \rightarrow e^+ \nu_e b\bar{b}d\bar{d}$	1 220	895
	$u\bar{d} \rightarrow e^+ \nu_e b\bar{b}gg$	3 923	5387
$W W b\bar{b} (m_b \neq 0)$	$d\bar{d} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}$	292	115
	$gg \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}$	1 068	*5 300
$W W b\bar{b} + 1j (m_b = 0)$	$u\bar{u} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}g$	3 612	*2 000
$H + 3j$ in GF	$dd \rightarrow Hgdd$	868	135
	$d\bar{d} \rightarrow Hgu\bar{u}$	467	56
	$d\bar{d} \rightarrow Hggg$	2 519	842
	$gg \rightarrow Hggg$	9 325	8 706
$t\bar{t}Z + 1j$	$u\bar{u} \rightarrow t\bar{t}e^+ e^- g$	1408	1 220
	$gg \rightarrow t\bar{t}e^+ e^- g$	4230	19 560
$t\bar{t}H + 1j$	$u\bar{u} \rightarrow Ht\bar{t}g$	320	76
	$gg \rightarrow t\bar{t}Hg$	1 517	1 505
$H + 3j$ in VBF	$u\bar{u} \rightarrow Hgu\bar{u}$	432	101
$H + 4j$ in VBF	$u\bar{u} \rightarrow Hgggu\bar{u}$	1 176	669
$H + 5j$ in VBF	$u\bar{u} \rightarrow Hgggu\bar{u}$	15 036	29 200

Table 5.2: A summary of results obtained with GoSAM+NINJA. Timings refer to full color- and helicity-summed amplitudes, using an Intel Core i7 CPU @ 3.40GHz, compiled with `ifort`. The timings indicated with an (*) are obtained with an Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz, compiled with `gfortran`. The C++ source code of NINJA was compiled with `g++`.

5.2 Higgs boson production plus a top quark pair and a jet

In this section we present the calculation of the cross section for Higgs boson production in association with a top quark pair plus one jet, at next-to-leading-order (NLO) accuracy in QCD. Besides the cross section, we show several distributions of physical observables, such as the invariant mass of the top quark pair, the transverse momentum and the pseudorapidity of the Higgs boson.

The production rate for a Higgs boson associated with a top-antitop pair ($t\bar{t}H$) is particularly interesting for the determination of the nature of the new boson observed at LHC [1, 2], since it is directly proportional to the SM Yukawa coupling of the Higgs boson to the top quark. As mentioned in Section 2.2, accurate predictions are necessary and will play a crucial role for the complete determination of the properties of such boson [110], and in particular to shed light on the structure of its couplings to the other particles. The study of differential observables and distributions involving a quark top pair and a Higgs as a final state will bring useful information for the determination of the couplings and the parity of the particle [144, 145].

The difficulties related to the analysis of the $t\bar{t}H$ channel are well known. This channel has small production rates, due to strongly suppressed parton distribution functions at the high center-of-mass energy required for the initial partons in order to produce the $t\bar{t}H$ final state. Additional difficulties are represented by the presence of various backgrounds and by the complexity of the final state, which make its kinematic reconstruction quite challenging [146].

At the parton level, the $t\bar{t}H$ production at next-to-leading order (NLO) in QCD has been known for some time [147–151]. More recently, this process has been employed in a number of studies, motivated by the new analyses performed at the LHC [144–146, 152].

Here we present the complete NLO QCD corrections to the process $pp \rightarrow t\bar{t}H + 1 \text{ jet}$ ($t\bar{t}Hj$) at the LHC. In Fig. 5.7 we show examples of diagrams contributing to the amplitudes for the two main sub-processes, namely

$$q\bar{q} \rightarrow t\bar{t}H g, \quad gg \rightarrow t\bar{t}H g.$$

All the other sub-processes can be obtained by crossings from these two. We illustrate the outcome of our calculation by showing the total cross section, and a selection of differential distributions.

The considered process can be important for the phenomenological analyses at the LHC, in particular in the high- p_T region, where the presence of the additional jet can be sensibly relevant. This also constitutes the first application of the Laurent series expansion method implemented in NINJA for the evaluation of one-loop amplitudes. Because of the presence of two mass scales and the top mass inside the loop, this process seriously compromises the numerical stability of traditional integrand-reduction algorithms, hence using NINJA for the reduction has been particularly important for both the performance and the accuracy of the

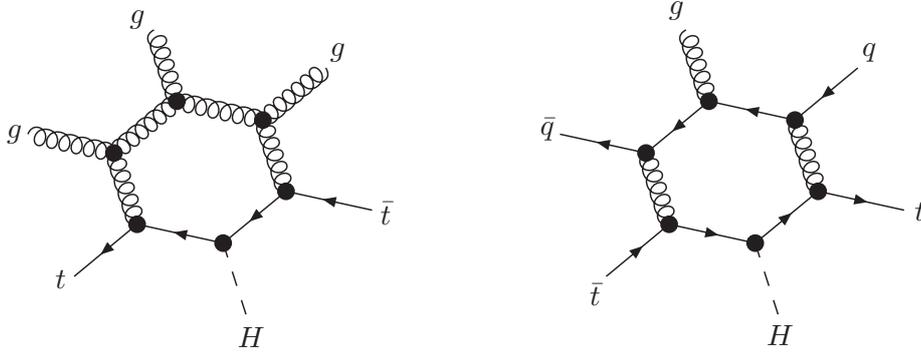


Figure 5.7: Sample of one-loop diagrams contributing to the NLO corrections to $gg \rightarrow t\bar{t}H$ and $q\bar{q} \rightarrow t\bar{t}Hg$.

computation. Indeed, as noted in the precision tests reported in Table 5.1, for the highly non-trivial process under consideration only a small set of phase-space points, of the order of few per mill, were detected as unstable. All these points have been recovered using the tensor reduction provided by GOLEM95 [139, 140], which is the default rescue system for GOSAM.

In order to compare the processes $pp \rightarrow t\bar{t}H$ and $pp \rightarrow t\bar{t}Hj$ at NLO QCD accuracy, we also used this framework for the computation of the cross section for $t\bar{t}H$ production. We found excellent agreement with the results presented in Ref.s [48, 144].

5.2.1 Numerical computation and results

Cross sections and differential distributions are obtained with an automated framework, using GOSAM as *One-Loop Provider* (OLP) and the Monte Carlo SHERPA [153]. For the virtual contributions computed with GOSAM, we used NINJA as reduction library. While GOSAM+NINJA provides the virtual corrections, the Born and the real emission matrix elements – necessary for the NLO cross section and distributions – are computed using SHERPA [153] with the library AMEGIC [154], which implements the Catani-Seymour dipole formalism [155, 156]. SHERPA also performs the integration over the phase space and the analysis. The code generated by GOSAM is linked to SHERPA by means of the Binoth Les Houches Accord (BLHA) [131] interface, as we briefly explained in Section 5.1.1.

The ultraviolet, the infrared, and the collinear singularities are regularized using *dimensional reduction*. The renormalization conditions are fixed along the lines of [148, 150], where the top mass is renormalized on-shell, while the strong coupling is renormalized in the $\overline{\text{MS}}$ scheme, decoupling the top quark from the running. The wave functions of the gluon and of the quarks are renormalized on-shell, i.e. the corresponding renormalization constants cancel

the external self-energy corrections exactly. We use the running of the strong coupling constant with one-loop and two-loop accuracy, for the computation of LO and NLO contributions respectively.

The values of double and the single poles, for each individual sub-process, have been checked to agree with the universal singular behavior of dimensionally regulated one-loop amplitudes [142]. The results fulfill gauge invariance, verified through the vanishing of the amplitudes when substituting the polarization vector of one or more gluons with the corresponding momentum.

In the following, we present numerical results for the integrated cross section and distributions. The masses of the particles involved and the parameters of the electroweak sector are shown in the following tables

PARAMETER	VALUE	PARAMETER	VALUE
\sqrt{s}	8 TeV	m_W	80.419 GeV
m_H	126 GeV	m_Z	91.1876 GeV
m_t	172.5 GeV	α_{EW}^{-1}	132.50698

The jets are clustered using the `antikt`-algorithm implemented in FASTJET [116, 157, 158] with radius $R = 0.5$, a minimum transverse momentum of $p_{T,jet} > 15$ GeV and pseudo-rapidity $|\eta| < 4.0$. The LO cross sections are computed with the LO parton-distribution functions `cteq6L1` [159], whereas at NLO we use `CT10` [160]. In order to study the scale dependence of the total cross section, we employ two different choices for the renormalization and factorization scales $\mu_R = \mu_F = \mu_0$, namely $\mu_0 = H_T$ and $\mu_0 = 2 \times GA_T$ with

$$H_T = \sum_{\substack{\text{final} \\ \text{states } f}} |p_{T,f}|, \quad (5.4)$$

$$GA_T = \sqrt[3]{m_{T,H} m_{T,t} m_{T,\bar{t}}} + \sum_{\text{jets } j} |p_{T,j}|. \quad (5.5)$$

Within this setup, for the two scale choices, we obtain the following total LO and NLO cross sections

CENTRAL SCALE	σ_{LO} [fb]	σ_{NLO} [fb]
$2 \times GA_T$	$80.03^{+35.64}_{-23.02}$	$100.6^{+0.00}_{-9.43}$
H_T	$88.93^{+41.41}_{-26.13}$	$102.3^{+0.00}_{-15.82}$

The scale dependence of the total cross section, depicted in Fig. 5.8, is strongly reduced by the inclusion of the NLO contributions. We notice that the two choices for the renormalization and factorization scales give very similar results.

In Fig. 5.9, we compare the distributions for the invariant mass of the top quark pair in $pp \rightarrow t\bar{t}Hj$ at LO and NLO with the NLO curve for $pp \rightarrow t\bar{t}H$. For $t\bar{t}Hj$, going from LO to

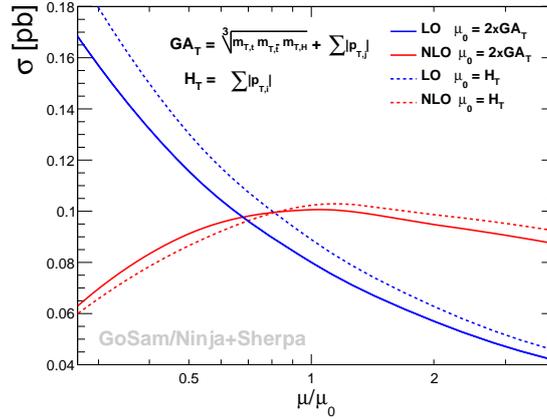


Figure 5.8: Scale dependence of the total cross section at LO and NLO.

NLO accuracy, we observe an increase in the distribution by 20–35% over the full kinematic range. On the other hand, when comparing the NLO $t\bar{t}H$ prediction with the NLO $t\bar{t}Hj$ curve, the cross section decreases due to the presence of the additional jet which takes away energy from the $t\bar{t}$ system. This is particularly evident near the $t\bar{t}$ production threshold, while for high values of the $t\bar{t}$ invariant mass the two NLO curves get closer.

In Fig. 5.10 and Fig. 5.11, we display the distributions of the transverse momentum p_T and the pseudorapidity η of the Higgs boson, respectively. Each plot contains the distributions at LO and NLO accuracy. The NLO corrections are particularly important for high values of the p_T , which are the kinematic regions involved in the boosted analyses [152, 161].

In Appendix D.18 we give, as a reference, benchmark points for the scattering amplitudes of the two independent partonic sub-processes we mentioned above.

The results we showed are a very non-trivial example of automated NLO calculation of physical observables using a framework which combines the one-loop generator GOSAM, the reduction library NINJA and the Monte Carlo SHERPA. This shows that this framework is capable of making physical predictions at NLO for multi-leg processes involving massive particles, both as final states and inside the loop of the virtual contribution. Moreover, with this computation we assessed the impact of further jet activity in $pp \rightarrow t\bar{t}H$, one of the most important channels for the direct determination of the coupling of the Higgs boson to fermions. We performed the computation for two different choices of renormalization and factorization scale, showing that they yield very similar results. The NLO QCD corrections reduce the scale uncertainty and their numerical impact can be sizable on the shapes of the distributions. Therefore they could be helpful for an accurate simulation of the signal in the experimental searches looking for Higgs production in association with a top-antitop pair at the LHC.

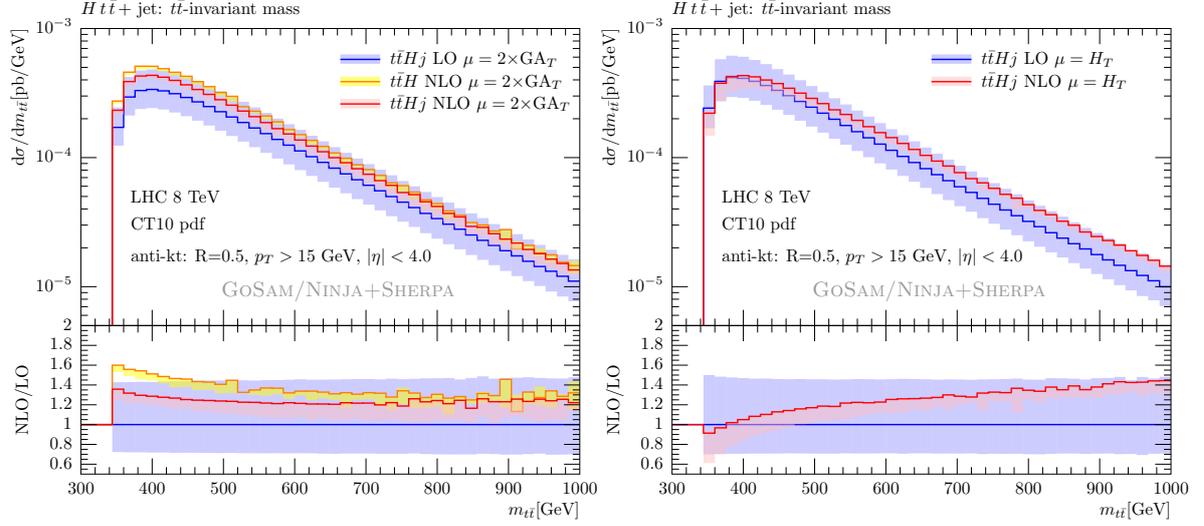


Figure 5.9: Invariant mass distributions of the $t\bar{t}$ -pairs for $t\bar{t}Hj$ at LO and NLO with the two scale choices $\mu = 2GA_T$ (left) and $\mu = H_T$ (right). On the left, we also show the comparison with $Ht\bar{t}$.

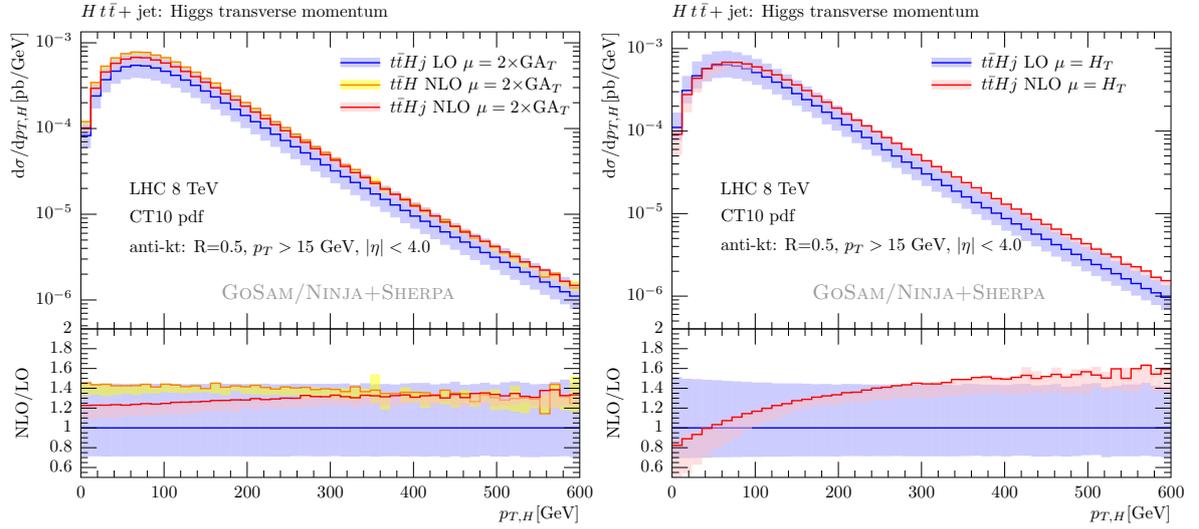


Figure 5.10: Transverse momentum distribution of the Higgs boson at LO and NLO accuracy for $t\bar{t}Hj$ with the two scale choices $\mu = 2GA_T$ (left) and $\mu = H_T$ (right). On the left, we also show the comparison with $Ht\bar{t}$.

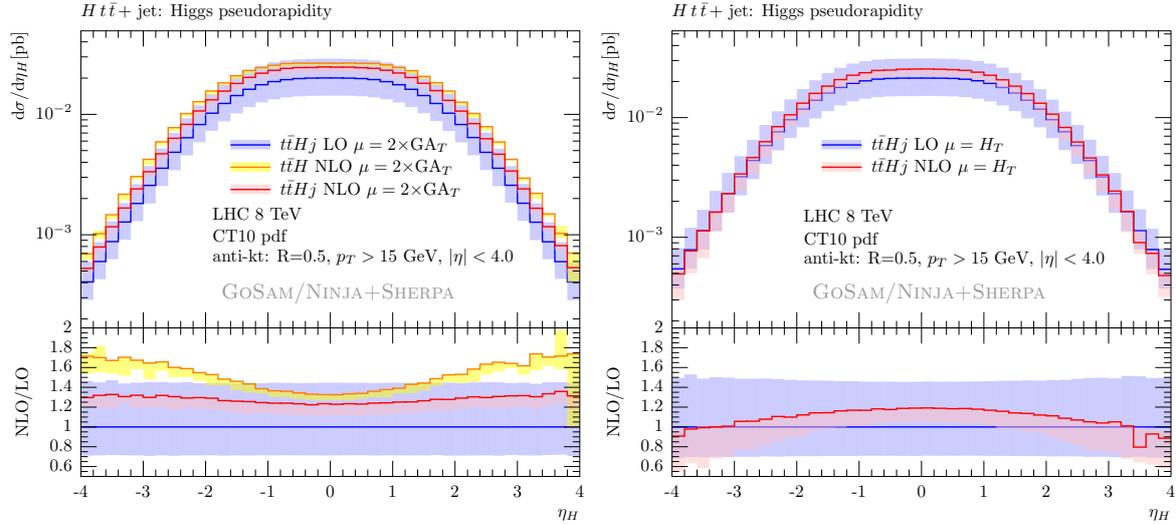


Figure 5.11: Pseudorapidity η of the Higgs boson at LO and NLO accuracy for $t\bar{t}Hj$ with the two scale choices $\mu = 2GA_T$ (left) and $\mu = H_T$ (right). On the left, we also show the comparison with $Ht\bar{t}$.

5.3 Higgs boson production plus two and three jets in gluon fusion

The production mechanism of the Higgs boson with the largest cross section at LHC is Gluon Fusion (GF). As we mentioned in Section 2.2, in this channel the Higgs couples with the gluons via a top-quark loop. In this section we describe the computation of NLO corrections for Higgs boson production in GF in association with two and three jets.

Accurate theoretical predictions for Higgs boson plus jets production can be important for several reasons. The GF channel, while being the dominant one for Higgs production at LHC, is also characterized by a large QCD background. This background can be reduced by applying jet vetoes (i.e. constraints on the observed jets in the final states). A correct estimation of the theoretical uncertainty, when a jet veto is applied, relies in turn on the knowledge of the cross section of Higgs boson production with jets. Another important reason is that Higgs plus jets production in the GF channel is the main background to Higgs production in Vector Boson Fusion (VBF). The latter is particularly important, because it allows to measure the couplings of the Higgs with the electroweak bosons.

The leading order (LO) contribution to the production of a Higgs boson in association with two jets ($H + 2j$) and three jets ($H + 3j$), have been computed in Ref.s [162, 163] and Ref. [164] respectively. These computations retain the full dependence on the top-mass m_t and showed that the top-mass approximation ($m_t \rightarrow \infty$) is valid whenever the mass of the Higgs particle and the p_T of the jets are not significantly larger than the mass of the top

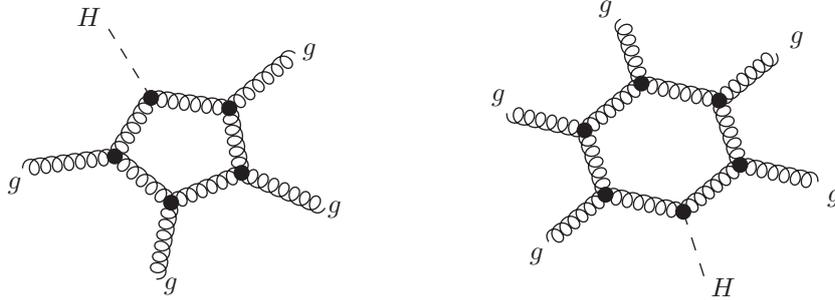


Figure 5.12: Example of diagrams contributing to $H + 2j$ (left) and $H + 3j$ (right) in the all-gluon partonic subprocess. In the infinite to-mass approximation, the Higgs couples with the gluons via an effective vertex.

quark. In this approximation, the top-quark loop which couples the Higgs and the gluons is replaced by an effective local interaction, as described more in detail in the Subsection 5.3.1. The results presented in this thesis will use this approximation.

The NLO corrections for $H+2$ -jets in GF at LHC were first computed in Ref. [165, 166] using amplitudes computed in Ref.s [167–178]. We more recently repeated the computation in Ref. [102], using for the first time an automated tool for the evaluation of both tree-level and loop amplitudes. In this computation we used GOSAM and SHERPA, while the reduction was performed using an extended version [138] of the library SAMURAI, which uses the higher-rank polynomials computed in Ref. [96] and in Eq. (3.28) of this thesis.

In Ref. [103] we computed, for the first time, NLO corrections to $H+3$ -jets in GF. This calculation has been particularly challenging due to the complexity of both the real and the virtual contributions. Results for the cross section have been obtained with a hybrid setup which combines the features of two different Monte Carlo tools. For the generation and integration of the Born and of the virtual contributions, we used an automated framework for fixed order NLO QCD calculations, based on the interplay of GOSAM+SAMURAI and SHERPA [153], where the tree-level matrix elements are obtained with the AMEGIC [154] library. For the integration of the real-radiation terms, the dipole-subtraction terms, and the integrated dipoles, we used a combination of MADGRAPH [179, 180] (matrix elements), MADDIPOLE [181, 182] (subtraction terms), and MADEVENT [183] (numerical integration).

More recently, we presented some new results for $H + 2j$ and $H + 3j$ [101] applying a set of ATLAS-like cuts. For this computation, more recent features of GOSAM were available, including the interface with the C++ library NINJA which has been used for the computation of the one-loop integrals, yielding a significant improvement in both performance and numerical accuracy. As we showed in Section 4.1.1, NINJA can be used for the computation

of higher-rank integrals such as the ones appearing in the processes involving the effective gluon-gluon-Higgs interaction.

5.3.1 The large top-mass limit

As mentioned, the computations for $H + 2j$ and $H + 3j$ are performed in the effective theory defined by the infinite top mass limit ($m_t \rightarrow \infty$). In this limit, the coupling between the gluons and the Higgs, which in the Standard Model is mediated by a massive quark loop, is described by an effective local interaction [184]

$$\mathcal{L}_{\text{eff}} = -\frac{g_{\text{eff}}}{4} H \text{Tr} (G_{\mu\nu} G^{\mu\nu}) \quad (5.6)$$

which couples the gluons and the Higgs boson directly. The effective Feynman rules of this interaction involve couplings between the Higgs and 2, 3 or 4 gluons. They are

$H \text{ --- } \bullet \begin{matrix} \nearrow g_1 \\ \searrow g_2 \end{matrix} = -ig_{\text{eff}} \mathcal{F}_{a_1 a_2}^{\mu_1 \mu_2}$
 $H \text{ --- } \bullet \begin{matrix} \nearrow g_1 \\ \rightarrow g_2 \\ \searrow g_3 \end{matrix} = -ig_{\text{eff}} \mathcal{F}_{a_1 a_2 a_3}^{\mu_1 \mu_2 \mu_3}$
 $H \text{ --- } \bullet \begin{matrix} \nearrow g_1 \\ \rightarrow g_2 \\ \searrow g_3 \\ \downarrow g_4 \end{matrix} = -ig_{\text{eff}} \mathcal{F}_{a_1 a_2 a_3 a_4}^{\mu_1 \mu_2 \mu_3 \mu_4}$

where

$$\begin{aligned}
\mathcal{F}_{a_1 a_2}^{\mu_1 \mu_2} &= \delta_{a_1 a_2} (k_1^{\mu_2} k_2^{\mu_1} - (k_1 \cdot k_2) g^{\mu_1 \mu_2}) \\
\mathcal{F}_{a_1 a_2 a_3}^{\mu_1 \mu_2 \mu_3} &= f_{a_1 a_2 a_3} [g^{\mu_1 \mu_2} (k_1^{\mu_3} - k_2^{\mu_3}) \\
&\quad + g^{\mu_2 \mu_3} (k_2^{\mu_1} - k_3^{\mu_1}) \\
&\quad + g^{\mu_3 \mu_1} (k_3^{\mu_2} - k_1^{\mu_2})] \\
\mathcal{F}_{a_1 a_2 a_3 a_4}^{\mu_1 \mu_2 \mu_3 \mu_4} &= f_{a_1 a_2 b} f_{a_3 a_4 b} [g^{\mu_1 \mu_4} g^{\mu_2 \mu_3} - g^{\mu_1 \mu_3} g^{\mu_2 \mu_4}] \\
&\quad + f_{a_1 a_3 b} f_{a_2 a_4 b} [g^{\mu_1 \mu_4} g^{\mu_2 \mu_3} - g^{\mu_1 \mu_2} g^{\mu_3 \mu_4}] \\
&\quad + f_{a_1 a_4 b} f_{a_2 a_3 b} [g^{\mu_1 \mu_3} g^{\mu_2 \mu_4} - g^{\mu_1 \mu_2} g^{\mu_3 \mu_4}].
\end{aligned} \quad (5.7)$$

The value of the effective coupling, which can be found e.g. from the $m_t \rightarrow \infty$ limit of the $gg \rightarrow H$ scattering amplitude, in the $\overline{\text{MS}}$ scheme is given by [185, 186]

$$g_{\text{eff}} = -\frac{\alpha_s}{3\pi v} \left(1 + \frac{11}{4\pi} \alpha_s \right) + \mathcal{O}(\alpha_s^3). \quad (5.8)$$

As one can see from Eq. (5.7), the Hgg 3-point vertex has rank 2 with respect to the momenta of the interacting particles. This yields loop integrands whose rank is higher than the number of loop denominators.

5.3.2 Numerical computation and results

For $H + 2j$, as in the case of $t\bar{t} + 1j$, the computation has been performed with an automated framework which uses GOSAM for the one-loop amplitudes and SHERPA as Monte Carlo program. As we stated, the two are interfaced using the standard Binoth-Les-Houces Accord. For $H + 3j$, as we mentioned, the very high complexity of the process required a combination of Monte Carlo tools, namely GOSAM+SHERPA for Born and Virtual contributions, and MADGRAPH/MADDIPOLE/MADEVENT for the real emission and subtraction terms. We verified the independence of our result under the variation of the so called α -parameter that fixes the amount of subtractions around the divergences of the real corrections. We first proved the consistency of our hybrid MC integration on $H + 2j$, verifying that the full cross section at NLO agrees with the one obtained using SHERPA and GOSAM alone. Moreover, we found excellent agreement between MADGRAPH and SHERPA for the LO cross section of $H + 3j$ as well.

The results have been computed for proton-proton collisions at LHC with center-of-mass energy $\sqrt{s} = 8$ TeV and the following values of the coupling constants

PARAMETER	VALUE
G_F	$1.16639 \cdot 10^{-5} \text{ GeV}^{-2}$
$\alpha_s^{LO}(m_Z)$	0.129783
$\alpha_s^{NLO}(m_Z)$	0.117981

where $\alpha_s^{LO}(m_Z)$ and $\alpha_s^{NLO}(m_Z)$ are the initial values of α_s at the scale m_Z used for LO and NLO computations respectively. The couplings are thus evolved to the chosen scales of the process using the running coupling equations at one- and two-loop accuracy for LO and NLO respectively. The ultraviolet, the infrared, and the collinear singularities are regularized using *dimensional reduction*. UV divergencies are renormalized in the $\overline{\text{MS}}$ scheme. The jets are clustered using the `antikt`-algorithm implemented in FASTJET [116, 157, 158].

In Appendix D.16 we give, as a reference, benchmarks points for the scattering amplitudes of $H + 3j$, choosing a set of independent partonic subprocesses (from which all the others can be found by crossing or relabeling of quark flavors), obtained using NINJA as reduction library.

In the following we give more details about each specific computation mentioned above.

$H + 2$ jets

Here we give more details about the NLO corrections to $H + 2j$ presented in Ref. [102]. The mass of the Higgs boson is set at $m_H = 125$ GeV. GOSAM automatically identified and

generated the following minimal set of subprocesses

$$gg \rightarrow Hgg, \quad gg \rightarrow Hq\bar{q}, \quad q\bar{q} \rightarrow Hq\bar{q}, \quad q\bar{q} \rightarrow Hq'\bar{q}',$$

while all the others can be obtained from crossing and proper relabeling of q, \bar{q}, q', \bar{q}' with the correct quark flavors.

We use the `cteq6L1` and `cteq6mE` parton distribution functions for the LO and NLO respectively, and the following cuts are applied to the anti- k_t jet algorithm

$$R = 0.5, \quad p_{t,jet} > 20\text{GeV}, \quad |\eta_{jet}| < 4.0. \quad (5.9)$$

The central value for the renormalization and factorization scales μ_R and μ_F is chosen as μ_0 defined by

$$\mu_0 = \hat{H}_T = \sqrt{m_H^2 + p_{t,H}^2} + \sum_{\text{jets } j} |p_{t,j}|. \quad (5.10)$$

The theoretical uncertainty is thus estimated by varying μ_R and μ_F from $\mu_0/2$ to $2\mu_0$.

Within this framework, we find the following total cross sections

$$\sigma_{\text{LO}}[\text{pb}] = 1.90_{-0.41}^{+0.58}, \quad \sigma_{\text{NLO}}[\text{pb}] = 2.90_{-0.20}^{+0.05}.$$

The LO distributions have been computed using 2.5×10^7 phase space points, whereas all NLO distributions have been obtained using 4.0×10^6 phase space points for the Born and the virtual corrections and 5.0×10^8 points for the real radiation for each scale.

In Fig. 5.13, we present the distribution of the transverse momentum p_T of the Higgs boson and its pseudorapidity η , respectively. Both of them show a K-factor between the LO and the NLO distribution of about 1.5 – 1.6, which is almost flat over a large fraction of the kinematic range. Furthermore both plots show a decrease of the scale uncertainty of about 50%. Fig. 5.14 displays the transverse momentum of the first and second jet, whereas their pseudorapidity is shown in Fig. 5.15. The previous considerations are also true for these plots. For the transverse momentum distributions, however, we observe a slight change of shapes. In the case of the leading jet, increasing the p_T , the K-factor decreases from 1.6 to 1.4. For the second leading jet, it increases from 1.4 to 1.6.

$H + 3$ jets

Here we give more details about the setup and the results for $H + 3j$ presented in Ref. [103]. The minimal set of subprocesses identified by GOSAM is

$$gg \rightarrow Hggg, \quad gg \rightarrow Hq\bar{q}g, \quad q\bar{q} \rightarrow Hq\bar{q}g, \quad q\bar{q} \rightarrow Hq'\bar{q}'g.$$

We used the same PDFs as in the $H + 2j$ case, as well as the same cuts given in Eq. (5.9). The central scale for renormalization and factorization has here been chosen as

$$\mu_0 = \frac{\hat{H}_T}{2} = \frac{1}{2} \left(\sqrt{m_H^2 + p_{t,H}^2} + \sum_{\text{jets } j} |p_{t,j}| \right). \quad (5.11)$$

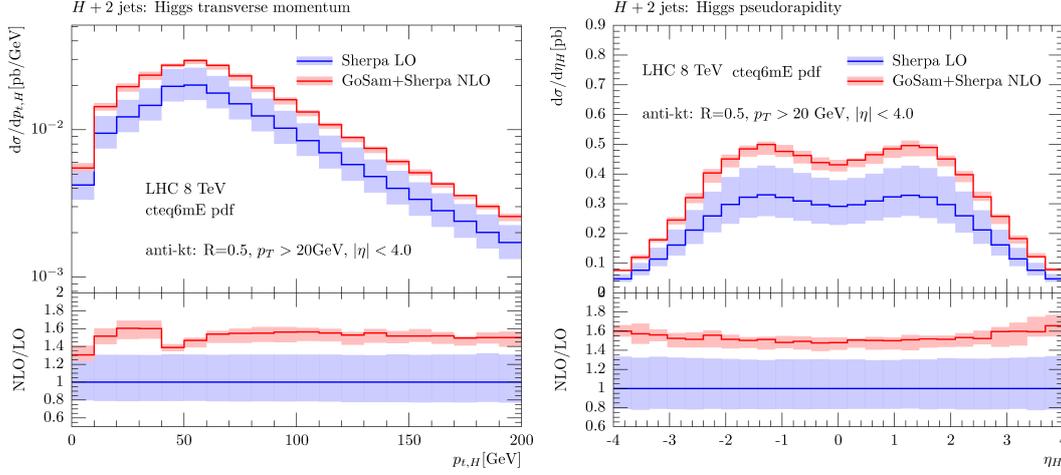


Figure 5.13: Transverse momentum p_T (left) and pseudorapidity η (right) of the Higgs boson in $H + 2j$.

The strong coupling is evaluated at different scales according to

$$\alpha_s^5 \rightarrow \alpha_s^2(m_H) \alpha_s^3(\mu_R).$$

As usual, the theoretical uncertainty is estimated by varying μ_R and μ_F from $\mu_0/2$ to $2\mu_0$.

Within this setup we obtain the following total cross section at LO and NLO:

$$\sigma_{\text{LO}}[\text{pb}] = 0.962_{-0.31}^{+0.51}, \quad \sigma_{\text{NLO}}[\text{pb}] = 1.18_{-0.22}^{+0.01}.$$

The scale dependence of the total cross section, depicted in Fig. 5.16, is strongly reduced by the inclusion of the NLO contributions.

In Fig. 5.17, we show the p_T distributions of the three jets and of the Higgs boson. The NLO corrections enhance all distributions for p_T values lower than 150 – 200 GeV, whereas their contribution is negative at higher p_T . This behavior is explicitly shown in the lower part of the distribution on the right of Fig. 5.17 for the case of the Higgs boson.

$H + 2$ and $H + 3$ jets with ATLAS-like cuts

In Ref. [101] some new analysis for $H + 2j$ and $H + 3j$ were presented. The main difference with the setup described above is the choice of ATLAS-like cuts,

$$R = 0.4, \quad p_{t,jet} > 30\text{GeV}, \quad |\eta_{jet}| < 4.4. \quad (5.12)$$

Moreover, since this calculation is more recent, it could use the interface between GOSAM and the NINJA library for the computation of one-loop integrals. This resulted in a reduction of the computational time of about 50% for the computation of a scattering amplitude in a

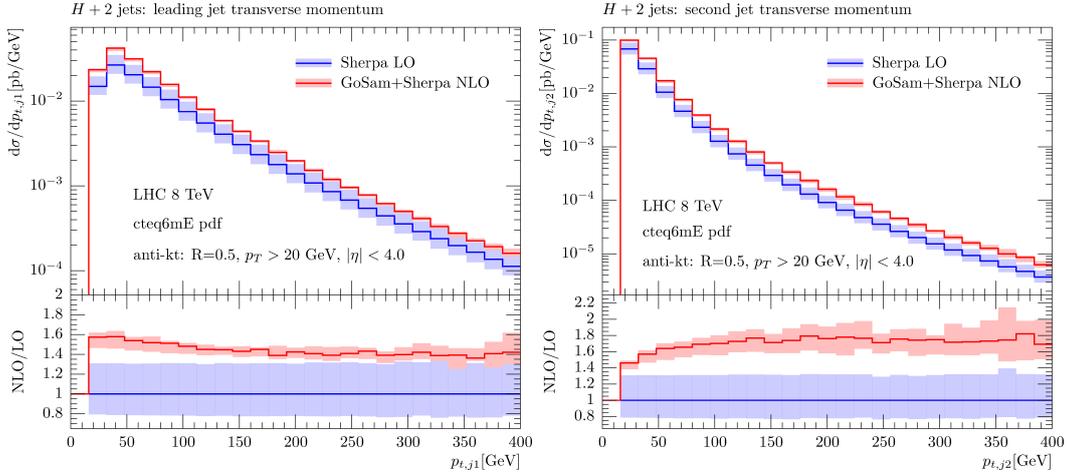


Figure 5.14: Transverse momentum p_T of the first (left) and the second (right) jet in $H + 2j$.

single phase-space point. The stability also improved, giving a rate of unstable points below one per mill, which in turn yields a further increase in performance because of a less frequent call of the rescue system. The Higgs mass is set to $m_H = 126$ GeV and central scale μ_0 for μ_R and μ_F is chosen according to Eq. (5.11).

With this setup we obtain the following total cross sections

$$\begin{aligned} \sigma_{LO}^{(H+2j)}([\text{pb}]) &= 1.23^{+37\%}_{-24\%}, & \sigma_{LO}^{(H+3j)}([\text{pb}]) &= 0.381^{+53\%}_{-32\%}, \\ \sigma_{NLO}^{(H+2j)}([\text{pb}]) &= 1.590^{-4\%}_{-7\%}, & \sigma_{NLO}^{(H+3j)}([\text{pb}]) &= 0.485^{-3\%}_{-13\%}. \end{aligned}$$

In Ref. [101] we also show several preliminary distribution. As an example, we plot in Fig. 5.18 the rapidity distribution of the Higgs for $H + 3j$.

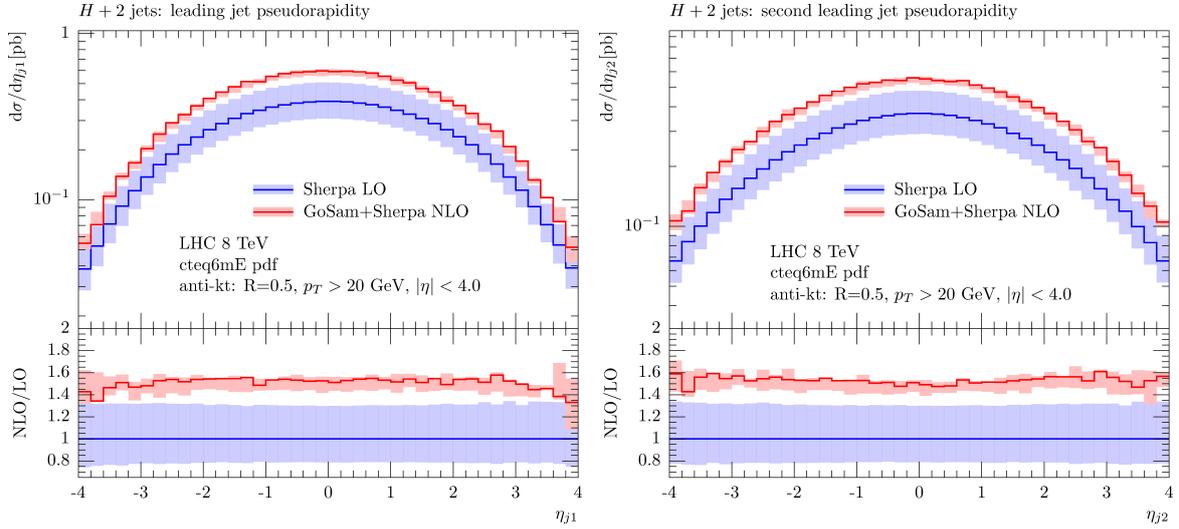


Figure 5.15: Pseudorapidity η of the first (left) and the second (right) jet in $H + 2j$.

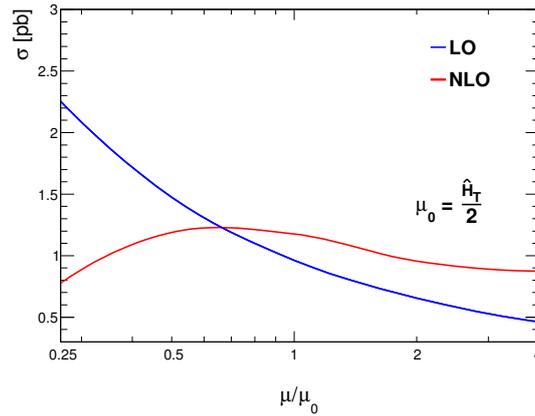


Figure 5.16: Scale dependence for $H + 3j$ of the total cross section at LO and NLO.

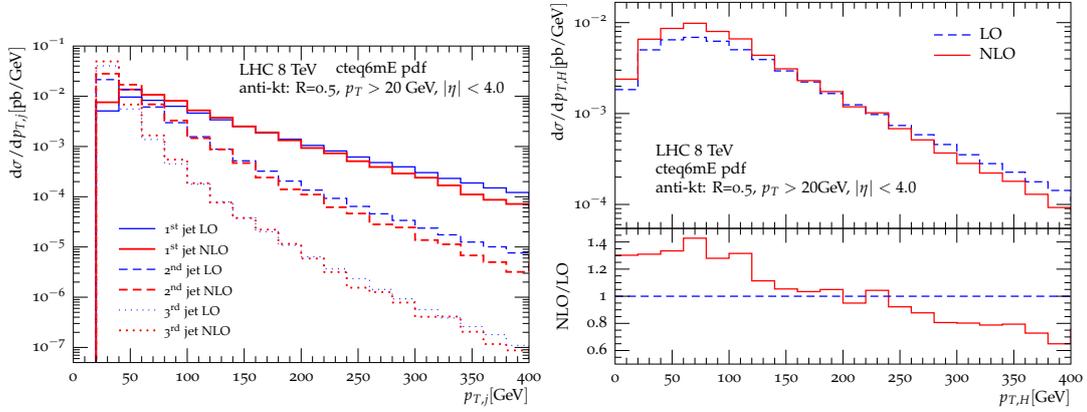


Figure 5.17: Distributions for $H + 3j$. On the left, the transverse momentum (p_T) of the first, second, and third leading jet. On the right, the transverse momentum (p_T) of the Higgs boson.

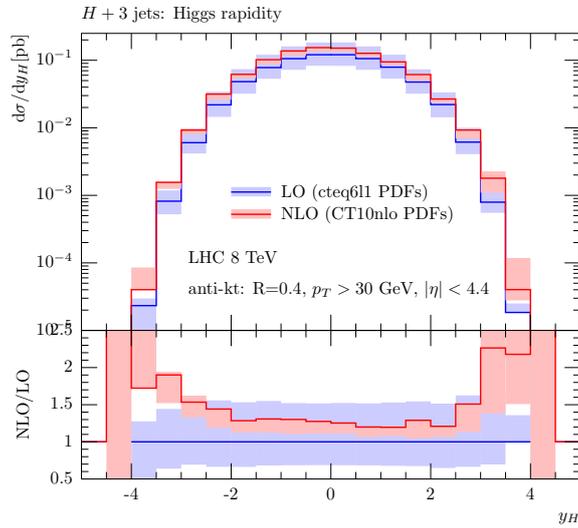


Figure 5.18: Higgs rapidity distribution for $H + 3j$, using ATLAS-like cuts.

Chapter 6

Integrand reduction at higher loops

In this chapter, we show some applications of the general integrand reduction method described in Chapter 3 to higher-loop diagrams. In Section 6.1 we show explicit two-loop applications of the *fit-on-the-cut approach* described in Section 3.3.1. In Section 6.2 we use instead the *divide-and-conquer approach* to perform the reduction of several higher-loop integrands, some of which cannot be treated with other unitarity-based approaches and integrand-reduction methods, because of the presence of higher powers of loop denominators in the integrands.

6.1 Fit on the cut at two loops

In this section we consider two-loop amplitudes in four dimensions. In this case, as we observed in Section 3.2, the irreducible polynomial residues entering the integrand decomposition can have a maximum of eight denominators, as a consequence of the *reducibility criterion*¹. Hence, the most general decomposition of an integrand is a special case of Eq. (3.6), namely

$$\mathcal{I}_{i_1 \dots i_n} \equiv \frac{\mathcal{N}_{i_1 \dots i_n}}{D_{i_1} \dots D_{i_n}} = \sum_{k=0}^8 \sum_{\{j_1, \dots, j_k\}} \frac{\Delta_{j_1 \dots j_k}}{D_{j_1} \dots D_{j_k}}, \quad (6.1)$$

or equivalently the numerator can be written as

$$\mathcal{N}_{i_1 \dots i_n}(q_1, q_2) = \sum_{k=0}^8 \sum_{\{j_1 \dots j_k\}} \Delta_{j_1 \dots j_k} \prod_{h \in \{i_1 \dots i_n\} \setminus \{j_1 \dots j_k\}} D_h. \quad (6.2)$$

The fit-on-the-cut approach can be applied once the parametric form of the residues is known. This can be found as the most general remainder of a polynomial division modulo a Gröbner basis of the ideal generated by the denominators of each residue.

¹A potential ambiguity on the application of the *reducibility criterion* may arise in topologies with nine denominators two of which are degenerate, so that only eight are actually independent. However, in this case the one-loop sub-topology contains at least six denominators, yielding thus a system of equations for the corresponding cut with no solution, and the criterion still applies.

For each sub-set $\{j_1, \dots, j_k\}$ of denominators corresponding to a residue to be computed, we proceed as follows:

1. We decompose the two loop momenta q_1 and q_2 in terms of suitably chosen four-dimensional bases $\{e_i\}_{i=1}^4$ and $\{\tau_i\}_{i=1}^4$ respectively as

$$\begin{aligned} q_1^\mu &= -r_1^\mu + x_1 e_1^\mu + x_2 e_2^\mu + x_3 e_3^\mu + x_4 e_4^\mu \\ q_2^\mu &= -r_2^\mu + y_1 \tau_1^\mu + y_2 \tau_2^\mu + y_3 \tau_3^\mu + y_4 \tau_4^\mu, \end{aligned} \quad (6.3)$$

so that $\mathcal{N} = \mathcal{N}(\mathbf{z})$, with $\mathbf{z} = (z_1, \dots, z_8) = (x_1, x_2, x_2, x_4, y_1, y_2, y_3, y_4)$ up to a choice of order of the variables.

2. We define the ideal $\mathcal{J}_{j_1 \dots j_k}$ and a Gröbner basis $\mathcal{G}_{j_1 \dots j_k}$ generated by the loop denominators $D_{j_1}, D_{j_2}, \dots, D_{j_k}$ as in Eq. (3.5).
3. We fix a maximum rank r_{\max} and we consider the most general numerator allowed by the theory

$$\mathcal{N}_{j_1 \dots j_k}(\mathbf{z}) = \sum_{\vec{r} \in R(r_{\max})} \alpha_{\vec{r}} \left(\prod_{i=1}^8 z_i^{r_i} \right), \quad R(r_{\max}) \equiv \left\{ \vec{r} \in \mathbb{N}^8 : \sum_{i=1}^8 r_i \leq r_{\max} \right\}. \quad (6.4)$$

The parametric form of the residue $\Delta_{j_1 \dots j_k}$ is thus the most general remainder of a generic numerator of the form of Eq. (6.4) and the Gröbner basis $\mathcal{G}_{j_1 \dots j_k}$.

4. For every phase-space point, we compute the coefficients which parametrize each residue by evaluating Eq. (6.2) on the solutions of the multiple cut $D_{j_1} = \dots = D_{j_k}$ and solving the resulting system of equations.

It is worth noticing that, depending on the rank of the numerator, one may also have a term Δ_\emptyset in Eq. (6.1), which is the sum of the quotients of the last step of the reduction. This term is spurious and does not need to be computed. It can however be determined by evaluating the integrand outside the solutions of the multiple cuts. In the examples presented in this thesis, this kind of terms are never present in the decomposition.

After integration, some terms of the integrand decomposition vanish, while the others give Master Integrals.

6.1.1 Integrands and parametric residues

The integrands we consider for these examples are contributions to 5-point amplitudes in $\mathcal{N} = 4$ Super YangMills (SYM) and $\mathcal{N} = 8$ Supergravity (SUGRA). The integrand reduction of these examples was first addressed in Ref. [91] and then in Ref. [187] using a formalism closer to the one presented in this thesis. The integrands of these theories are particularly simple and are therefore suited to be used as examples and first applications of new methods.

In this thesis, we do not intend to discuss the details of such theories, whose integrands are only used as toy examples for the reduction methods we illustrated.

The five-point amplitudes in $\mathcal{N} = 4$ SYM and $\mathcal{N} = 8$ SUGRA can be expressed in terms of six diagrams [188]. The color-ordered amplitude is given by a sum over the cyclic permutations of the external momenta. We apply the integrand reduction only to the three diagrams depicted in Fig. 6.1, which are the only ones with a non-trivial reduction. The other three diagrams are instead already expressed in terms of scalar integrals, since their numerator is independent of the loop momenta. Each diagram in Fig. 6.1 will be reduced using the fit-on-the-cut approach, following the strategy outlined above.

In the case of $\mathcal{N} = 4$ SYM, the integrands of the three diagrams in Fig. 6.1, with kinematics $k_1, k_2, k_3, k_4, k_5 \rightarrow 0$, can be written as [188]

$$\mathcal{I}_{1\dots 8}^{(4,a)}(q_1, q_2) = \frac{\mathcal{N}_{1\dots 8}^{(4,a)}(q_1, q_2)}{D_1 \cdots D_8}, \quad \mathcal{N}_{1\dots 8}^{(4,a)}(q_1, q_2) = 2 q_1 \cdot v_1 + \beta_1, \quad (6.5)$$

$$\mathcal{I}_{1\dots 8}^{(4,b)}(q_1, q_2) = \frac{\mathcal{N}_{1\dots 8}^{(4,b)}(q_1, q_2)}{D_1 \cdots D_8}, \quad \mathcal{N}_{1\dots 8}^{(4,b)}(q_1, q_2) = 2 q_1 \cdot v_1 + \beta_1, \quad (6.6)$$

$$\mathcal{I}_{1\dots 8}^{(4,c)}(q_1, q_2) = \frac{\mathcal{N}_{1\dots 8}^{(4,c)}(q_1, q_2)}{D_1 \cdots D_8}, \quad \mathcal{N}_{1\dots 8}^{(4,c)}(q_1, q_2) = 2 q_1 \cdot v_2 + 2 q_2 \cdot v_3 + \beta_2 + \beta_3, \quad (6.7)$$

respectively, where the vectors v_i^μ and the constants β_i are defined as

$$v_1^\mu = \frac{1}{4} \left(\gamma_{35124} (k_5^\mu - k_3^\mu) + \gamma_{34125} (k_4^\mu - k_3^\mu) + \gamma_{45123} (k_5^\mu - k_4^\mu) + 2 \gamma_{12345} (k_2^\mu - k_1^\mu) \right) \quad (6.8)$$

$$v_2^\mu = \frac{1}{4} \left(\gamma_{23145} (k_2^\mu - k_3^\mu) + \gamma_{24135} (k_2^\mu - k_4^\mu) + \gamma_{34125} (k_3^\mu - k_4^\mu) + 2 \gamma_{15234} (k_1^\mu - k_5^\mu) \right) \quad (6.9)$$

$$v_3^\mu = \frac{1}{4} \left(\gamma_{12345} (k_1^\mu - k_2^\mu) + \gamma_{25134} (k_2^\mu - k_5^\mu) + \gamma_{15234} (k_1^\mu - k_5^\mu) + 2 \gamma_{34125} (k_3^\mu - k_4^\mu) \right) \quad (6.10)$$

$$\beta_1 = \frac{1}{4} \left(\gamma_{35124} (s_{34} + s_{12} + s_{35}) + 2 \gamma_{34125} s_{12} + \gamma_{45123} (s_{34} + s_{12} + s_{35}) + 2 \gamma_{12345} (s_{23} - s_{13}) \right) \quad (6.11)$$

$$\beta_2 = \frac{1}{4} \left(-(\gamma_{23145} + \gamma_{24135}) s_{23} + \gamma_{34125} (s_{15} + s_{34} + 2 s_{23}) - 2 \gamma_{15234} (s_{13} - s_{35}) \right) \quad (6.12)$$

$$\beta_3 = \frac{1}{4} \left((\gamma_{12345} - \gamma_{25134}) s_{12} + \gamma_{15234} (s_{34} + s_{15} + 2 s_{12}) - 2 \gamma_{34125} (s_{13} - s_{14}) \right). \quad (6.13)$$

The kinematic invariants s_{ij} and the functions γ are defined as

$$s_{ij} \equiv (k_i + k_j)^2 = 2 (k_i \cdot k_j) \quad (6.14)$$

$$\gamma_{12345} \equiv \left(\frac{[1\ 2][2\ 3][3\ 4][4\ 5][5\ 1]}{[1\ 4][2\ 3]\langle 1\ 2\rangle\langle 3\ 4\rangle - [1\ 2][3\ 4]\langle 1\ 4\rangle\langle 2\ 3\rangle} \right) - (1 \leftrightarrow 2), \quad (6.15)$$

in terms of spinor products (see Appendix A) built from the massless external momenta k_i .

In $\mathcal{N} = 4$ SYM, given the simple form of the numerators, the multi-pole decomposition of the integrands only requires one iteration. The numerators can be decomposed as

$$\mathcal{N}_{1\dots 8}^{(4,x)}(q, k) = \Delta_{12345678} + \sum_{i=1}^7 \Delta_{1\dots(i-1)(i+1)\dots 8} D_i, \quad x = a, b, c. \quad (6.16)$$

One can easily check that, given the simple form of the numerator expressions, the number of 7-ple residues of the integrands $\mathcal{N}^{(4,a)}$ and $\mathcal{N}^{(4,b)}$ is almost halved since $\Delta_{1\dots(i-1)(i+1)\dots 8} = 0$ for $i \neq 4, 5, 6, 7$.

In $\mathcal{N} = 8$ SUGRA, the numerator of each integrand is obtained by squaring the corresponding numerator in $\mathcal{N} = 4$ SYM, as shown in [188]. The numerators are of rank two in the loop momenta. One can easily show that their decomposition can be expressed in terms of 8-, 7-, and 6-denominator integrands,

$$\begin{aligned} \mathcal{N}_{1\dots 8}^{(8,x)}(q_1, q_2) &= \Delta_{12345678} + \sum_{i=1}^8 \Delta_{1\dots(i-1)(i+1)\dots 8} D_i \\ &+ \sum_{ij}^8 \Delta_{1\dots(i-1)(i+1)\dots(j-1)(j+1)\dots 8} D_i D_j, \quad x = a, b, c. \end{aligned} \quad (6.17)$$

The corresponding decomposition for the integrands $\mathcal{I}_{1\dots 8}^{(4,a)}$, $\mathcal{I}_{1\dots 8}^{(4,b)}$ and $\mathcal{I}_{1\dots 8}^{(4,c)}$ reads

$$\begin{aligned} \mathcal{I}_{1\dots 8}^{(4,x)}(q_1, q_2) &= \frac{\Delta_{12345678}}{D_1 \dots D_8} + \sum_{i=1}^8 \frac{\Delta_{1\dots(i-1)(i+1)\dots 8}}{\prod_{h \neq i}^8 D_h} \\ &+ \sum_{ij}^8 \frac{\Delta_{1\dots(i-1)(i+1)\dots(j-1)(j+1)\dots 8}}{\prod_{h \neq i,j}^8 D_h}, \quad x = a, b, c. \end{aligned} \quad (6.18)$$

Since the numerators $\mathcal{N}_{1\dots 8}^{(8,a)}$ and $\mathcal{N}_{1\dots 8}^{(8,b)}$ are of rank two in q and independent of k , their decomposition is significantly simplified. Indeed in these cases $\Delta_{1\dots(i-1)(i+1)\dots 8} = 0$ for $i \neq 4, 5, 6, 7$ and $\Delta_{1\dots(i-1)(i+1)\dots(j-1)(j+1)\dots 8} = 0$ for $i, j \neq 4, 5, 6, 7$. Since the rank of the numerator is equal to 2, one can easily check that all the 6-point residues are constants

$$\Delta_{i_1 \dots i_6} = c_{i_1 \dots i_6, 0}. \quad (6.19)$$

In the following, we list the parametrization of the residues entering our computation, namely the of 8-ple and 7-ple cuts. All the residues of 8-ple cuts are related to *maximum cuts*. According to the maximum-cut theorem (see Section 3.2), the number of coefficients needed to parametrize these residues is finite and equal to the number of the solutions of the corresponding cut. For each diagram, we found the most general parametrization of the 8-ple cut residue, which is process-independent and valid for numerators of any rank in both q_1 and q_2 . The parametrization of the 7-ple residues is also process-independent and it is given for the case of renormalizable numerators of rank six at most, which is more than we need for the applications presented in this section, although extending them to arbitrarily high ranks is straightforward. Since, as we already observed in Chapter 3, the parametric form of the residues is universal, the same can be used for both $\mathcal{N} = 4$ SYM and $\mathcal{N} = 8$ SUGRA.

In Table 6.1 and 6.2 we show the parametrization of the residues of the diagrams in Fig. 6.1a and 6.1b respectively. For each multiple cut $\{j_1, \dots, j_k\}$, the corresponding parametric residue is a generic linear combinations of the monomials $\mathcal{S}_{j_1 \dots j_k}$ listed there, in terms

cut	bases	\mathbf{z}	Monomials in the residue
(12345678)	Eq. (6.20)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{12345678} = \{1, x_1, y_1, y_2\}$
(1234568)	Eq. (6.20)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234568} = \{1, x_1, x_1^2, x_1^3, x_1^4, x_2, x_1x_2, x_1^2x_2, x_1^3x_2, y_1, x_1y_1, x_1^2y_1, x_1^3y_1, x_1^4y_1, x_2y_1, x_1x_2y_1, x_1^2x_2y_1, x_1^3x_2y_1, y_1^2, x_1y_1^2, x_2y_1^2, y_1^3, x_1y_1^3, x_2y_1^3, y_1^4, x_1y_1^4, x_2y_1^4, y_2, x_1y_2, y_1y_2, y_1^2y_2, y_1^3y_2\}$
(1234678)	Eq. (6.20)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234678} = \mathcal{S}_{1234568}$
(1234578)	Eq. (6.21)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234578} = \{1, x_2, x_2^2, x_2^3, x_2^4, x_4, x_2x_4, x_2^2x_4, x_2^3x_4, y_1, x_2y_1, x_2^2y_1, x_2^3y_1, x_2^4y_1, x_4y_1, x_2x_4y_1, x_2^2x_4y_1, x_2^3x_4y_1, y_1^2, x_2y_1^2, x_4y_1^2, y_1^3, x_2y_1^3, x_4y_1^3, y_1^4, x_2y_1^4, x_4y_1^4, y_4, x_2y_4, y_1y_4, y_1^2y_4, y_1^3y_4\}$
(1235678)	Eq. (6.22)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1235678} = \{1, x_1, x_1^2, x_1^3, x_1^4, x_2, x_1x_2, x_1^2x_2, x_1^3x_2, y_1, x_1y_1, x_1^2y_1, x_1^3y_1, x_1^4y_1, x_2y_1, x_1x_2y_1, x_1^2x_2y_1, x_1^3x_2y_1, y_1^2, x_1y_1^2, x_2y_1^2, y_1^3, x_1y_1^3, x_2y_1^3, y_1^4, x_1y_1^4, x_2y_1^4, y_3, x_1y_3, y_1y_3, y_1^2y_3, y_1^3y_3\}$

Table 6.1: Set of monomials which parametrize the residues entering the decomposition of the five-point pentabox diagram in Fig. 6.1. They have all been found using degree lexicographic monomial ordering. For each cut the bases and the chosen ordering for loop variables are shown as well.

cut	bases	\mathbf{z}	Monomials in the residue
(12345678)	Eq. (6.20)	$(x_4, x_3, x_2, y_3, y_4, x_1, y_2, y_1)$	$\mathcal{S}_{12345678} = \{1, x_1, y_1, y_2\}$
(1234568)	Eq. (6.20)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234568} = \{1, x_1, x_1^2, x_1^3, x_1^4, x_1^5, x_1^6, x_2,$ $x_1x_2, x_1^2x_2, x_1^3x_2, x_1^4x_2, x_1^5x_2,$ $y_1, x_1y_1, x_1^2y_1, x_1^3y_1, x_1^4y_1, x_1^5y_1, x_2y_1,$ $x_1x_2y_1, x_1^2x_2y_1, x_1^3x_2y_1, x_1^4x_2y_1, y_1^2, x_1y_1^2,$ $x_2y_1^2, y_1^3, x_1y_1^3, x_2y_1^3, y_1^4, x_1y_1^4, x_2y_1^4, y_2,$ $x_1y_2, y_1y_2, y_1^2y_2, y_1^3y_2\}$
(1234678)	Eq. (6.20)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234678} = \mathcal{S}_{1234568}$
(1234578)	Eq. (6.21)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1234578} = \{1, x_2, x_2^2, x_2^3, x_2^4, x_2^5, x_2^6, x_4,$ $x_2x_4, x_2^2x_4, x_2^3x_4, x_2^4x_4, x_2^5x_4, y_1, x_2y_1,$ $x_2^2y_1, x_2^3y_1, x_2^4y_1, x_2^5y_1, x_4y_1, x_2x_4y_1, x_2^2x_4y_1,$ $x_2^3x_4y_1, x_2^4x_4y_1, y_1^2, x_2y_1^2, x_4y_1^2, y_1^3, x_2y_1^3,$ $x_4y_1^3, y_1^4, x_2y_1^4, x_4y_1^4, y_4, x_2y_4, y_1y_4, y_1^2y_4, y_1^3y_4\}$
(1235678)	Eq. (6.22)	$(y_4, y_2, y_3, y_1, x_4, x_3, x_2, x_1)$	$\mathcal{S}_{1235678} = \{1, x_1, x_1^2, x_1^3, x_1^4, x_1^5, x_1^6, x_2,$ $x_1x_2, x_1^2x_2, x_1^3x_2, x_1^4x_2, x_1^5x_2, y_1,$ $x_1y_1, x_1^2y_1, x_2y_1, x_1x_2y_1, y_1^2, x_1y_1^2, x_1^2y_1^2,$ $x_2y_1^2, x_1x_2y_1^2, y_1^3, x_1y_1^3, x_1^2y_1^3, x_2y_1^3, x_1x_2y_1^3,$ $y_1^4, x_1y_1^4, x_1^2y_1^4, x_2y_1^4, x_1x_2y_1^4, y_3, x_1y_3,$ $y_1y_3, y_1^2y_3, y_1^3y_3\}$

Table 6.2: The same as Table 6.1, but for the five-point crossed pentabox diagram in Fig. 6.1.

cut	bases	\mathbf{z}	Monomials in the residue
(12345678)	Eq. (6.24)	$(y_4, y_3, y_2, y_1, x_4, x_3, x_1, x_2)$	$\mathcal{S}_{12345678} = \{1, y_1, x_2, y_1x_2, x_2^2, x_2^3, x_1, x_2x_1\}$
(1345678)	Eq. (6.24)	$(x_4, x_3, x_2, x_1, y_4, y_3, y_2, y_1)$	$\mathcal{S}_{1345678} = \{1, y_1, y_1^2, y_1^3, y_1^4, y_1^5, y_1^6, y_2, y_1y_2, y_1^2y_2, y_1^3y_2, y_1^4y_2, y_1^5y_2, x_1, y_1x_1, y_1^2x_1, y_1^3x_1, y_1^4x_1, y_1^5x_1, y_2x_1, y_1y_2x_1, y_1^2y_2x_1, y_1^3y_2x_1, y_1^4y_2x_1, x_1^2, y_1x_1^2, y_2x_1^2, x_1^3, y_1x_1^3, y_2x_1^3, x_1^4, y_1x_1^4, y_2x_1^4, x_2, y_1x_2, x_1x_2, x_1^2x_2, x_1^3x_2\}$
(1245678)	Eq. (6.24)	$(x_4, x_3, x_2, x_1, y_4, y_3, y_2, y_1)$	$\mathcal{S}_{1245678} = \mathcal{S}_{1345678}$
(2345678)	Eq. (6.25)	$(x_1, x_3, x_2, x_4, y_3, y_4, y_2, y_1)$	$\mathcal{S}_{2345678} = \{1, y_1, y_1^2, y_1^3, y_1^4, y_1^5, y_1^6, y_4, y_1y_4, y_1^2y_4, y_1^3y_4, y_1^4y_4, y_1^5y_4, x_2, y_1x_2, x_4, y_1x_4, y_1^2x_4, y_1^3x_4, y_1^4x_4, y_1^5x_4, y_4x_4, y_1y_4x_4, y_1^2y_4x_4, y_1^3y_4x_4, y_1^4y_4x_4, x_2x_4, x_4^2, y_1x_4^2, y_4x_4^2, x_2x_4^2, x_4^3, y_1x_4^3, y_4x_4^3, x_2x_4^3, x_4^4, y_1x_4^4, y_4x_4^4\}$,
(1234567)	Eq. (6.24)	$(x_4, x_3, x_2, x_1, y_4, y_3, y_2, y_1)$	$\mathcal{S}_{1234567} = \{1, y_1, y_1^2, y_1^3, y_1^4, y_2, y_1y_2, y_1^2y_2, y_1^3y_2, x_1, y_1x_1, y_1^2x_1, y_1^3x_1, y_1^4x_1, y_2x_1, y_1y_2x_1, y_1^2y_2x_1, y_1^3y_2x_1, x_1^2, y_1x_1^2, y_2x_1^2, x_1^3, y_1x_1^3, y_2x_1^3, x_1^4, y_1x_1^4, y_2x_1^4, x_2, y_1x_2, x_1x_2, x_1^2x_2, x_1^3x_2\}$

Table 6.3: The same as Table 6.1, but for the five-point double pentagon diagram in Fig. 6.1.

of the parametrization of the loop momentum given in Eq. (6.3) and the following bases

$$\begin{cases} r_1^\mu = 0, & e_1^\mu = k_3^\mu, & e_2^\mu = k_4^\mu, & e_3^\mu = \frac{\langle 3|\gamma^\mu|4\rangle}{2}, & e_4^\mu = \frac{\langle 4|\gamma^\mu|3\rangle}{2}, \\ r_2^\mu = 0, & \tau_1^\mu = k_2^\mu, & \tau_2^\mu = k_1^\mu, & \tau_3^\mu = \frac{\langle 2|\gamma^\mu|1\rangle}{2}, & \tau_4^\mu = \frac{\langle 1|\gamma^\mu|2\rangle}{2}, \\ x_1 = \frac{(q_1 \cdot k_1)}{(k_1 \cdot k_2)}, & x_2 = \frac{(q_1 \cdot k_2)}{(k_1 \cdot k_2)}, & y_1 = \frac{(q_2 \cdot k_4)}{(k_3 \cdot k_4)}, & y_2 = \frac{(q_2 \cdot k_3)}{(k_3 \cdot k_4)}; \end{cases} \quad (6.20)$$

$$\begin{cases} r_1^\mu = 0, & e_1^\mu = k_1^\mu, & e_2^\mu = k_3^\mu, & e_{3,4}^\mu = \frac{\langle 3|2|1\rangle\langle 1|\gamma^\mu|3\rangle \pm \langle 1|2|3\rangle\langle 3|\gamma^\mu|1\rangle}{4}, \\ r_2^\mu = 0, & \tau_1^\mu = k_1^\mu, & \tau_2^\mu = k_3^\mu, & \tau_{3,4}^\mu = \frac{\langle 3|2|1\rangle\langle 1|\gamma^\mu|3\rangle \pm \langle 1|2|3\rangle\langle 3|\gamma^\mu|1\rangle}{4}, \\ x_1 = \frac{(q_1 \cdot k_3)}{(k_1 \cdot k_3)}, & x_4 = \frac{(q_1 \cdot \tau_4)}{\tau_4^2}, & y_1 = \frac{(q_2 \cdot k_3)}{(k_1 \cdot k_3)}, & y_4 = \frac{(q_2 \cdot e_4)}{e_4^2}; \end{cases} \quad (6.21)$$

$$\begin{cases} r_1^\mu = 0, & e_{1,4}^\mu = \frac{\langle 3|2|1\rangle\langle 1|\gamma^\mu|3\rangle \mp \langle 1|2|3\rangle\langle 3|\gamma^\mu|1\rangle}{4}, & e_2^\mu = k_3^\mu, & e_3^\mu = k_1^\mu, \\ r_2^\mu = -k_4^\mu, & \tau_{1,4}^\mu = \frac{\langle 3|2|1\rangle\langle 1|\gamma^\mu|3\rangle \mp \langle 1|2|3\rangle\langle 3|\gamma^\mu|1\rangle}{4}, & \tau_2^\mu = k_3^\mu, & \tau_3^\mu = k_1^\mu, \\ x_1 = \frac{((q_1 - k_4) \cdot e_1)}{e_1^2}, & x_2 = \frac{((q_1 - k_4) \cdot k_1)}{(k_1 \cdot k_3)}, & y_1 = \frac{(q_2 \cdot \tau_1)}{\tau_1^2}, & y_3 = \frac{(q_2 \cdot k_3)}{(k_1 \cdot k_3)}. \end{cases} \quad (6.22)$$

The integrand of the double pentagon diagram in Fig. 6.1c is symmetric under the transformation

$$k_1^\mu \leftrightarrow k_3^\mu, \quad k_4^\mu \leftrightarrow k_5^\mu, \quad q_1^\mu \leftrightarrow q_2^\mu. \quad (6.23)$$

Therefore, we limit ourselves to list the parametrization of the 8-ple cuts and the 7-ple residues given in Table 6.3, with the bases for the loop momentum defined as

$$\begin{cases} r_1^\mu = 0, & e_1^\mu = k_4^\mu, & e_2^\mu = k_3^\mu, & e_3^\mu = \frac{\langle 4|\gamma^\mu|3\rangle}{2}, & e_4^\mu = \frac{\langle 3|\gamma^\mu|4\rangle}{2}, \\ r_2^\mu = 0, & \tau_1^\mu = k_5^\mu, & \tau_2^\mu = k_1^\mu, & \tau_3^\mu = \frac{\langle 5|\gamma^\mu|1\rangle}{2}, & \tau_4^\mu = \frac{\langle 1|\gamma^\mu|5\rangle}{2}, \\ x_1 = \frac{(q_1 \cdot k_1)}{(k_5 \cdot k_1)}, & x_2 = \frac{(q_1 \cdot k_5)}{(k_5 \cdot k_1)}, & y_1 = \frac{(q_2 \cdot k_3)}{(k_3 \cdot k_4)}, & y_2 = \frac{(q_2 \cdot k_4)}{(k_3 \cdot k_4)}; \end{cases} \quad (6.24)$$

$$\begin{cases} r_1^\mu = 0, & e_1^\mu = k_1^\mu, & e_2^\mu = k_3^\mu, & e_{3,4}^\mu = \frac{\langle 1|4|3\rangle\langle 3|\gamma^\mu|1\rangle \pm \langle 3|4|1\rangle\langle 1|\gamma^\mu|3\rangle}{4}, \\ r_2^\mu = -k_3^\mu, & \tau_1^\mu = k_1^\mu, & \tau_2^\mu = k_3^\mu, & \tau_{3,4}^\mu = \frac{\langle 1|4|3\rangle\langle 3|\gamma^\mu|1\rangle \pm \langle 3|4|1\rangle\langle 1|\gamma^\mu|3\rangle}{4}, \\ x_2 = \frac{((q_1 - k_3) \cdot k_1)}{(k_1 \cdot k_3)}, & x_4 = \frac{((q_1 - k_3) \cdot \tau_4)}{\tau_4^2}, & y_1 = \frac{(q_2 \cdot k_3)}{(k_1 \cdot k_3)}, & y_4 = \frac{(q_2 \cdot e_4)}{e_4^2}. \end{cases} \quad (6.25)$$

All the others 7-ple cuts can be easily obtained from the ones we listed, using the transformation in Eq. 6.23.

6.1.2 Semi-numerical reduction

Once the general structure of the residues has been determined, we can numerically perform the integrand reduction to extract the values of all process-dependent coefficients which appear in their parametric form. The decomposition can be checked by verifying the identity between the original numerator and its reconstruction, i.e. between l.h.s. and r.h.s. of Eq. (6.2), for arbitrary values of the integration momenta q_i . This procedure is known as global $\mathcal{N} = \mathcal{N}$ test of the integrand reduction. Using the same universal parametrization of the residues, we will perform the reduction of the integrands in $\mathcal{N} = 4$ SYM and $\mathcal{N} = 8$ SUGRA.

Reduction in $\mathcal{N} = 4$ SYM

We will first consider the integrands of $\mathcal{N} = 4$ SYM, given in Eq. (6.5), (6.6) and (6.7) for the three diagrams in Fig. 6.1 respectively.

The 8-ple cut of the pentabox integrand of Eq. (6.5) can be parametrized using the monomials in Table 6.1,

$$\Delta_{12345678} = c_{12345678,0} + c_{12345678,1}(q_1 \cdot k_1) + c_{12345678,2}(q_2 \cdot k_3) + c_{12345678,3}(q_2 \cdot k_4). \quad (6.26)$$

The number of solutions equals the number of coefficients, in accordance with the maximum-cut theorem. Therefore the four coefficients appearing in Eq. (6.26) can be obtained by sampling the numerator on the four solutions of the 8-ple cut, where the decomposition becomes

$$\mathcal{N}_{1\dots 8}^{(4,a)} = \Delta_{12345678}. \quad (6.27)$$

In our case we find that only $c_{12345678,0}$ and $c_{12345678,1}$ are non-vanishing. The residue $\Delta_{i_1\dots i_7}$ of the generic 7-ple cut can be parametrized using the results listed in Table 6.1. For the process at hand, the simple structure of the numerator ensures that the residue can be parametrized just by a constant term,

$$\Delta_{i_1\dots i_7} = c_{i_1\dots i_7,0}. \quad (6.28)$$

The value of $c_{i_1\dots i_7,0}$ is obtained by sampling the numerator and the residue of the 8-ple cut in correspondence of one solution of the 7-ple cut, where

$$\Delta_{i_1\dots i_7} = \frac{\mathcal{N}_{1\dots 8}^{(4,a)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1, \dots, i_7} D_h}. \quad (6.29)$$

The multi-pole decomposition of the integrand $\mathcal{I}_{1\dots 8}^{(4,a)}$ thus becomes

$$\mathcal{I}_{1\dots 8}^{(4,a)}(q_1, q_2) = \frac{c_{12345678,0} + c_{12345678,1}(q_1 \cdot k_1)}{D_1 \cdots D_8} + \sum_{i=4}^7 \frac{c_{1\dots(i-1)(i+1)\dots 8,0}}{\prod_{h \neq i}^8 D_h}. \quad (6.30)$$

This result also shows the decomposition of the integral as linear combination of two Master Integrals with eight denominators and four Master Integrals with seven denominators.

The integrand $\mathcal{I}_{1\dots 8}^{(b)}$ of the crossed pentagon of Eq. (6.6) has the same numerator as the integrand $\mathcal{I}_{1\dots 8}^{(a)}$. One can proceed with the same method we used for the latter obtaining an identical decomposition for the numerator. The multi-pole decomposition will therefore have the same form as Eq. (6.30).

The third integrand, whose expression is given in Eq. (6.7), can be similarly decomposed in terms of 8- and 7-point residues. The parametrization of the residue of the 8-ple cut is given in Table 6.3 and can be written as

$$\begin{aligned} \Delta_{12345678} &= c_{12345678,0} + c_{12345678,1}(q_2 \cdot k_3) + c_{12345678,2}(q_1 \cdot k_5) \\ &\quad + c_{12345678,3}(q_1 \cdot k_1) + c_{12345678,4}(q_1 \cdot k_5)^2 + c_{12345678,5}(q_1 \cdot k_5)(q_1 \cdot k_1) \\ &\quad + c_{12345678,6}(q_2 \cdot k_3)(q_1 \cdot k_5) + c_{12345678,7}(q_1 \cdot k_5)^3. \end{aligned} \quad (6.31)$$

The 8-ple cut is a maximum cut. Its eight solutions allow one to determine the coefficients in Eq. (6.31) using the relation

$$\mathcal{N}_{1\dots 8}^{(4,b)} = \Delta_{12345678} \quad (6.32)$$

which holds at values of q_i such that $D_1 = \dots = D_8$. The non-vanishing coefficients are $c_{12345678,i}$ for $i \leq 4$. Since the numerator has rank one, one can easily see that the generic 7-ple residue $\Delta_{i_1\dots i_7}$ entering Eq. (6.16) can be parametrized by a constant term, i.e.

$$\Delta_{i_1\dots i_7} = c_{i_1\dots i_7,0}. \quad (6.33)$$

The value of $c_{i_1\dots i_7,0}$ is obtained by sampling the numerator and the residue of the 8-ple cut at a solution of the cut $D_{i_1} = \dots = D_{i_7}$, where the relation

$$\Delta_{i_1\dots i_7} = \frac{\mathcal{N}_{1\dots 8}^{(4,c)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1\dots i_7}^8 D_h} \quad (6.34)$$

holds. The multi-pole decomposition of the integrand of the double pentagon reads as follows

$$\begin{aligned} \mathcal{I}_{1\dots 8}^{(4,c)}(q_1, q_2) &= \frac{c_{12345678,0} + c_{12345678,1}(q_2 \cdot k_3) + c_{12345678,2}(q_1 \cdot k_5) + c_{12345678,3}(q_1 \cdot k_1)}{D_1 \cdots D_8} \\ &\quad + \sum_{i=1}^8 \frac{c_{1\dots(i-1)(i+1)\dots 8,0}}{\prod_{h \neq i}^8 D_h}. \end{aligned} \quad (6.35)$$

The corresponding decomposition of the integral is a linear combination of four Master Integrals with eight denominators and eight Master Integrals with seven denominators.

Reduction in $\mathcal{N} = 8$ SUGRA

We will now repeat the reduction for $\mathcal{N} = 8$ SUGRA. This requires more operations, because the rank is higher and also 6-ple cuts are needed, besides 8-ple and 7-ple cuts, as shown in Eq. (6.17).

We first consider the pentabox diagram in Fig. 6.1a. The computation of the residue of the 8-ple cut $\Delta_{12345678}$ follows the same pattern as the $\mathcal{N} = 4$ SYM planar pentabox, described above. The non-vanishing coefficients are $c_{12345678,0}$ and $c_{12345678,1}$. The residue of the generic 7-ple cut $D_{i_1} = \dots = D_{i_7}$ in Eq. (6.17) can be parametrized in terms of the monomials collected in Table 6.1. The structure of the numerator guarantees that the residue contains rank-one terms at most:

$$\begin{aligned} \Delta_{i_1 \dots i_7} &= c_{i_1 \dots i_7,0} + c_{i_1 \dots i_7,1}(q + w_0) \cdot w_1 + c_{i_1 \dots i_7,2}(q + w_0) \cdot w_2 \\ &\quad + c_{i_1 \dots i_7,3}(k + w_3) \cdot w_4 + c_{i_1 \dots i_7,4}(k + w_3) \cdot w_5. \end{aligned} \quad (6.36)$$

The momenta w_i^μ depend on the cut $(i_1 \dots i_7)$ and can be read from Table 6.1. The actual value of the coefficients can be obtained by sampling the numerator on five independent solutions of the 7-ple cut, where

$$\Delta_{i_1 \dots i_7} = \frac{\mathcal{N}_{1 \dots 8}^{(8,a)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1, \dots, i_7}^8 D_h}. \quad (6.37)$$

In this case, all the coefficients but $c_{i_1 \dots i_7,0}$, $c_{i_1 \dots i_7,1}$, and $c_{i_1 \dots i_7,2}$ vanish. The numerator $\mathcal{N}_{1 \dots 8}^{(8,a)}$ has rank two in q and is independent of k . The only non-vanishing term in the residue of the generic 6-pole cut $D_{i_1} = \dots = D_{i_6}$ in (6.17) is the constant. Hence, we have

$$\Delta_{i_1 \dots i_6} = c_{i_1 \dots i_6,0}. \quad (6.38)$$

The actual value of the constant can be obtained evaluating the decomposition (6.17) at one solution of the 6-ple cut, where

$$\Delta_{i_1 \dots i_6} = \frac{\mathcal{N}_{1 \dots 8}^{(8,a)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1, \dots, i_6}^8 D_h} - \sum_{\substack{h=4 \\ h \neq i_1, \dots, i_6}}^7 \frac{\Delta_{i_1 \dots h \dots i_6}}{D_h}. \quad (6.39)$$

After polynomial fitting of $\Delta_{12345678}$, $\Delta_{i_1 \dots i_7}$ and $\Delta_{i_1 \dots i_6}$, the resulting multi-pole decomposition of Eq. (6.18) contains 20 non-vanishing coefficients two of which are spurious (i.e. their contribution vanishes upon integration) while the others give rise to Master Integrals, namely two with eight denominators, ten with seven denominators and six with six denominators.

As in the $\mathcal{N} = 4$ SYM case, the crossed pentabox in Fig 6.1b has the same numerator and the same decomposition as the planar pentabox. Therefore the coefficients of the former are exactly the same as the coefficients of the latter.

Finally, we address the reduction of the double pentagon in Fig. 6.1c. The computation of the residue of the 8-ple cut of the double pentagon follows the same lines of the $\mathcal{N} = 4$ SYM double pentagon. The parametrization of the residue is given in Eq. (6.31). In this case the only vanishing coefficient is $c_{12345678,7}$. A 7-point residue $\Delta_{i_1 \dots i_7}$ of the generic cut $D_{i_1} = \dots = D_{i_7}$ can be parametrized using Eq. (6.36). At the 7-ple cut $D_{i_1} = \dots = D_{i_7}$ the

decomposition (6.17) reduces to

$$\Delta_{i_1 \dots i_7} = \frac{\mathcal{N}_{1 \dots 8}^{(8,c)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1, \dots, i_7}^8 D_h}. \quad (6.40)$$

The coefficients are then computed by sampling Eq. (6.40) at five solutions of the 7-ple cut. The non-vanishing ones are those multiplying constant or linear terms in the loop momenta. The residue $\Delta_{i_1 \dots i_6}$ of the generic 6-ple cut $D_{i_1} = \dots = D_{i_6}$ can be parametrized by a constant, as in Eq. (6.38). The constant is computed using one solution of the 6-ple cut and the expression of the decomposition (6.17) at the 6-ple cut,

$$\Delta_{i_1 \dots i_6} = \frac{\mathcal{N}_{1 \dots 8}^{(8,a)}(q_1, q_2) - \Delta_{12345678}}{\prod_{h \neq i_1, \dots, i_6}^8 D_h} - \sum_{h \neq i_1, \dots, i_6}^8 \frac{\Delta_{i_1 \dots h \dots i_6}}{D_h}. \quad (6.41)$$

We find that $\Delta_{123456} = 0$, while the residues of all the other 6-ple cuts are non-vanishing. After polynomial fitting of $\Delta_{12345678}$, $\Delta_{i_1 \dots i_7}$ and $\Delta_{i_1 \dots i_6}$, the resulting multi-pole decomposition of Eq. (6.18) contains in this case 74 non-vanishing coefficients, four of which are spurious. The integral can be decomposed as a linear combination of seven Master Integrals with eight denominators, 36 Master Integrals with seven denominators and 27 Master Integrals with six denominators.

6.1.3 Analytic computation

In this section we perform the reduction of the five-point diagrams analytically. In order to make the computation simpler, we apply a two-loop generalization of the integrand reduction via Laurent expansion we presented in Chapter 4. As in the one-loop case, the Laurent expansion allows one to find simpler formulas for the coefficients entering the decomposition. Moreover, the subtraction of higher-point residues can be performed at the coefficient level rather than at the integrand level. Indeed, in the asymptotic limit defined by the Laurent expansion, both the integrand and the subtraction terms are polynomial (while, in general, only their difference is polynomial). Therefore the subtraction can be implemented at the coefficient level, i.e. by correcting the terms of the Laurent expansion of the integrands with suitable counter-terms which are functions of higher-point residues, similarly to the one-loop case. For simplicity, we will focus on the rank-one numerators in the five-point integrands of $\mathcal{N} = 4$ SYM. The method can, however, be extended to numerators where the rank is higher, such as the ones of $\mathcal{N} = 8$ SUGRA, as we will briefly show at the end of the section.

Pentabox diagram

We will first show the analytic decomposition of the pentabox diagram of Fig. 6.1a. We will actually perform a more general computation, valid for any numerator of the type in Eq. (6.5), i.e. for any rank-one numerator depending on q_1 only. The results for $\mathcal{N} = 4$ SYM will be recovered at the very end, using Eq. (6.8) and (6.11).

The four solutions of the 8-ple cut $D_1 = \dots = D_8 = 0$ are

$$\begin{aligned} (q_{1,(1)}^\mu, q_{2,(1)}^\mu) &= \left(\frac{\langle 45 \rangle \langle 3|\gamma^\mu|4\rangle}{\langle 35 \rangle 2}, \frac{\langle 32 \rangle \langle 1|\gamma^\mu|2\rangle}{\langle 13 \rangle 2} \right) \\ (q_{1,(2)}^\mu, q_{2,(2)}^\mu) &= \left(\frac{\langle 45 \rangle \langle 3|\gamma^\mu|4\rangle}{\langle 35 \rangle 2}, \frac{\langle 15 \rangle \langle 2|\gamma^\mu|1\rangle}{\langle 25 \rangle 2} \right) \\ (q_{1,(3)}^\mu, q_{2,(3)}^\mu) &= \left(\frac{[45] \langle 4|\gamma^\mu|3\rangle}{[35] 2}, \frac{[15] \langle 1|\gamma^\mu|2\rangle}{[25] 2} \right) \\ (q_{1,(4)}^\mu, q_{2,(4)}^\mu) &= \left(\frac{[45] \langle 4|\gamma^\mu|3\rangle}{[35] 2}, \frac{[32] \langle 2|\gamma^\mu|1\rangle}{[13] 2} \right). \end{aligned}$$

The general parametrization of the residue $\Delta_{12345678}$ is given in Eq. (6.26). The simple form of the numerator $\mathcal{N}_{1\dots 8}^{(4,a)}$ implies that the coefficients $c_{12345678,2}$ and $c_{12345678,3}$ vanish. The non-vanishing coefficients can be obtained by sampling at the solutions $(q_{1,(1)}^\mu, q_{2,(1)}^\mu)$ and $(q_{1,(3)}^\mu, q_{2,(3)}^\mu)$ only. The outcome is

$$\begin{aligned} c_{12345678,0} &= -\frac{1}{\langle 54 \rangle \langle 31 \rangle [53] [41] - \langle 53 \rangle \langle 41 \rangle [54] [31]} \\ &\quad \times \left(\langle 54 \rangle \langle 41 \rangle \langle 3|v_1|4\rangle [54] [31] - \langle 54 \rangle \langle 31 \rangle \langle 4|v_1|3\rangle [54] [41] \right. \\ &\quad \left. - \beta_1 \langle 54 \rangle \langle 31 \rangle [53] [41] + \beta_1 \langle 53 \rangle \langle 41 \rangle [54] [31] \right) \end{aligned} \quad (6.42)$$

$$c_{12345678,1} = -2 \frac{\langle 54 \rangle \langle 3|v_1|4\rangle [53] - \langle 53 \rangle \langle 4|v_1|3\rangle [54]}{\langle 54 \rangle \langle 31 \rangle [53] [41] - \langle 53 \rangle \langle 41 \rangle [54] [31]}. \quad (6.43)$$

One can show that all the 7-ple cuts $D_{i_1} = \dots = D_{i_7} = 0$ of the diagram in Fig. 6.1a have solutions of the form

$$(q_1^\mu, q_2^\mu) = (a_1^\mu t + a_0^\mu, b_0^\mu), \quad (6.44)$$

where a_0^μ , a_1^μ , and b_0^μ are momenta which are independent from the loop coordinates, while t is a variable which is not fixed by the cut conditions. An easy way to find the only non-vanishing coefficient of $\Delta_{i_1\dots i_7}$ is evaluating Eq. (6.29) on these solutions and take the asymptotic limit $t \rightarrow \infty$. In general, neither of the two contributions on the r.h.s. of Eq. (6.29) is polynomial in t , but only their difference is. However, similarly to the one-loop case, when taking the limit $t \rightarrow \infty$ both contributions are constant,

$$\left. \frac{\mathcal{N}_{1\dots 8}^{(4,a)}}{D_{i_8}} \right|_{t \rightarrow \infty} = n_{i_1\dots i_7,0} + \mathcal{O}\left(\frac{1}{t}\right), \quad \left. \frac{\Delta_{12345678}}{D_{i_8}} \right|_{t \rightarrow \infty} = c_{i_1\dots i_7,0}^{(s,i_8)} + \mathcal{O}\left(\frac{1}{t}\right).$$

Therefore the coefficients $n_{i_1\dots i_7,0}$ and $c_{i_1\dots i_7,0}^{(s,i_8)}$ can be computed separately, obtaining the coefficient $c_{i_1\dots i_7,0}$ as their difference

$$c_{i_1\dots i_7,0} = n_{i_1\dots i_7,0} - c_{i_1\dots i_7,0}^{(s,i_8)} \quad (6.45)$$

Moreover, the known structure of $\Delta_{12345678}$ allows one to compute the coefficient $c_{i_1\dots i_7,0}^{(s,i_8)}$ once and for all as a parametric function of 8-ple cut coefficients. Its expression is given by

$$c_{i_1\dots i_7,0}^{(s,i_8)} = \frac{c_{12345678,1}(k_1 \cdot a_1)}{2(p_{i_8} + a_0) \cdot a_1}, \quad (6.46)$$

where p_{i_8} is the momentum appearing in the uncut denominator, defined in Eq. (3.4). The coefficients $n_{i_1 \dots i_7, 0}$ read as follows

$$\begin{aligned}
n_{12345678,0} &= -\frac{v_1 \cdot a_1}{k_5 \cdot a_1} && \text{with } a_1^\mu = \frac{\langle 3|\gamma^\mu|4\rangle}{2}, \\
n_{1234678,0} &= \frac{v_1 \cdot a_1}{k_3 \cdot a_1} && \text{with } a_1^\mu = \frac{\langle 5|\gamma^\mu|4\rangle}{2}, \\
n_{1234578,0} &= -\frac{v_1 \cdot a_1}{k_4 \cdot a_1} && \text{with } a_1^\mu = \frac{\langle 3|\gamma^\mu|K_{345}\rangle}{2}, \\
n_{1235678,0} &= -\frac{v_1 \cdot a_1}{k_4 \cdot a_1} && \text{with } a_1^\mu = \frac{\langle 5|\gamma^\mu|K_{543}\rangle}{2},
\end{aligned} \tag{6.47}$$

where the massless momentum K_{abc} is defined as

$$K_{ijl}^\mu \equiv k_j^\mu + k_l^\mu - \frac{s_{jl}}{2(k_j + k_l) \cdot k_i} k_i^\mu. \tag{6.48}$$

As already stated, the foregoing discussion applies to any numerator of the form given in Eq. (6.5). By using the explicit expressions of v_1 and β_1 given in Eq.s (6.8) and (6.11), we can write down the results for the coefficients in $\mathcal{N} = 4$ SYM in terms of the functions γ defined in Eq (6.15)

$$\begin{aligned}
c_{12345678,0} &= \frac{1}{2} (\gamma_{12345}(s_{23} - s_{13} - s_{45}) + s_{12}(\gamma_{34125} + \gamma_{35124} + \gamma_{45123})) \\
c_{12345678,1} &= -2\gamma_{12345} \\
c_{1234568,0} &= \frac{1}{4} (-\gamma_{35124} - \gamma_{45123} + 2\gamma_{12345}) \\
c_{1234578,0} &= \frac{1}{4} (-\gamma_{34125} + \gamma_{35124} + 2\gamma_{45123}) \\
c_{1234678,0} &= \frac{1}{4} (-\gamma_{35124} - \gamma_{34125} - 2\gamma_{12345}) \\
c_{1235678,0} &= \frac{1}{4} (-\gamma_{45123} + \gamma_{35124} + 2\gamma_{34125}).
\end{aligned} \tag{6.49}$$

The complete integrand decomposition is obtained plugging the coefficients of Eq. (6.49) in Eq. (6.30). These results are in agreement with the ones found in the numerical computation.

Crossed pentabox diagram

As already noticed in the discussion of the numerical computation, the numerators of the crossed pentabox of Fig. 6.1b and the planar pentabox of Fig. 6.1a have the same decomposition and the same coefficients in the residues.

Double pentagon diagram

The numerator $\mathcal{N}_{1\dots 8}^{(4,c)}$, as already noted, is invariant upon the transformation of Eq. (6.23). This can be exploited in order to simplify the computation. Indeed, we can rewrite the

numerator as

$$\mathcal{N}_{1\dots 8}^{(4,c)}(q_1, q_2) = \widehat{\mathcal{N}}_{1\dots 8}^{(4,c)}(q_1) - \left[k_1^\mu \leftrightarrow k_3^\mu, k_4^\mu \leftrightarrow k_5^\mu, q_1^\mu \leftrightarrow q_2^\mu \right], \quad (6.50)$$

where

$$\widehat{\mathcal{N}}_{1\dots 8}^{(4,c)}(q_1) \equiv 2q_1 \cdot v_2 + \beta_2. \quad (6.51)$$

Therefore, we can perform the reduction of (6.51) and then use Eq. (6.50) in order to get the final result. Since $\widehat{\mathcal{N}}_{1\dots 8}^{(4,c)}$ depends only on one of the two loop momenta, its decomposition is simpler, involving fewer residues and coefficients. More in detail, we have

$$\widehat{\mathcal{N}}_{1\dots 8}^{(4,c)} = \Delta_{12345678} + \sum_{i=1}^3 \Delta_{1\dots(i-1)(i+1)\dots 8} D_i. \quad (6.52)$$

Similarly to the previous case, we will first perform the reduction of a generic numerator of the form of Eq. (6.51) and then we will use the definitions in Eq. (6.9) and (6.12), as well as Eq. (6.50), in order to get the result for $\mathcal{N} = 4$ SYM.

As in the previous cases, we start from the maximum cut, which is an 8-ple cut. The solutions of the system $D_1 = \dots = D_8 = 0$ are eight and they can be used to compute the eight coefficients of the residue

$$\begin{aligned} \Delta_{12345678} &= \hat{c}_{12345678,0} + \hat{c}_{12345678,1}(q_2 \cdot k_3) + \hat{c}_{12345678,2}(q_1 \cdot k_5) \\ &\quad + \hat{c}_{12345678,3}(q_1 \cdot k_1) + \hat{c}_{12345678,4}(q_1 \cdot k_5)^2 + \hat{c}_{12345678,5}(q_1 \cdot k_5)(q \cdot k_1) \\ &\quad + \hat{c}_{12345678,6}(q_2 \cdot k_3)(q_1 \cdot k_5) + \hat{c}_{12345678,7}(q_1 \cdot k_5)^3 \end{aligned} \quad (6.53)$$

The rank of $\widehat{\mathcal{N}}_{1\dots 8}^{(4,c)}$ implies that $c_{12345678,i} = 0$ for $i \geq 4$. The simplicity of the numerator simplifies the computation even further. Indeed decomposing u_2^μ in the basis $\{k_i\}_{i=1,\dots,4}$, and using the conditions $D_1 = D_2 = D_3 = 0$ one gets

$$\hat{c}_{12345678,0} = \beta_2, \quad \hat{c}_{12345678,1} = 0. \quad (6.54)$$

The two remaining coefficients can be obtained by sampling the numerator on two solutions of the 8-ple cut, e.g.

$$\begin{aligned} (q_{1,(1)}^\mu, q_{2,(1)}^\mu) &= \left(\frac{\langle 35 \rangle \langle 4|\gamma^\mu|3 \rangle}{\langle 45 \rangle} \frac{\langle 41 \rangle \langle 5|\gamma^\mu|1 \rangle}{2}, \frac{\langle 41 \rangle \langle 5|\gamma^\mu|1 \rangle}{\langle 45 \rangle} \frac{\langle 35 \rangle \langle 4|\gamma^\mu|3 \rangle}{2} \right), \\ (q_{1,(2)}^\mu, q_{2,(2)}^\mu) &= \left(\frac{[35] \langle 3|\gamma^\mu|4 \rangle}{[45]} \frac{[41] \langle 1|\gamma^\mu|5 \rangle}{2}, \frac{[41] \langle 1|\gamma^\mu|5 \rangle}{[45]} \frac{[35] \langle 3|\gamma^\mu|4 \rangle}{2} \right), \end{aligned}$$

The value of the two remaining non-vanishing coefficients of the 8-ple-cut are thus found to be

$$\begin{aligned} \hat{c}_{12345678,2} &= -2 \frac{\langle 4|1|3 \rangle \langle 3|u_2|4 \rangle - \langle 3|1|4 \rangle \langle 4|u_2|3 \rangle}{\langle 4|5|3 \rangle \langle 3|1|4 \rangle - \langle 3|5|4 \rangle \langle 4|1|3 \rangle}, \\ \hat{c}_{12345678,3} &= 2 \frac{\langle 4|5|3 \rangle \langle 3|u_2|4 \rangle - \langle 3|5|4 \rangle \langle 4|u_2|3 \rangle}{\langle 4|5|3 \rangle \langle 3|1|4 \rangle - \langle 3|5|4 \rangle \langle 4|1|3 \rangle}. \end{aligned} \quad (6.55)$$

At the next step of the reduction, we consider the generic 7-ple cut residue $\Delta_{i_1 \dots i_7}$ appearing in the decomposition of the numerator, as in Eq. (6.52). The solutions of the cut depend on one free variable t . In particular, each cut has a set of solutions which have the following asymptotic behavior for $t \rightarrow \infty$

$$(q_1^\mu, q_2^\mu) = \left(a_1^\mu t + a_0^\mu + \mathcal{O}\left(\frac{1}{t}\right), b_1^\mu t + b_0^\mu + \mathcal{O}\left(\frac{1}{t}\right) \right). \quad (6.56)$$

We compute the coefficient $\hat{c}_{1345678,0}$ by evaluating the decomposition (6.52) on these parametric solutions in the asymptotic limit, where

$$\left[\frac{\widehat{N}_{1\dots 8}^{(4,c)}}{D_{i_8}} - \frac{\Delta_{12345678}}{D_{i_8}} \right]_{t \rightarrow \infty} = \hat{c}_{i_1 \dots i_7, 0}, \quad (6.57)$$

Similarly to the case of the planar diagram, the $t \rightarrow \infty$ limit makes both

$$\frac{\widehat{N}_{1\dots 8}^{(4,c)}}{D_{i_8}} \quad \text{and} \quad \frac{\Delta_{12345678}}{D_{i_8}}$$

polynomial in t , and in this special case constant, namely

$$\left. \frac{\widehat{N}_{1\dots 8}^{(4,c)}}{D_{i_8}} \right|_{t \rightarrow \infty} = \hat{n}_{i_1 \dots i_7, 0} + \mathcal{O}\left(\frac{1}{t}\right), \quad \left. \frac{\Delta_{12345678}}{D_{i_8}} \right|_{t \rightarrow \infty} = \hat{c}_{i_1 \dots i_7, 0}^{(s, i_8)} + \mathcal{O}\left(\frac{1}{t}\right). \quad (6.58)$$

The coefficients $\hat{n}_{i_1 \dots i_7, 0}$ and $\hat{c}_{i_1 \dots i_7, 0}^{(s, i_8)}$ can be computed separately, and

$$\hat{c}_{i_1 \dots i_7, 0} = \hat{n}_{i_1 \dots i_7, 0} - \hat{c}_{i_1 \dots i_7, 0}^{(s, i_8)}. \quad (6.59)$$

Therefore the subtraction can be performed at the coefficient level via the universal function

$$\hat{c}_{i_1 \dots i_7, 0}^{(s, i_8)} = \frac{\hat{c}_{12345678, 2}(a_1 \cdot k_5) + \hat{c}_{12345678, 3}(a_1 \cdot k_1)}{2(a_0 + p_{i_8}) \cdot a_1}. \quad (6.60)$$

The coefficients $\hat{n}_{i_1 \dots i_7, 0}$ are given by

$$\begin{aligned} \hat{n}_{1345678, 0} &= -\frac{v_2 \cdot a_1}{p_3 \cdot a_1}, & \text{with } v_{q_1}^\mu &= \eta_1 \frac{\langle 4|\gamma^\mu|3\rangle}{2} + \eta_2 p_4^\mu, \\ \hat{n}_{1245678, 0} &= \frac{v_2 \cdot a_1}{p_4 \cdot a_1}, & \text{with } v_{q_1}^\mu &= \eta_3 \frac{\langle 4|\gamma^\mu|3\rangle}{2} + \eta_4 p_3^\mu, \\ \hat{n}_{1234578, 0} &= \frac{v_2 \cdot a_1}{p_3 \cdot a_1}, & \text{with } v_{q_1}^\mu &= \eta_5 (p_3^\mu - p_4^\mu) + \eta_6 \frac{\langle 3|\gamma^\mu|4\rangle}{2} + \eta_7 \frac{\langle 4|\gamma^\mu|3\rangle}{2}, \end{aligned} \quad (6.61)$$

where

$$\begin{aligned}
\eta_1 &= -\frac{\langle 5\ 2\rangle[4\ 1]}{\langle 4\ 2\rangle[3\ 4]}, \\
\eta_2 &= \frac{\langle 5\ 2\rangle[3\ 1]}{\langle 4\ 2\rangle[3\ 4]}, \\
\eta_3 &= \frac{\langle 3\ 5\rangle[1\ 2]}{\langle 3\ 4\rangle[3\ 2]}, \\
\eta_4 &= -\frac{\langle 4\ 5\rangle[1\ 2]}{\langle 3\ 4\rangle[3\ 2]}, \\
\eta_5 &= \frac{-\sigma_1 + \sqrt{\sigma_1^2 - 4\sigma_2\sigma_3}}{2\sigma_2}, \\
\eta_6 &= \frac{2\sigma_2\sigma_8 - \sigma_1\sigma_4 + \sigma_4\sqrt{\sigma_1^2 - 4\sigma_2\sigma_3}}{2\sigma_2\sigma_5}, \\
\eta_7 &= -\frac{2\sigma_2\sigma_7 - \sigma_1\sigma_6 + \sigma_6\sqrt{\sigma_1^2 - 4\sigma_2\sigma_3}}{2\sigma_2\sigma_5},
\end{aligned} \tag{6.62}$$

in terms of

$$\begin{aligned}
\sigma_1 &= -4(\sigma_7\sigma_4 + \sigma_6\sigma_8)(p_1 \cdot p_2), \\
\sigma_2 &= -4(\sigma_6\sigma_4 - \sigma_5^2)(p_1 \cdot p_2), \\
\sigma_3 &= -4\sigma_7\sigma_8(p_1 \cdot p_2), \\
\sigma_4 &= 2(p_1 \cdot p_5)\langle 4\ 5\rangle[3\ 1] - 2(p_2 \cdot p_5)\langle 4\ 5\rangle[3\ 1] + \langle 3\ 5\rangle\langle 4\ 2\rangle[3\ 1][3\ 2] - \langle 4\ 5\rangle\langle 4\ 2\rangle[3\ 2][4\ 1], \\
\sigma_5 &= -\langle 3\ 5\rangle\langle 4\ 2\rangle[3\ 2][4\ 1] + \langle 3\ 2\rangle\langle 4\ 5\rangle[3\ 1][4\ 2], \\
\sigma_6 &= 2(p_2 \cdot p_3)\langle 3\ 5\rangle[4\ 1] - 2(p_2 \cdot p_5)\langle 3\ 5\rangle[4\ 1] + \langle 3\ 5\rangle\langle 3\ 2\rangle[3\ 1][4\ 2] - \langle 3\ 2\rangle\langle 4\ 5\rangle[4\ 1][4\ 2], \\
\sigma_7 &= \langle 3\ 5\rangle\langle 5\ 2\rangle[4\ 1][1\ 2], \\
\sigma_8 &= \langle 4\ 5\rangle\langle 5\ 2\rangle[3\ 1][1\ 2].
\end{aligned} \tag{6.63}$$

This completes the reduction of a generic numerator $\widehat{\mathcal{N}}_{1\dots 8}^{(4,c)}$ of the form given in Eq. (6.51). As previously stated, the reduction of the full numerator $\mathcal{N}_{1\dots 8}^{(4,c)}$ of $\mathcal{N} = 4$ SYM can be recovered from the one discussed in this section, by means of Eq.s (6.50) and (6.50). To that purpose, we observe that the substitutions in Eq. (6.23) give the one-to-one mapping between denominators

$$D_1 \leftrightarrow D_4, \quad D_2 \leftrightarrow D_6, \quad D_3 \leftrightarrow D_5, \quad D_7 \leftrightarrow D_8.$$

Putting everything together and using the definitions of v_2 and β_2 of Eq. (6.9) and (6.12) we

recover the full multi-pole decomposition of Eq. (6.35), with coefficients

$$\begin{aligned}
c_{12345678,0} &= \frac{1}{4} \left(\gamma_{34125} (2s_{13} - 2s_{35} + 2s_{23} + s_{15} + s_{34}) \right. \\
&\quad - \gamma_{15234} (2s_{13} - 2s_{35} + 2s_{12} + s_{15} + s_{34}) \\
&\quad \left. - (\gamma_{23145} + \gamma_{24135}) s_{23} - (\gamma_{25134} - \gamma_{12345}) (s_{35} - s_{14} - s_{12}) \right), \\
c_{12345678,1} &= -2 \gamma_{34125}, \\
c_{12345678,2} &= \frac{1}{2} (2\gamma_{34125} - \gamma_{23145} - \gamma_{24135} - 2\gamma_{15234} + \gamma_{25134} - \gamma_{12345}), \\
c_{12345678,3} &= \frac{1}{2} (2\gamma_{34125} - \gamma_{23145} - \gamma_{24135} + 2\gamma_{15234} + \gamma_{25134} - \gamma_{12345}), \\
c_{2345678,0} &= \frac{1}{4} (2\gamma_{34125} - \gamma_{23145} + \gamma_{24135}), \\
c_{1345678,0} &= \frac{1}{4} (-3\gamma_{34125} + 2\gamma_{23145} + \gamma_{24135} - \gamma_{25134} + \gamma_{12345}), \\
c_{1245678,0} &= \frac{1}{4} (\gamma_{34125} - \gamma_{23145} - 2\gamma_{24135} + \gamma_{25134} - \gamma_{12345}), \\
c_{1235678,0} &= \frac{1}{4} (-2\gamma_{15234} - \gamma_{25134} - \gamma_{12345}), \\
c_{1234678,0} &= \frac{1}{4} (-2\gamma_{34125} + \gamma_{15234} + \gamma_{25134}), \\
c_{1234578,0} &= \frac{1}{4} (2\gamma_{34125} + \gamma_{15234} + \gamma_{12345}), \\
c_{1234568,0} &= \frac{1}{4} (2\gamma_{34125} + \gamma_{25134} - \gamma_{12345}), \\
c_{1234567,0} &= \frac{1}{4} (-2\gamma_{34125} - \gamma_{25134} + \gamma_{12345}). \tag{6.64}
\end{aligned}$$

The coefficients of Eq. (6.64) enter the integrand decomposition of Eq. (6.35). These results are in agreement with the semi-numerical computation.

Higher-rank integrands

The analytic reduction can also be used for numerators of higher rank. As an example we perform the computation of the 7-ple cut residues for the $\mathcal{N} = 8$ SUGRA two-loop five-points planar pentabox, whose numerator has rank two in the loop momentum q_1 .

The most general parametrization of the generic 7-point residue $\Delta_{i_1 \dots i_7}$ is

$$\Delta_{i_1 \dots i_7} = c_{i_1 \dots i_7,0} + c_{i_1 \dots i_7,1} (q_1 + w_0) \cdot w_1 + c_{i_1 \dots i_7,2} (q_1 + w_0) \cdot w_2. \tag{6.65}$$

The momenta w_0^μ , w_1^μ , and w_2^μ are a linear combination of the external momenta and depend on the cut $(i_1 \dots i_7)$. Their actual form is not relevant for this discussion but can be read from Table 6.1. We consider two t -dependent solutions of the 7-ple cut:

$$(q_{1,(i)}^\mu, q_{2,(i)}^\mu) = \left(a_{(i),1}^\mu t + a_{(i)}^\mu, b_{(i)}^\mu \right) \quad \text{with } i = 1, 2. \tag{6.66}$$

The coefficients in Eq. (6.65) can be computed evaluating the numerator at the solutions in Eq. (6.66), where the decomposition becomes

$$\begin{aligned} \frac{\mathcal{N}_{1\dots 8}^{(8,a)} - \Delta_{12345678}}{D_{i_8}} &= \Delta_{i_1\dots i_7} \\ &= c_{i_1\dots i_7,0} + c_{i_1\dots i_7,1}(a_{(i),0} + w_0) \cdot w_1 \\ &\quad + c_{i_1\dots i_7,2}(a_{(i),0} + w_0) \cdot w_2 + c_{i_1\dots i_7,1}(a_{(i),1} \cdot w_1)t \\ &\quad + c_{i_1\dots i_7,2}(a_{(i),1} \cdot w_2)t, \end{aligned} \quad (6.67)$$

where D_{i_8} is the uncut denominator. The Laurent expansion around $t = \infty$ simplifies the computation. Indeed in this limit both

$$\frac{\mathcal{N}_{1\dots 8}^{(8,a)}(q_{1,(i)}, q_{2,(i)})}{D_{i_8}(q_{1,(i)}, q_{2,(i)})} \quad \text{and} \quad \frac{\Delta_{12345678}(q_{1,(i)}, q_{2,(i)})}{D_{i_8}(q_{1,(i)}, q_{2,(i)})}$$

have the same polynomial structure of the residue:

$$\begin{aligned} \left. \frac{\mathcal{N}_{1\dots 8}^{(8,a)}}{D_{i_8}} \right|_{t \rightarrow \infty} &= n_{i_1\dots i_7,0}^{(i)} + n_{i_1\dots i_7,1}^{(i)}t + \mathcal{O}\left(\frac{1}{t}\right) \\ \left. \frac{\Delta_{12345678}}{D_{i_8}} \right|_{t \rightarrow \infty} &= c_{i_1\dots i_7,0}^{(s,i_8,i)} + \mathcal{O}\left(\frac{1}{t}\right). \end{aligned} \quad (6.68)$$

The expression of the coefficients is obtained by plugging the expansions (6.68) in Eq. (6.67) and by comparing both sides. In particular $c_{i_1\dots i_7,1}$ and $c_{i_1\dots i_7,2}$ are the solution of the system

$$\begin{cases} c_{i_1\dots i_7,1}(a_{(1),1} \cdot w_1) + c_{i_1\dots i_7,2}(a_{(1),1} \cdot w_2) &= n_{i_1\dots i_7,1}^{(1)} \\ c_{i_1\dots i_7,1}(a_{(2),1} \cdot w_1) + c_{i_1\dots i_7,2}(a_{(2),1} \cdot w_2) &= n_{i_1\dots i_7,1}^{(2)} \end{cases}. \quad (6.69)$$

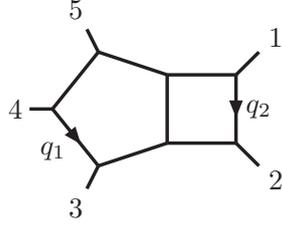
The coefficient $c_{i_1\dots i_7,0}$ is given by

$$\begin{aligned} c_{i_1\dots i_7,0} &= n_{i_1\dots i_7,0}^{(1)} - c_{i_1\dots i_7,0}^{(s,i_8,1)} - c_{i_1\dots i_7,1}(a_{(1),0} + w_0) \cdot w_1 - c_{i_1\dots i_7,2}(a_{(1),0} + w_0) \cdot w_2 \\ &= n_{i_1\dots i_7,0}^{(2)} - c_{i_1\dots i_7,0}^{(s,i_8,2)} - c_{i_1\dots i_7,1}(a_{(2),0} + w_0) \cdot w_1 - c_{i_1\dots i_7,2}(a_{(2),0} + w_0) \cdot w_2, \end{aligned} \quad (6.70)$$

in terms of the functions

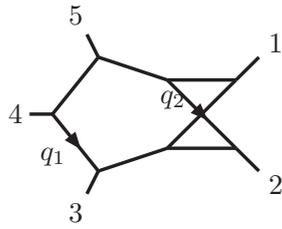
$$c_{i_1\dots i_7,0}^{(s,i_8,i)} = \frac{c_{12345678,1}(p_1 \cdot a_{(i),1})}{2(a_{(i),0} + p_{i_8}) \cdot a_{(i),1}}. \quad (6.71)$$

Eq. (6.70) shows that the coefficient $c_{i_1\dots i_7,0}$ can be written as the constant term $n_{i_1\dots i_7,0}^{(i)}$ of the Laurent expansion of the integrand, corrected by two kinds of contributions. The first, $c_{i_1\dots i_7,0}^{(s,i_8,i)}$, implements the 8-ple-cut subtraction as a correction at the coefficient level. The other terms are proportional to the higher-rank coefficients of the same cut found as solutions of the system in Eq. (6.69).



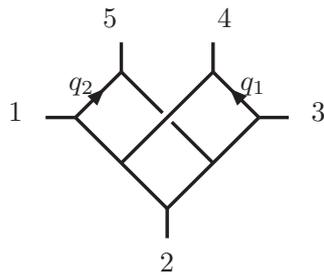
$$\begin{aligned}
 D_1 &= q_2^2 \\
 D_2 &= (q_2 + k_2)^2 \\
 D_3 &= (q_2 - k_1)^2 \\
 D_4 &= q_1^2 \\
 D_5 &= (q_1 + k_3)^2 \\
 D_6 &= (q_1 - k_4)^2 \\
 D_7 &= (q_1 - k_4 - k_5)^2 \\
 D_8 &= (q_1 + q_2 + k_2 + k_3)^2
 \end{aligned}$$

(a) Pentabox diagram



$$\begin{aligned}
 D_1 &= q_2^2 \\
 D_2 &= (q_2 + k_2)^2 \\
 D_3 &= (q_2 + q_1 - k_4 - k_5)^2 \\
 D_4 &= q_1^2 \\
 D_5 &= (q_1 + k_3)^2 \\
 D_6 &= (q_1 - k_4)^2 \\
 D_7 &= (q_1 - k_4 - k_5)^2 \\
 D_8 &= (q_1 + q_2 + k_2 + k_3)^2
 \end{aligned}$$

(b) Crossed pentabox diagram



$$\begin{aligned}
 D_1 &= q_1^2 \\
 D_2 &= (q_1 - k_3)^2 \\
 D_3 &= (q_1 + k_4)^2 \\
 D_4 &= q_2^2 \\
 D_5 &= (q_2 + k_5)^2 \\
 D_6 &= (q_2 - k_1)^2 \\
 D_7 &= (q_2 - q_1 + k_3 + k_5)^2 \\
 D_8 &= (q_2 - q_1 - k_1 - k_4)^2
 \end{aligned}$$

(c) Double pentagon diagram

Figure 6.1: Five-point diagrams entering the amplitudes in $\mathcal{N} = 4$ SYM and $\mathcal{N} = 8$ SUGRA. For each diagram, the definition of the denominators is shown as well.

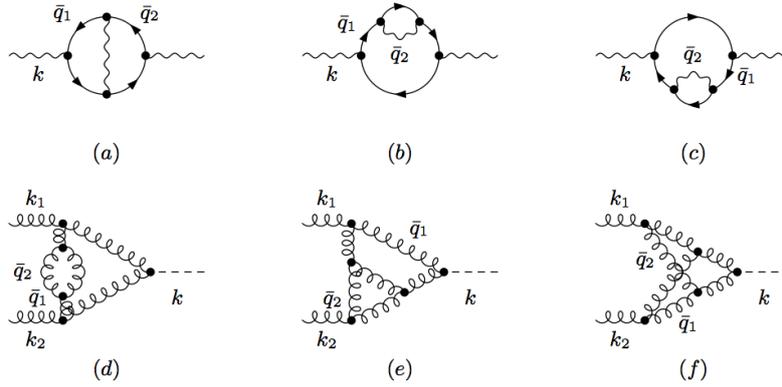


Figure 6.2: First row: diagrams leading to the two-loop QED corrections to the photon self energy. Second row: two-loop diagrams entering the QCD corrections to $gg \rightarrow H$ in the heavy top mass approximation.

6.2 Algebraic multi-loop reduction via polynomial division

In Section 3.3.2 we explained that the whole reduction of a loop integrand can be carried out using purely algebraic operations, by means of the recursive formula of Eq. (3.9). This allows to address the problem of the reduction of any loop amplitude without the need of computing the solutions of the multiple cuts or the parametric form of the residues. In particular, it can also be applied to cases where the presence of higher powers of loop denominators prevents one to use other methods such as the fit-on-the-cut approach.

In the examples which follow, the computation has been performed using the implementation of the iterative algorithm described in Section 3.4. The diagrams we used in these examples are depicted in Fig. 6.2. Despite their simplicity, these show the broadness of applicability of the method which is not affected by the presence of massive propagators, non planar diagrams, higher powers of loop denominators or higher-rank contributions in the numerator.

6.2.1 Photon vacuum polarization

As a first example we consider the two-loop contributions to the transverse part $\Pi(k^2)$ of the vacuum polarization in QED with a massive fermion [189]. The integrand of $\Pi(k^2)$ gets contributions from the three self-energy diagrams in the first row of Figure 6.2. As we showed in Eq. (3.2), the d -dimensional loop momenta \bar{q}_i are split into a 4-dimensional and (-2ϵ) -dimensional part, $\bar{q}_i = q_i + \vec{\mu}_i$, with $q_i \cdot \vec{\mu}_j = 0$ and $\vec{\mu}_i \cdot \vec{\mu}_j \equiv \mu_{ij}$. In this case the variables \mathbf{z} are $\mu_{11}, \mu_{22}, \mu_{12}$ and the components of q_i in the basis $\{k, k_\perp, e_3, e_4\}$, such that

$$k \cdot k_\perp = k \cdot e_j = k_\perp \cdot e_j = e_j^2 = 0.$$

The integrand of the diagram (a) is

$$\mathcal{I}_{12345}^{(a)} = \frac{1}{3 - 2\epsilon} \frac{\mathcal{N}_{12345}^{(a)}}{D_1 D_2 D_3 D_4 D_5}, \quad (6.72)$$

while its denominators are

$$\begin{aligned} D_1 &= \bar{q}_1^2 - m^2 \\ D_2 &= (\bar{q}_1 + k)^2 - m^2 \\ D_3 &= \bar{q}_2^2 - m^2 \\ D_4 &= (\bar{q}_2 + k)^2 - m^2 \\ D_5 &= (\bar{q}_1 - \bar{q}_2)^2. \end{aligned} \quad (6.73)$$

The first step of the reduction requires the division $\mathcal{N}_{12345}^{(a)}/\mathcal{G}_{12345}$, i.e. the multivariate polynomial division of the numerator with respect to the Gröbner basis defined by the ideal \mathcal{J}_{12345} . The result reads

$$\mathcal{N}_{12345}^{(a)} = \Delta_{12345} + \mathcal{N}_{1235} D_4 + \mathcal{N}_{2345} D_1 + \mathcal{N}_{1345} D_2 + \mathcal{N}_{1245} D_3 + \mathcal{N}_{1234} D_5, \quad (6.74)$$

where Δ_{12345} is the remainder of the division, while the quotient, which belongs to the ideal \mathcal{J}_{12345} , is written as a combination of loop denominators (see definition in Eq. (3.5)). In the second step, the (sub-)numerators $\mathcal{N}_{i_1 i_2 i_3 i_4}$ are reduced performing the division $\mathcal{N}_{i_1 i_2 i_3 i_4}/\mathcal{G}_{i_1 i_2 i_3 i_4}$,

$$\begin{aligned} \mathcal{N}_{12345}^{(a)} &= \Delta_{12345} + \Delta_{1235} D_4 + \Delta_{2345} D_1 + \Delta_{1345} D_2 + \Delta_{1245} D_3 + \Delta_{1234} D_5 \\ &\quad + \mathcal{N}_{123} D_4 D_5 + \mathcal{N}_{124} D_3 D_5 + \mathcal{N}_{134} D_2 D_5 + \mathcal{N}_{234} D_1 D_5 + \mathcal{N}_{125} D_3 D_4 \\ &\quad + \mathcal{N}_{135} D_2 D_4 + \mathcal{N}_{245} D_1 D_3 + \mathcal{N}_{345} D_1 D_2 + \mathcal{N}_{145} D_2 D_3 + \mathcal{N}_{235} D_1 D_4. \end{aligned} \quad (6.75)$$

The complete decomposition of $\mathcal{N}_{12345}^{(a)}$ is obtained by iterating the procedure twice,

$$\begin{aligned} \mathcal{N}_{12345}^{(a)} &= \Delta_{12345} + \Delta_{1235} D_4 + \Delta_{2345} D_1 + \Delta_{1345} D_2 + \Delta_{1245} D_3 + \Delta_{1234} D_5 \\ &\quad + \Delta_{123} D_4 D_5 + \Delta_{124} D_3 D_5 + \Delta_{134} D_2 D_5 + \Delta_{234} D_1 D_5 + \Delta_{125} D_3 D_4 \\ &\quad + \Delta_{135} D_2 D_4 + \Delta_{245} D_1 D_3 + \Delta_{345} D_1 D_2 + \Delta_{145} D_2 D_3 + \Delta_{235} D_1 D_4 \\ &\quad + \Delta_{13} D_2 D_4 D_5 + \Delta_{24} D_1 D_3 D_5 + \Delta_{14} D_2 D_3 D_5 + \Delta_{23} D_1 D_4 D_5. \end{aligned} \quad (6.76)$$

The residues in Eq. (6.76) read as follows:

$$\begin{aligned}
\Delta_{12345} &= 8(4m^4 - k^4 + k^2(k^2 - 2m^2)\epsilon) \\
\Delta_{1234} &= -4 \left[(4m^2 + k^2(3 - \epsilon - 2\epsilon^2)) \right. \\
&\quad \left. + 4(1 - \epsilon) \left(\mu_{12} - \frac{(q_1 \cdot k_\perp)(q_2 \cdot k_\perp)}{k_\perp^2} \right) \right. \\
&\quad \left. - \frac{(q_1 \cdot e_3)(q_2 \cdot e_4)}{(e_3 \cdot e_4)} - \frac{(q_1 \cdot e_4)(q_2 \cdot e_3)}{(e_3 \cdot e_4)} \right] \\
\Delta_{1235} &= \Delta_{2345} = \Delta_{1345} = \Delta_{1245} = 8(m^2 + k^2(1 - \epsilon)) \\
\Delta_{123} &= \Delta_{124} = \Delta_{134} = \Delta_{234} = 4(1 - \epsilon) \\
\Delta_{125} &= \Delta_{135} = \Delta_{245} = \Delta_{345} = -8(1 - \epsilon) \\
\Delta_{145} &= \Delta_{235} = 8\epsilon(1 - \epsilon) \\
\Delta_{13} &= \Delta_{24} = -\Delta_{14} = -\Delta_{23} = \frac{4(1 - \epsilon)}{k^2}.
\end{aligned} \tag{6.77}$$

The diagram (b) contains a double propagator,

$$\mathcal{I}_{11234}^{(b)} = \frac{1}{3 - 2\epsilon} \frac{\mathcal{N}_{11234}^{(b)}}{D_1^2 D_2 D_3 D_4}, \tag{6.78}$$

where the denominators are

$$\begin{aligned}
D_1 &= \bar{q}_1^2 - m^2 \\
D_2 &= (\bar{q}_1 - k)^2 - m^2 \\
D_3 &= \bar{q}_2^2 \\
D_4 &= (\bar{q}_1 + \bar{q}_2)^2 - m^2.
\end{aligned}$$

The first step of the reduction requires the division $\mathcal{N}_{11234}^{(b)}/\mathcal{G}_{11234}$ (note that $\mathcal{G}_{112345} \equiv \mathcal{G}_{1234}$) which yields

$$\mathcal{N}_{11234}^{(b)} = \Delta_{11234} + \mathcal{N}_{1234} D_1 + \mathcal{N}_{1123} D_4 + \mathcal{N}_{1134} D_2 + \mathcal{N}_{1124} D_3. \tag{6.79}$$

In the second step we perform the divisions $\mathcal{N}_{i_1 i_2 i_3 i_4}/\mathcal{G}_{i_1 i_2 i_3 i_4}$, obtaining

$$\begin{aligned}
\mathcal{N}_{11234} &= \mathcal{N}_{234} D_1^2 + \mathcal{N}_{134} D_1 D_2 + \mathcal{N}_{124} D_1 D_3 + \mathcal{N}_{123} D_1 D_4 + \mathcal{N}_{114} D_2 D_3 + \mathcal{N}_{113} D_2 D_4 \\
&\quad + \Delta_{1234} D_1 + \Delta_{1134} D_2 + \Delta_{1124} D_3 + \Delta_{1123} D_4 + \\
&\quad + \Delta_{11234}
\end{aligned} \tag{6.80}$$

The reduction is completed by performing the divisions $\mathcal{N}_{i_1 i_2 i_3}/\mathcal{G}_{i_1 i_2 i_3}$, along the lines of the previous steps, obtaining

$$\begin{aligned}
\mathcal{N}_{11234}^{(b)} &= \Delta_{11234} + \Delta_{1234} D_1 + \Delta_{1123} D_4 + \Delta_{1134} D_2 + \Delta_{1124} D_3 \\
&\quad + \Delta_{113} D_2 D_4 + \Delta_{114} D_2 D_3 + \Delta_{234} D_1^2
\end{aligned} \tag{6.81}$$

in terms of the residues

$$\begin{aligned}
\Delta_{11234} &= 16m^2 (k^2 + 2m^2 - k^2\epsilon) \\
\Delta_{1234} &= 16 [(q_2 \cdot k)(1 - \epsilon)^2 + m^2] \\
\Delta_{1124} &= -\Delta_{1123} = 8(1 - \epsilon) [k^2(1 - \epsilon) + 2m^2] \\
\Delta_{1134} &= -16m^2 (1 - \epsilon) \\
\Delta_{113} &= -\Delta_{114} = \Delta_{234} = 8(1 - \epsilon)^2.
\end{aligned} \tag{6.82}$$

The integrand of the diagram (c) and its decomposition are obtained by performing the replacement $k \rightarrow -k$ in the ones of the diagram (b).

We observe that the residues can also be expressed in terms of normal forms (see Appendix B.2, in particular Eq. (B.9)). For instance, in the case of $\mathcal{I}^{(b)}$, Δ_{1123} and Δ_{113} can be written as

$$\begin{aligned}
\Delta_{1123} &= \frac{\left[\mathcal{N}_{11234}^{(b)} - \Delta_{11234} \right]_{\mathcal{J}_{123}}}{[D_4]_{\mathcal{J}_{123}}} \\
\Delta_{113} &= \frac{\left[\mathcal{N}_{11234}^{(b)} - \Delta_{11234} - \Delta_{1123}D_4 - \Delta_{1134}D_2 \right]_{\mathcal{J}_{13}}}{[D_2D_4]_{\mathcal{J}_{13}}}
\end{aligned}$$

In other words, evaluating the integrand on a multiple cut $D_{j_1} = \dots = D_{j_k} = 0$ is equivalent to the algebraic operation of performing a reduction modulo the ideal $\mathcal{J}_{j_1 \dots j_k}$.

6.2.2 Diagrams for Higgs production via gluon fusion

We also consider the three-point diagrams in the second row of Figure 6.2, which enter the two-loop QCD corrections to Higgs boson production via gluon fusion in the heavy top limit [190]. In this case, the variables \mathbf{z} are μ_{11} , μ_{22} , μ_{12} and the components of the four-vectors q_i in the basis of massless vectors $\{k_1, k_2, e_3, e_4\}$, such that $k_i \cdot e_j = 0$ and $e_3 \cdot e_4 \neq 0$. Within the *divide-and-conquer* approach, the integrand of the generic diagram is decomposed as

$$\mathcal{I}^{(x)} = \sum_{k=2}^6 \sum_{\{i_1 \dots i_k\}} \frac{\Delta_{i_1 \dots i_k}}{D_{i_1} \dots D_{i_k}} \quad x = d, e, f. \tag{6.83}$$

For the diagram (d) the second sum runs over the multisubsets of $\{1, 1, 2, 3, 4, 5\}$, while for the diagram (e) and (f) it runs over the subsets of $\{1, \dots, 6\}$.

The expressions for the residues are lengthy and they are omitted, but they are available upon request.

Chapter 7

Integrand Reduction and independent Master Integrals

The general integrand reduction method presented in Chapter 3, and applied in this thesis to several examples both at one-loop (in Chapter 5) and higher-loops (in Chapter 6), rewrites scattering amplitudes as linear combination of Master Integrals, which we may call the *Master Integrals of the integrand reduction*, or more briefly *Master Integrands*. These are indeed irreducible contributions which are independent at the integrand level, but further identities might be valid between the corresponding integrals. In this chapter we propose a new way of finding these identities which consists in the combination of integrand reduction techniques with relations between integrals in different dimensions.

The traditional method for finding these identities is *Integration by Parts (IBP)* [53, 54], which consists in working out differentiation in expressions of the form

$$\int d^d \bar{q} \frac{\partial}{\partial \bar{q}^\mu} \mathcal{I}^\mu = 0, \quad (7.1)$$

where \mathcal{I}^μ could be any loop integrand with a free Lorentz index. This integral vanishes in dimensional regularization because of the boundary conditions. After computing the derivatives explicitly, it becomes a linear combination of integrals and thus the equation yields an identity (also known as *IBP identity*) between different integrals. This method can be applied using systematic ways for choosing the integrands \mathcal{I}^μ , among which the most well known is the Laporta algorithm [55]. IBP algorithms have been implemented in several public codes [191–194] and the method has been so far the most successful for the reduction of higher-loop amplitudes with non-trivial complexity to a minimal set of independent Master Integrals. The application of IBP becomes however increasingly complicated when dealing with integrals with many external legs. A possible solution would be using integrand reduction methods, which would reduce the multiplicity or the rank of higher-point integrands, followed IBP, whose application to lower-point or lower-rank integrals would be easier. This approach was followed e.g. in Ref.s [92, 195, 196].

In this Chapter we propose an alternative approach, where integrand reduction plays an important role also in finding these identities which are valid at the integral (but not necessarily at the integrand) level. With the *divide-and-conquer* technique (see Chapter 3, in particular Section 3.3.2) for the reduction, this approach also allows to re-use (some of) the *Gröbner basis* and *polynomial divisions* computed for the reduction in order to obtain additional identities between the integrals. The idea behind the method is combining integrand reduction with recurrence relations on the dimension of the integrals. As we shall see shortly, given an ℓ loop integral \mathcal{I} with external legs k_1, \dots, k_n , one can derive identities of the form

$$\mathcal{I}[S(2\epsilon; \vec{\mu}_1, \dots, \vec{\mu}_\ell)] \propto \mathcal{I}^{(d+2)} \quad (7.2)$$

$$\mathcal{I}[S(4; q_1, \dots, q_\ell, k_1, \dots, k_{n-1})] \propto \mathcal{I}^{(d+2)}, \quad (7.3)$$

$$\mathcal{I}[S(d; \bar{q}_1, \dots, \bar{q}_\ell, k_1, \dots, k_{n-1})] \propto \mathcal{I}^{(d+2)}, \quad (7.4)$$

where $S(D; v_1, \dots)$ is a function of a number of dimensions D and the momenta v_i , called *Shouten polynomial* [197], which we will formally define in Eq. 7.5. The notation $\mathcal{I}^{(D)}[\mathcal{N}]$ stands for the same loop integral as $\mathcal{I}[\mathcal{N}]$ (with $\mathcal{I}^{(D)} \equiv \mathcal{I}^{(D)}[1]$) but with the loop momentum taken to be D -dimensional. Note that the dimensional shift only involves the components of the loop momenta, while external legs are still four-dimensional. We will omit the superscript of the integral when it is evaluated in the usual number $d = 4 - 2\epsilon$ of dimensions (i.e. $\mathcal{I}[\mathcal{N}] \equiv \mathcal{I}^{(d)}[\mathcal{N}]$). The relation in Eq. (7.2) is an identity between integrals with the extra-dimensional variables μ_{ij} in the numerator, and the corresponding scalar integral in a number of dimensions shifted by two. Eq. (7.3) is instead a relation between an integrand with a specially chosen four-dimensional numerator and the same integral in $d + 2$, while Eq. (7.4) is a similar relation with a d -dimensional numerator at the l.h.s. In all cases the coefficient of proportionality between l.h.s. and r.h.s. is very easy to find. As we shall see, by performing an integrand reduction of the l.h.s. of Eq. (7.3) or Eq. (7.4) and combining the result with Eq. (7.2) and other recurrence relations, one can obtain identities where only integrals in a given number D of dimensions appear, with $D = d + 2d_s$ and d_s integer. With the shift $d \rightarrow d - d_s$ one thus obtains identities between integrals in $d = 4 - 2\epsilon$ dimensions. A systematic application of this method (similarly to IBP algorithms) requires a *bottom-up* approach, where relations for lower point integrals are computed first and then substituted into the ones of higher point integrals.

The analytic results presented in this chapter have been obtained with the help of the implementation of the divide-and-conquer algorithm described in Section 3.4.

7.1 Shouten polynomials, orthogonal vectors and orthogonal tensors

Here we give the formal definition of a Shouten polynomial. For a set of n momenta k_1, \dots, k_n , we define¹

$$S(D; k_1, \dots, k_n) \equiv \left(\epsilon^{(D)}(k_1, \dots, k_n) \right)^2 \quad (7.5)$$

where $\epsilon_{\mu_1 \dots \mu_n}^{(D)}$ is an anti-symmetric Levi-Civita tensor whose indexes formally live in D dimensions. The r.h.s. of the formula can be written as a combination of D -dimensional scalar products $(k_i \cdot k_j)$ by means of the identity

$$\epsilon_{\mu_1 \dots \mu_n}^{(D)} \epsilon_{\nu_1 \dots \nu_n}^{(D)} = \det \begin{pmatrix} g_{\mu_1 \nu_1}^{(D)} & g_{\mu_1 \nu_2}^{(D)} & \cdots & g_{\mu_1 \nu_n}^{(D)} \\ g_{\mu_2 \nu_1}^{(D)} & g_{\mu_2 \nu_2}^{(D)} & \cdots & g_{\mu_2 \nu_n}^{(D)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{\mu_n \nu_1}^{(D)} & g_{\mu_n \nu_2}^{(D)} & \cdots & g_{\mu_n \nu_n}^{(D)} \end{pmatrix}, \quad (7.6)$$

where $g_{\mu_1 \nu_1}^{(D)}$ is the D -dimensional metric tensor. These are also known as *Gram determinants* and the Computer Algebra System FORM [124] can compute them very efficiently.

The Shouten polynomial on the l.h.s. of Eq. (7.2) is thus a polynomial in the variables μ_{ij} defined in Eq. (3.2). Also notice that $S(-2\epsilon; \bar{q}_1, \dots, \bar{q}_\ell) = S(-2\epsilon; \vec{\mu}_1, \dots, \vec{\mu}_\ell)$. As for the Shouten polynomials on the l.h.s. of Eq. (7.3), we can give a very useful interpretation of them in terms of *orthogonal vectors* or *orthogonal tensors*. This interpretation would in turn allow us to find, at any loop order, the correct factors in the equation. In the one-loop case the four-dimensional Shouten polynomial in Eq. (7.3) reads

$$S(4; q, k_1, \dots, k_{n-1}) = \left(\epsilon^{(4)}(q, k_1, \dots, k_{n-1}) \right)^2. \quad (7.7)$$

We can define a vector v_\perp^μ

$$v_{\perp\mu} = \epsilon_{\mu\nu_1 \dots \nu_{n-1}}^{(4)} k_1^{\nu_1} \cdots k_{n-1}^{\nu_{n-1}} \quad (7.8)$$

which, because of the anti-symmetric properties of the Levi-Civita tensor, is orthogonal to all the external legs k_i of the loop diagram. The Shouten polynomial can thus be written as

$$S(4; q, k_1, \dots, k_{n-1}) = (q \cdot v_\perp)^2. \quad (7.9)$$

We can proceed similarly at ℓ loops, by defining an anti-symmetric orthogonal tensor $t_{\perp}^{\mu_1 \dots \mu_\ell}$ as

$$t_{\perp\mu_1 \dots \mu_\ell} = \epsilon_{\mu_1 \dots \mu_\ell \nu_1 \dots \nu_{n-1}}^{(4)} k_1^{\nu_1} \cdots k_{n-1}^{\nu_{n-1}}. \quad (7.10)$$

¹We use here the so-called SCHOONSHIP notation. For any tensor $T^{\mu_1 \dots \mu_n}$ and vectors k_1^μ, \dots, k_n^μ we define

$$T(k_1, \dots, k_n) \equiv T_{\mu_1 \dots \mu_n} k_1^{\mu_1} \cdots k_n^{\mu_n}.$$

This tensor is completely anti-symmetric and vanishes if any of its indexes is contracted with an external leg of the diagram. The ℓ -loop four-dimensional Shouten polynomial can thus be written as

$$S(4; q_1, \dots, q_\ell, k_1, \dots, k_{n-1}) = \left(t_\perp(q_1, \dots, q_\ell) \right)^2. \quad (7.11)$$

Similarly, we can also define d -dimensional orthogonal vectors and tensors

$$v_{(d)\perp\mu} = \epsilon_{\mu\nu_1 \dots \nu_{n-1}}^{(d)} k_1^{\nu_1} \dots k_{n-1}^{\nu_{n-1}} \quad (7.12)$$

$$t_{(d)\perp\mu_1 \dots \mu_\ell} = \epsilon_{\mu_1 \dots \mu_\ell \nu_1 \dots \nu_{n-1}}^{(d)} k_1^{\nu_1} \dots k_{n-1}^{\nu_{n-1}}. \quad (7.13)$$

which are related to d -dimensional Shouten polynomials by

$$S(d; \bar{q}, k_1, \dots, k_{n-1}) = (\bar{q} \cdot v_{(d)\perp})^2 \quad (7.14)$$

$$S(d; \bar{q}_1, \dots, \bar{q}_\ell, k_1, \dots, k_{n-1}) = \left(t_{(d)\perp}(\bar{q}_1, \dots, \bar{q}_\ell) \right)^2. \quad (7.15)$$

7.2 Dimensionally shifted integrals via Schwinger parametrization

In this section we show how to find the recurrence relations of the form of Eq. (7.2) at every loop order, following the method proposed in Ref. [198] for the two-loop case. The corresponding identities of Eq. (7.3) and (7.4) can then be easily derived from the first, as we will explicitly show for the one- and two-loop cases in the next sections.

We use the following definition for loop integrals

$$\mathcal{I}[\mathcal{N}] = \int \left(\prod_{i=1}^l \frac{d^d \bar{q}_i}{\pi^{d/2}} \right) \frac{\mathcal{N}}{D_1^{\alpha_1} \dots D_n^{\alpha_n}}, \quad \mathcal{I} \equiv \mathcal{I}[1]. \quad (7.16)$$

All the identities which are homogeneous in the number of dimensions are obviously independent of the choice of the normalization factor $1/\pi^{d/2}$. Also note that here we assume to have n distinct denominators raised to some power $\alpha_i \geq 1$. Using the identity

$$\left(-\frac{1}{D_i} \right)^{\alpha_i} = \frac{1}{(\alpha_i - 1)!} \int_0^\infty dx x^{\alpha_i - 1} \exp(x D_i), \quad (7.17)$$

one gets the well known representation of loop integrals in terms of *Schwinger parameters* x_i ,

$$\begin{aligned} \mathcal{I} &= (-)^\alpha \int \left(\prod_{i=1}^l \frac{d^d \bar{q}_i}{\pi^{d/2}} \right) \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i - 1}}{(\alpha_i - 1)!} \right) \exp \left(\sum_{i=1}^n x_i D_i \right) \\ &= (-)^\alpha \int \left(\prod_{i=1}^l \frac{d^d \bar{q}_i}{\pi^{d/2}} \right) \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i - 1}}{(\alpha_i - 1)!} \right) \exp \left(\sum_{i,j=1}^l A_{ij}(\bar{q}_i \cdot \bar{q}_j) + 2 \sum_i (q_i \cdot B_i) + C_0 \right), \end{aligned} \quad (7.18)$$

where $\alpha \equiv \sum_i \alpha_i$. The last equality defines the symmetric $\ell \times \ell$ matrix $A_{ij} = A_{ij}(x_1, \dots, x_n)$, the 4-dimensional vectors $B_i = B_i(x_1, \dots, x_n)$ for $i = 1, \dots, n$, and the constant $C_0 = C_0(x_1, \dots, x_n)$. This parametrization of the exponent holds because the loop denominators are *quadratic polynomials* in the components of the loop momenta.

We perform the (4-dimensional) change of variables

$$q'_i = q_i + \sum_j A_{ij}^{-1} B_j$$

and Eq. (7.18) becomes, after relabeling $q'_i \rightarrow q_i$,

$$\mathcal{I} = (-)^{\alpha} \int \left(\prod_{i=1}^l \frac{d^d \bar{q}_i}{\pi^{d/2}} \right) \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i-1}}{(\alpha_i-1)!} \right) \exp \left(\sum_{i,j=1}^l A_{ij}(\bar{q}_i \cdot \bar{q}_j) + C \right), \quad (7.19)$$

with $C = C_0 - \sum_{ij} A_{ij}^{-1} (B_i \cdot B_j)$. Because A_{ij} is real and symmetric, it can be diagonalized by a unitary change of variables

$$\bar{Q}_i = \sum_{j=1}^l U_{ij} \bar{q}_j$$

and the integral becomes

$$\begin{aligned} \mathcal{I} &= (-)^{\alpha} \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i-1}}{(\alpha_i-1)!} \right) \exp(C) \int \left(\prod_{i=1}^l \frac{d^d \bar{Q}_i}{\pi^{d/2}} \right) \exp \left(\sum_{i=1}^l \lambda_i \bar{Q}_i^2 \right) \\ &= (-)^{\alpha} \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i-1}}{(\alpha_i-1)!} \right) \exp(C) \Delta^{-d/2}, \end{aligned} \quad (7.20)$$

where the λ_i are the eigenvalues of A_{ij} and

$$\Delta = \prod_i \lambda_i = \det(A_{ij}). \quad (7.21)$$

In the last step of Eq. (7.20) we switched to euclidean coordinates $d^d Q_i \rightarrow d^d Q_{E,i}$ with $Q_{E,i}^2 = -Q_i^2$, using a Wick rotation, and we used the well-known Gaussian integral

$$\int d^d x \exp(-ax^2) = \left(\frac{\pi}{a} \right)^{d/2}. \quad (7.22)$$

We can also repeat the procedure with an integrand of the form

$$\mathcal{I}[\mathcal{N}(\mu_{ij})] = \int \left(\prod_{i=1}^l \frac{d^d \bar{q}_i}{\pi^{d/2}} \right) \frac{\mathcal{N}(\mu_{ij})}{D_1^{\alpha_1} \dots D_n^{\alpha_n}}, \quad (7.23)$$

i.e. an integrand whose numerator is a function of the (-2ϵ) -dimensional components of the loop momenta \bar{q}_i . It is easy to check that, with trivial modifications, everything is still valid

until Eq. (7.19), which now we formally rewrite by splitting the 4- and (-2ϵ) -dimensional integration as

$$\begin{aligned} \mathcal{I}[\mathcal{N}(\mu_{ij})] &= (-)^\alpha \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i-1}}{(\alpha_i-1)!} \right) \int \left(\prod_{i=1}^l \frac{d^4 q_i}{\pi^2} \right) \exp \left(\sum_{i,j=1}^l A_{ij} (q_i \cdot q_j) + C \right) \\ &\quad \times \int \left(\prod_{i=1}^l \frac{d^{-2\epsilon} \vec{\mu}_i}{\pi^{-\epsilon}} \right) \mathcal{N}(\mu_{ij}) \exp \left(- \sum_{i,j=1}^l A_{ij} \mu_{ij} \right) \\ &= (-)^\alpha \int \left(\prod_{i=1}^n \frac{dx_i x_i^{\alpha_i-1}}{(\alpha_i-1)!} \right) e^C \Delta^{-2} \int \left(\prod_{i=1}^l \frac{d^{-2\epsilon} \vec{\mu}_i}{\pi^{-\epsilon}} \right) \mathcal{N}(\mu_{ij}) \exp \left(- \sum_{i,j=1}^l A_{ij} \mu_{ij} \right). \end{aligned} \quad (7.24)$$

The special case $\mathcal{N} = 1$ is recovered by using

$$\int \left(\prod_{i=1}^l \frac{d^{-2\epsilon} \vec{\mu}_i}{\pi^{-\epsilon}} \right) \exp \left(- \sum_{i,j=1}^l A_{ij} \mu_{ij} \right) = \Delta^\epsilon. \quad (7.25)$$

Therefore, in order to find a recursion relation of the form of Eq. (7.2), we can proceed at any loop order as suggested in Ref. [198] for the two-loop case. We use Eq. (7.25) and we take ℓ derivatives with respect to the matrix elements A_{ij} . On one the l.h.s. this generates a factor $\mathcal{N}[\mu_{ij}]$,

$$\Delta^\epsilon \xrightarrow{\ell \text{ derivatives } \partial/\partial A_{ij}} (-)^\ell \mathcal{N}[\mu_{ij}] \Delta^\epsilon, \quad (7.26)$$

which can be interpreted as a numerator according to Eq. (7.24) (we conventionally pull out a $(-)^\ell$ which takes into account the minus sign in the exponent). On the r.h.s., for a proper combination of derivatives, the effect is a shift in the exponent $\epsilon \rightarrow \epsilon - 1$ and the multiplication by an ϵ -dependent factor $c(\epsilon)$,

$$\Delta^\epsilon \xrightarrow{\ell \text{ derivatives } \partial/\partial A_{ij}} c(\epsilon) \Delta^{\epsilon-1}. \quad (7.27)$$

When inserted in Eq. (7.20), this gives a dimensionally shifted scalar integral $\mathcal{I}^{(d+2)}$ by changing $\Delta^{-d/2} \rightarrow \Delta^{-d/2} \Delta^{-1} = \Delta^{-(d+2)/2}$. Putting everything together we end up with

$$\mathcal{I}[\mathcal{N}(\mu_{ij})] = c(\epsilon) \mathcal{I}^{(d+2)}. \quad (7.28)$$

The only issue is now finding the right combination of derivatives which has the effect described by Eq. (7.27). This is however quite easy considering that Δ , as a function of the elements A_{ij} , is nothing else than the determinant of a symmetric matrix. By following the discussion of Section 7.1, it is straightforward to see that this is in turn a Shouten polynomial of rank ℓ , defined by Eq. (7.5) followed the substitution $(k_i \cdot k_j) \rightarrow A_{ij}$. Since, with the trivial exception of the one-loop case, taking ℓ derivatives of Δ^ϵ will generate several terms proportional to $\Delta^{\epsilon-1}, \Delta^{\epsilon-2}, \dots, \Delta^{\epsilon-\ell}$, as well as to the matrix elements A_{ij} , we need a combination

of derivatives which generates other factors Δ in order to get an expression of the form of the r.h.s. of Eq. (7.27), where everything is proportional to $\Delta^{\epsilon-1}$. It turns out that one can do this by taking a combination of derivatives which has also the form of a Shouten polynomial.

Putting everything together, motivated by the (heuristic) arguments of the previous paragraph, we come up with the following algorithm

- write down the explicit expression for

$$\Delta = \det(A_{ij}) \quad (7.29)$$

as a function of the generic matrix elements A_{ij} (in other words, this is the determinant of a generic $\ell \times \ell$ symmetric matrix)

- the expression for $\mathcal{N}[\mu_{ij}]$, which is proportional to a (-2ϵ) -dimensional Shouten polynomial, is thus given by the substitution

$$\mathcal{N}[\mu_{ij}] = (-)^\ell \Delta \Big|_{A_{ij} \rightarrow 2\mu_{ij}} \quad (7.30)$$

- the right combination of derivatives is in turn the differential operator \mathcal{D} which is formally obtained from the algebraic expression of Δ using the substitutions

$$\mathcal{D} = (-)^\ell \Delta \quad \text{with} \quad A_{ij} \rightarrow \begin{cases} \frac{\partial}{\partial A_{ij}} & \text{if } i \neq j \\ 2\frac{\partial}{\partial A_{ij}} & \text{if } i = j, \end{cases} \quad (7.31)$$

where the extra factor 2 for $i = j$ compensates the missing factor two in the exponent $\sum_{ij} A_{ij}\mu_{ij}$ which is only present for non-diagonal elements

- with these definitions, the recurrence relation is thus given by

$$\mathcal{I}[\mathcal{N}(\mu_{ij})] = \frac{1}{\Delta^{\epsilon-1}} (\mathcal{D}\Delta^\epsilon) \mathcal{I}^{(d+2)}. \quad (7.32)$$

In the last equation, $(\mathcal{D}\Delta)/\Delta^{\epsilon-1} = c(\epsilon)$, i.e. it only depends on ϵ and not on the matrix elements A_{ij} . This is a necessary and sufficient condition for the algorithm to work, which we verified up to 4 loops.

The algorithm has been implemented in a small MATHEMATICA code which we used to find the recurrence relation from one to four loops. Here are the results

- one loop

$$\mathcal{I}[\mu_{11}] = -\epsilon \mathcal{I}^{(d+2)} \quad (7.33)$$

- two loops

$$4\mathcal{I}[\mu_{11}\mu_{22} - \mu_{12}^2] = 2\epsilon(1 + 2\epsilon)\mathcal{I}^{(d+2)} \quad (7.34)$$

- three loops

$$8\mathcal{I}[\mu_{13}^2\mu_{22} - 2\mu_{12}\mu_{13}\mu_{23} + \mu_{11}\mu_{23}^2 + \mu_{12}^2\mu_{33} - \mu_{11}\mu_{22}\mu_{33}] = 4\epsilon(1+\epsilon)(1+2\epsilon)\mathcal{I}^{(d+2)} \quad (7.35)$$

- four loops

$$\begin{aligned} & 16\mathcal{I}[\mu_{14}^2\mu_{23}^2 - 2\mu_{13}\mu_{14}\mu_{23}\mu_{24} + \mu_{13}^2\mu_{24}^2 - \mu_{14}^2\mu_{22}\mu_{33} + 2\mu_{12}\mu_{14}\mu_{24}\mu_{33} \\ & \quad - \mu_{11}\mu_{24}^2\mu_{33} + 2\mu_{13}\mu_{14}\mu_{22}\mu_{34} - 2\mu_{12}\mu_{14}\mu_{23}\mu_{34} - 2\mu_{12}\mu_{13}\mu_{24}\mu_{34} \\ & \quad + 2\mu_{11}\mu_{23}\mu_{24}\mu_{34} + \mu_{12}^2\mu_{34}^2 - \mu_{11}\mu_{22}\mu_{34}^2 - \mu_{13}^2\mu_{22}\mu_{44} + 2\mu_{12}\mu_{13}\mu_{23}\mu_{44} \\ & \quad - \mu_{11}\mu_{23}^2\mu_{44} - \mu_{12}^2\mu_{33}\mu_{44} + \mu_{11}\mu_{22}\mu_{33}\mu_{44}] \\ & = 4\epsilon(1+\epsilon)(1+2\epsilon)(3+2\epsilon)\mathcal{I}^{(d+2)}. \end{aligned} \quad (7.36)$$

In the following sections, we will use these recurrence relations at one and two loops in order to find the corresponding ones with the form of Eq. (7.3) and (7.4). Then, by performing an integrand reduction of the numerator

$$S(4; q_1, \dots, q_\ell, k_1, \dots, k_{n-1}) \quad \text{or} \quad S(d; \bar{q}_1, \dots, \bar{q}_\ell, k_1, \dots, k_{n-1})$$

we will find, in several examples, identities which so far have been traditionally computed with IBP methods or tensor decomposition of integrals.

7.3 One-loop relations

At one loop we should use the recurrence relation in Eq. (7.33), which using the common notation $\mu^2 \equiv \mu_{11}$ reads

$$\mathcal{I}[\mu^2] = -\epsilon\mathcal{I}^{(d+2)}. \quad (7.37)$$

In order to find a similar relation for a four-dimensional numerator, we can use the following tensor decomposition which is valid for any number of external legs

$$\mathcal{I}[\bar{q}^\mu \bar{q}^\nu] = g_{(d)}^{\mu\nu} A_{00} + \mathcal{O}(k)^{\mu\nu}, \quad (7.38)$$

where we introduced the notation $\mathcal{O}(k)^{\mu_1 \dots \mu_r}$ which denotes a rank- r tensor proportional to at least one external momentum $k_i^{\mu_j}$. As shown in Ref. [199], by contracting this decomposition with $g_{-2\epsilon}^{\mu\nu}$ one finds

$$\mathcal{I}[\mu^2] = (2\epsilon) A_{00}, \quad (7.39)$$

which combined with Eq. (7.33) gives

$$A_{00} = -\frac{1}{2}\mathcal{I}^{(d+2)}. \quad (7.40)$$

If we instead contract the tensor decomposition with $v_\perp^\mu v_\perp^\nu$, where v_\perp^μ is any orthogonal vector such that $(v_\perp \cdot k_i) = 0$, we obtain

$$\mathcal{I}[(q \cdot v_\perp)^2] = v_\perp^2 A_{00}, \quad (7.41)$$

which combined with Eq. (7.40) gives

$$\mathcal{I}[(q \cdot v_\perp)^2] = -\frac{v_\perp^2}{2} \mathcal{I}^{(d+2)}. \quad (7.42)$$

A particularly convenient choice of v_\perp is the one of Eq. (7.8), which using Eq. (7.9) gives

$$\mathcal{I}[S(4; q, k_1, \dots, k_{n-1})] = -\frac{v_\perp^2}{2} \mathcal{I}^{(d+2)}. \quad (7.43)$$

In this case the computation of v_\perp^2 can be performed by means Eq. (7.6) or by using a Computer Algebra System such as FORM which has an efficient implementation of the contraction appearing in the same equation. Equivalently, we can choose the d -dimensional $v_{(d)\perp}$ defined by Eq. (7.12), which using Eq. (7.14) gives

$$\mathcal{I}[S(d; \bar{q}, k_1, \dots, k_{n-1})] = -\frac{v_{(d)\perp}^2}{2} \mathcal{I}^{(d+2)}. \quad (7.44)$$

Notice that, in general, while v_\perp^2 only depends on the kinematic invariants, $v_{(d)\perp}^2$ also depends on d .

Similar relations can also be found for higher-rank (i.e. non-scalar) integrals. From the decomposition

$$\mathcal{I}[\bar{q}^\mu \bar{q}^\nu \bar{q}^\rho \bar{q}^\sigma] = A_{0000} \left(g_{(d)}^{\mu\nu} g_{(d)}^{\rho\sigma} + g_{(d)}^{\mu\rho} g_{(d)}^{\nu\sigma} + g_{(d)}^{\mu\sigma} g_{(d)}^{\nu\rho} \right) + \mathcal{O}(k)^{\mu\nu\rho\sigma}, \quad (7.45)$$

we can easily derive

$$2\epsilon v_{\perp,2}^2 A_{0000} = \mathcal{I}_n[\mu^2 (q \cdot v_\perp)^2] \quad (7.46)$$

$$v_\perp^2 A_{0000} = -\frac{1}{2} \mathcal{I}_n^{(d+2)}[(q \cdot v_\perp)^2], \quad (7.47)$$

where v_\perp could either be the Shouten vector defined in Eq. (7.8) or any other four-dimensional vector orthogonal to the external momenta k_i . Alternatively, one can shift in $d+4$ using twice Eq. (7.37), which yields

$$-\epsilon(1-\epsilon) \mathcal{I}_n^{(d+4)} = \mathcal{I}_n[\mu^4] \quad (7.48)$$

$$-4\epsilon(1-\epsilon) A_{0000} = \mathcal{I}_n[\mu^4] \quad (7.49)$$

$$A_{0000} = \frac{1}{4} \mathcal{I}_n^{(d+4)}. \quad (7.50)$$

By comparison of Eq. (7.47) and (7.50) we have

$$\mathcal{I}_n^{(d+2)}[(q \cdot v_{\perp,2})^2] = -\frac{v_{\perp,2}^2}{2} \mathcal{I}^{(d+4)} \quad (7.51)$$

or

$$\mathcal{I}_n[(q \cdot v_\perp)^2] = -\frac{v_\perp^2}{2} \mathcal{I}^{(d+2)}, \quad (7.52)$$

which is valid for any four-dimensional vector v_{\perp}^{μ} orthogonal to the external legs k_i of the loop diagram.

In the following we will apply this strategy to the Master Integrals of the one-loop integrand reduction, finding suitable recurrence relations. The recurrence relations for scalar integrals were already computed in Ref. [23] with a different method. We will then show how this new technique can also be used to find further relations, such as those involving higher-rank integrals. Next we will show some more specific examples where we reproduce known IBP identities with the method outlined in this chapter. In the one-loop case, this consists in performing an integrand reduction of $S(4; q, k_1, \dots, k_{n-1})$ and combine the result with the basic recurrence relations above.

7.3.1 Recurrence relations for 1-point integrals

For tadpoles

$$\mathcal{I}_0[\mathcal{N}] = \frac{\mathcal{N}}{D_0} \quad (7.53)$$

we have no external leg, hence we take $v_{\perp}^{\mu} = \epsilon_{(4)}^{\mu}$ and $v_{\perp}^2 = 4$. The Shouten polynomial is

$$S(q) = q^2 \quad (7.54)$$

which is trivially decomposed as

$$S(q) = D_0 + \mu^2 + m_0^2. \quad (7.55)$$

After integration, using Eq. (7.37) and (7.42) one gets

$$-2\mathcal{I}_0^{(d+2)} = \frac{d-4}{2}\mathcal{I}_0^{(d+2)} + m_0^2\mathcal{I}_0. \quad (7.56)$$

Hence, the recurrence relation for scalar tadpoles is

$$-2m_0^2\mathcal{I}_0 = d\mathcal{I}_0^{(d+2)}. \quad (7.57)$$

The same result could have been obtained using Eq. (7.44) and $v_{(d)\perp}^2 = d$.

We can also work out rank-2 tadpoles, which will be useful later. Since there are no external legs, Eq. (7.52) is valid for a generic vector v^{μ} ,

$$\mathcal{I}_0[(q \cdot v)^2] = -\frac{v^2}{2}\mathcal{I}_0^{(d+2)} = \frac{v^2 m_0^2}{d}\mathcal{I}_0. \quad (7.58)$$

A more general case can be derived from this by replacing $v \rightarrow v + w$, which combined with the previous one yields

$$\mathcal{I}_0[(q \cdot v)(q \cdot w)] = \frac{(v \cdot w) m_0^2}{d}\mathcal{I}_0 = -\frac{(v \cdot w)}{2}\mathcal{I}_0^{(d+2)}. \quad (7.59)$$

The first equality is the same relation one would get from a Passarino-Veltman decomposition.

7.3.2 Recurrence relations for 2-point integrals

We consider the most general bubble

$$\mathcal{I}_{01}[\mathcal{N}] = \frac{\mathcal{N}}{D_0 D_1}, \quad \text{with } D_0 = \bar{q}^2 - m_0^2, \quad D_1 = (\bar{q} + k)^2 - m_1^2. \quad (7.60)$$

The recursion relation of Eq. (7.43) becomes

$$\mathcal{I}_{01}[S(4; q, k)] = -\frac{3k^2}{2} \mathcal{I}_{01}^{(d+2)}, \quad (7.61)$$

with

$$S(4; q, k) = q^2 k^2 - (q \cdot k)^2. \quad (7.62)$$

In the following we distinguish the general case $k^2 \neq 0$ from the degenerate cases $k^2 = 0, m_0 \neq m_1$ and $k^2 = 0, m_0 = m_1$. In particular, if $k^2 = 0$ we will no longer have a dimensional recurrence from the previous equation, but in these cases the scalar bubbles are known to be reducible to tadpoles and we will show how to find the corresponding identities with our method.

Case $k^2 \neq 0$

We choose a basis $\{k, E_2, e_3, e_4\}$ with

$$k \cdot E_2 = k \cdot e_3 = k \cdot e_4 = E_2 \cdot e_3 = E_2 \cdot e_4 = e_3^2 = e_4^2 = 0.$$

The integrand reduction via polynomial division of $S(4; q, k)$ yields the decomposition

$$\begin{aligned} S(4; q, k) = & \left(-\frac{1}{4} k^4 + \frac{1}{2} m_1^2 k^2 - \frac{1}{4} m_1^4 + \frac{1}{2} m_0^2 k^2 + \frac{1}{2} m_0^2 m_1^2 - \frac{1}{4} m_0^4 \right) \\ & + (k^2) \mu^2 \\ & + \left(\frac{1}{2} ((q+k) \cdot k) + \frac{1}{4} (k^2 + m_1^2 - m_0^2) \right) D_0 \\ & + \left(-\frac{1}{2} (q \cdot k) + \frac{1}{4} (k^2 - m_1^2 + m_0^2) \right) D_1, \end{aligned} \quad (7.63)$$

which integrated (using also Eq. (7.59)) gives the general recurrence relation for scalar bubbles

$$\begin{aligned} -\frac{(d-1)k^2}{2} \mathcal{I}_{01}^{(d+2)} = & \left(-\frac{1}{4} k^4 + \frac{1}{2} m_1^2 k^2 - \frac{1}{4} m_1^4 + \frac{1}{2} m_0^2 k^2 + \frac{1}{2} m_0^2 m_1^2 - \frac{1}{4} m_0^4 \right) \mathcal{I}_{01} \\ & + \frac{1}{4} (k^2 + m_1^2 - m_0^2) \mathcal{I}_1 \\ & + \frac{1}{4} (k^2 - m_1^2 + m_0^2) \mathcal{I}_0. \end{aligned} \quad (7.64)$$

With this choice of basis, the only non-spurious term in the 2-point residue for a renormalizable theory is the quadratic bubble $(q \cdot E_2)^2$. Using Eq. (7.52), we simply have

$$\mathcal{I}_{01}[(q \cdot E_2)^2] = -\frac{E_2^2}{2} \mathcal{I}_{01}^{(d+2)}, \quad (7.65)$$

where the r.h.s. can be directly read from Eq. (7.64) in terms of scalar bubbles and tadpole integrals in d dimensions only.

Case $k^2 = 0$ and $m_0 \neq m_1$

If $k^2 = 0$ the previous choice of basis is not valid anymore (we cannot find three independent vectors which are orthogonal to a massless vector). Therefore we replace the orthogonal vector E_2 with a reference vector e_2 such that $e_2^2 = 0$ but $k \cdot e_2 \neq 0$.

We can first observe that Eq. (7.64), which is valid in the general case, is smooth in the limit $k^2 \rightarrow 0$. In this limit however the dependence on $\mathcal{I}_{01}^{(d+2)}$ drops out and we will no longer have a recurrence relation, but an homogeneous relation in the number d of dimensions which reduces the scalar bubble to tadpoles

$$\mathcal{I}_{01} = \frac{1}{m_1^2 - m_0^2} (\mathcal{I}_1 - \mathcal{I}_0). \quad (7.66)$$

If a recurrence relation is needed, the r.h.s. can be shifted in $d+2$ or $d-2$ using the recurrence relation for tadpoles in Eq. (7.57).

For the linear term $q \cdot e_2$ in the 2-point residue, we can start from a different relation. We derive it from

$$\mathcal{I}_{01}[q^\mu q^\nu q^\rho] = (g^{\mu\nu} k^\rho + g^{\mu\rho} k^\nu + g^{\nu\rho} k^\mu) B_{001} + \mathcal{O}(k)^{\mu\nu\rho} \quad (7.67)$$

Assuming $k^2 = e_2^2 = 0$, one can see that also the Shouten vector $v_\perp^2 = 0$ and we get

$$\mathcal{I}_{01}[S(4; q, k)(q \cdot e_2)] = 0. \quad (7.68)$$

The integrand reduction reads

$$S(4; q, k) = -\frac{1}{4}(q \cdot e_2) m_1^4 + \frac{1}{2}(q \cdot e_2) m_0^2 m_1^2 - \frac{1}{4}(q \cdot e_2) m_0^2 + \Delta_0 D_1 + \Delta_1 D_0. \quad (7.69)$$

where the tadpole residues Δ_0 and Δ_1 have rank 2 and can be integrated using relations we already found, namely

$$\mathcal{I}_i[(q \cdot e_3)(q \cdot e_4)] = \frac{(e_3 \cdot e_4) m_i^2}{d} \mathcal{I}_i, \quad \mathcal{I}_i[\mu^2] = \frac{d-4}{2} \mathcal{I}_i^{(d+2)} = -\frac{d-4}{d} m_i^2 \mathcal{I}_i. \quad (7.70)$$

Putting everything together we get

$$(m_1^2 - m_0^2)^2 \mathcal{I}_{01}[(q \cdot e_2)] = (e_2 \cdot k) \left(\frac{2}{d} m_1^2 - m_1^2 + m_0^2 \right) \mathcal{I}_1 + (e_2 \cdot k) \left(-\frac{2}{d} m_0^2 \right) \mathcal{I}_0. \quad (7.71)$$

For the quadratic term $(q \cdot e_2)^2$ in the 2-point residue, we use instead the recurrence relation

$$\mathcal{I}[S(4; q, k)(q \cdot e_2)^2] = -\frac{(k \cdot e_2)^2}{2} \mathcal{I}^{(d+4)}, \quad (7.72)$$

which can be derived from Eq. (7.50) and $(e_2 \cdot v_\perp)^2 = S(4; e_2, k) = -(e_2 \cdot k)^2$. By performing an integrand reduction of the l.h.s. and integrating with same formulas we used in the previous cases, we get

$$\begin{aligned} -\frac{(k \cdot e_2)^2}{2} \mathcal{I}^{(d+4)} &= \mathcal{I}[S(4; q, k)(q \cdot e_2)^2] \\ &= -\frac{1}{4}(m_1^2 - m_0^2)^2 \mathcal{I}_{01}[(q \cdot e_2)^2] + \frac{(e_2 \cdot k)^2}{4} \left(-\frac{4}{d} m_1^2 + m_1^2 - m_0^2 \right) \mathcal{I}_1. \end{aligned} \quad (7.73)$$

The result is therefore

$$\frac{1}{4}(m_1^2 - m_0^2)^2 \mathcal{I}_{01}[(q \cdot e_2)^2] = \frac{(k \cdot e_2)^2}{2} \mathcal{I}^{(d+4)} + \frac{(e_2 \cdot k)^2}{4} \left(-\frac{4}{d} m_1^2 + m_1^2 - m_0^2 \right) \mathcal{I}_1. \quad (7.74)$$

The integral $\mathcal{I}^{(d+4)}$ cannot be shifted back into d directly, but we can use Eq. (7.66) to reduce it to tadpoles, which then can be easily shifted using

$$d \mathcal{I}_i^{(d+2)} = -2m_i^2 \mathcal{I}_i \quad \Rightarrow \quad d(d+2) \mathcal{I}_i^{(d+4)} = 4m_i^4 \mathcal{I} \quad (7.75)$$

hence

$$\mathcal{I}^{(d+4)} = \frac{1}{m_1^2 - m_0^2} \left(\mathcal{I}_1^{(d+4)} - \mathcal{I}_0^{(d+4)} \right) = \frac{4}{d(d+2)(m_1^2 - m_0^2)} \left(m_1^4 \mathcal{I}_1 - m_0^4 \mathcal{I}_0 \right), \quad (7.76)$$

which plugged back into Eq. (7.74) expresses the quadratic bubble in terms of tadpoles, in the same number of dimensions.

Case $k^2 = 0$ and $m_0 = m_1$

As one can see, most of the formulas we derived for $k^2 = 0$ break down if $m_0 = m_1$ due to a division by zero. However, Eq. (7.74) above is also valid in this case and becomes

$$\mathcal{I}_{01}^{(d+4)} = \frac{2}{d} m_1^2 \mathcal{I}_1. \quad (7.77)$$

Using Eq. (7.75) we get an homogeneous equation in $d+4$

$$\mathcal{I}_{01}^{(d+4)} = \frac{d+2}{2m_1^2} \mathcal{I}_1^{(d+4)}, \quad (7.78)$$

which after $d \rightarrow d-4$ becomes

$$\mathcal{I}_{01} = \frac{d-2}{2m_1^2} \mathcal{I}_1. \quad (7.79)$$

Higher rank terms can in principle be computed starting from recurrence relations of rank 3 or higher.

7.3.3 Recurrence relations for 3-, 4- and 5-point integrals

A 3-point one-loop integrand has the form

$$\mathcal{I}_{012}[\mathcal{N}] = \frac{\mathcal{N}}{D_0 D_1 D_2}, \quad (7.80)$$

By performing the integrand reduction of $\mathcal{I}[(q \cdot v_\perp)^2] = \mathcal{I}[S(4; k_1, k_2)]$, where k_1 and k_2 are, as usual, two of the external legs, we get a decomposition of the form

$$\frac{1}{v_\perp^2} S(4; k_1, k_2) = c_0^{(012)} + \frac{1}{2} \mu^2 + \text{subdiagrams}. \quad (7.81)$$

After integration, using the formulas in the previous subsections for the linear bubbles, we find that all the tadpole contributions drop out and we get a relation of the form

$$\frac{1}{4}(2-d)\mathcal{I}_{012}^{(d+2)} = c_0^{(012)}\mathcal{I}_{012} + \sum_{ij} c_{ij}\mathcal{I}_{ij}, \quad (7.82)$$

which is the recurrence relation for scalar triangles.

The 4-point integrals

$$\mathcal{I}_{0123}[\mathcal{N}] = \frac{\mathcal{N}}{D_0 D_1 D_2 D_3}, \quad (7.83)$$

can be treated in a similar way, using $\mathcal{N} = S(q, k_1, k_2, k_3)$ as integrand, whose decomposition reads

$$\begin{aligned} \frac{1}{v_\perp^2} S(q, k_1, k_2, k_3) &= c_0^{(0123)} + \mu^2 \\ &+ \left(c_0^{(0123)} + c_1^{(012)}(q \cdot e_3^{(012)}) + c_4^{(012)}(q \cdot e_4^{(012)}) \right) D_3 \\ &+ \left(c_0^{(013)} + c_1^{(013)}(q \cdot e_3^{(013)}) + c_4^{(013)}(q \cdot e_4^{(013)}) \right) D_2 \\ &+ \left(c_0^{(023)} + c_1^{(023)}(q \cdot e_3^{(023)}) + c_4^{(023)}(q \cdot e_4^{(023)}) \right) D_1 \\ &+ \left(c_0^{(123)} + c_1^{(123)}(q \cdot e_3^{(123)}) + c_4^{(123)}(q \cdot e_4^{(123)}) \right) D_0. \end{aligned} \quad (7.84)$$

Integrating we get the recurrence relation for scalar box integrals

$$\frac{1}{2}(3-d)\mathcal{I}_{0123}^{(d+2)} = c_0^{(0123)}\mathcal{I}_{0123} + \sum_{ijk} c_0^{(ijk)}\mathcal{I}_{ijk}. \quad (7.85)$$

Finally, for 5-point integrals,

$$\mathcal{I}_{01234}[\mathcal{N}] = \frac{\mathcal{N}}{D_0 D_1 D_2 D_3 D_4}, \quad (7.86)$$

there is no orthogonal vector (since in general we have 4 independent external legs), but we can directly use μ^2 for the integrand reduction. Indeed, the polynomial division of μ^2 reads

$$\begin{aligned} \mu^2 &= c_0^{(01234)} \\ &+ \left(c_0^{(0123)} + c_1^{(0123)}(q \cdot v_\perp^{(0123)}) \right) D_4 \\ &+ \left(c_0^{(0124)} + c_1^{(0124)}(q \cdot v_\perp^{(0124)}) \right) D_3 \\ &+ \left(c_0^{(0134)} + c_1^{(0134)}(q \cdot v_\perp^{(0134)}) \right) D_2 \\ &+ \left(c_0^{(0234)} + c_1^{(0234)}(q \cdot v_\perp^{(0234)}) \right) D_1 \\ &+ \left(c_0^{(1234)} + c_1^{(1234)}((q + p_1) \cdot v_\perp^{(1234)}) \right) D_0 \end{aligned} \quad (7.87)$$

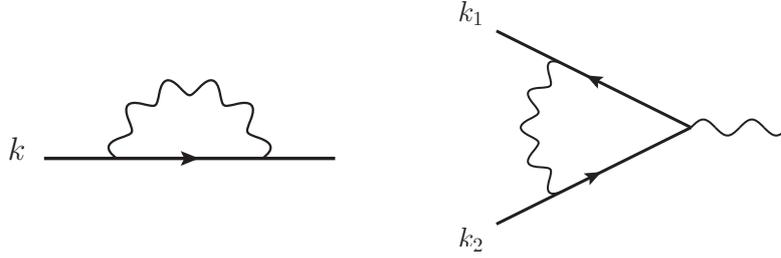


Figure 7.1: Two- and three-point one-loop diagrams in QED, which can be reduced at the integral level when the external fermion legs are on-shell.

which integrated gives the recurrence relation for pentagon integrals

$$\begin{aligned} \frac{d-4}{2} \mathcal{I}_{01234}^{(d+2)} &= c_0^{(01234)} \mathcal{I}_{01234} \\ &+ c_0^{(0123)} \mathcal{I}_{0123} + c_0^{(0124)} \mathcal{I}_{0124} + c_0^{(0134)} \mathcal{I}_{0134} \\ &+ c_0^{(0234)} \mathcal{I}_{0234} + c_0^{(1234)} \mathcal{I}_{1234}. \end{aligned} \quad (7.88)$$

One can easily see, by power counting, that $\mathcal{I}_{01234}^{(d+2)}$ is finite and therefore the l.h.s. of the previous equation vanishes for $d \rightarrow 4$. For this reason, for one-loop calculations in $d = 4 - 2\epsilon$ dimensions the pentagons can be reduced to boxes. As we have already seen in Section 3.5, this also implies that we can choose $\Delta_{i_1 \dots i_k} = c_0 \mu^2$ instead of $\Delta_{i_1 \dots i_k} = c_0$ for the parametric form of the residue of the pentagons. In this case, pentagons are spurious, i.e. they don't give contributions to the final term of one-loop scattering amplitudes. Within the Laurent expansion method for the integrand reduction implemented in the library NINJA (see Chapter 4), this is further exploited by completely skipping the computation of 5-point residues during the reduction.

7.3.4 Examples with QED kinematics

In this Section we reproduce two known one-loop IBP relations, involving the diagrams in Fig. 7.1, by combining recurrence relations found with the method described above.

We start from a 2-point integral \mathcal{I}_{01} with the following kinematics

$$D_0 = q^2, \quad D_1 = q^2 + 2(q \cdot k), \quad (\text{i.e. } m_0^2 = 0, \quad k^2 = m_1^2 = m^2). \quad (7.89)$$

We can use Eq. (7.64), which in this case reads

$$-\frac{(d-1)m^2}{2} \mathcal{I}_{01}^{(d+2)} = \frac{m^2}{2} \mathcal{I}_1 \quad (7.90)$$

Using Eq. (7.57) to shift the tadpole integral in $d+2$ we get

$$(d-1) \mathcal{I}_{01}^{(d+2)} = \frac{d}{2m^2} \mathcal{I}_1^{(d+2)}. \quad (7.91)$$

After the shift $d \rightarrow d - 2$ we obtain

$$(d - 3) \mathcal{I}_{01} = \frac{1}{2m^2} (d - 2) \mathcal{I}_1. \quad (7.92)$$

This is the same relation one would get by using IBP on the integrands $\mathcal{I}_{01}[\bar{q}^\mu]$ and $\mathcal{I}_1[\bar{q}^\mu]$. Similarly, the rank-2 relation can be obtained from Eq. (7.65) and (7.90),

$$(d - 1) \mathcal{I}_{01}[(q \cdot E_2)^2] = \frac{E_2^2}{2} \mathcal{I}_1. \quad (7.93)$$

Next we consider the three-point integral \mathcal{I}_{012} with kinematics corresponding to a QED vertex with on-shell fermions,

$$\begin{aligned} D_0 &= \bar{q}^2, & D_1 &= (\bar{q} + k_1)^2 - m^2, & D_2 &= (\bar{q} - k_2)^2 - m^2, \\ \text{with } m_0^2 &= 0, & k_1^2 &= k_2^2 = m_1^2 = m_2^2 = m^2, & (k_1 + k_2)^2 &= s. \end{aligned} \quad (7.94)$$

The recurrence relations for the scalar triangle \mathcal{I}_{012} and the bubble subdiagram \mathcal{I}_{01} are particularly simple

$$(2 - d) \mathcal{I}_{012}^{(d+2)} = \mathcal{I}_{12} \quad (7.95)$$

$$(1 - d) \mathcal{I}_{12}^{(d+2)} = \frac{4m^2 - s}{2} \mathcal{I}_{12} + \mathcal{I}_1. \quad (7.96)$$

We adopt a bottom up approach. We first use the recurrence relation of Eq. (7.57) for tadpoles and we plug it into Eq. (7.96), which then expresses \mathcal{I}_{12} in terms of integrals in $d + 2$. The result is then substituted in Eq. (7.95) which becomes homogeneous in the number of dimensions,

$$(2 - d) \mathcal{I}_{012}^{(d+2)} = \frac{2}{4m^2 - s} \left((1 - d) \mathcal{I}_{12}^{(d+2)} + \frac{d}{2m^2} \mathcal{I}_1^{(d+2)} \right).$$

After the shift $d \rightarrow d - 2$ we get,

$$(4 - d) \mathcal{I}_{012} = \frac{2}{4m^2 - s} \left((3 - d) \mathcal{I}_{12} + \frac{d - 2}{2m^2} \mathcal{I}_1 \right), \quad (7.97)$$

which is also a relation which can be obtained as an IBP identity.

7.4 Two-loop relations

For any two-loop topology, we define the form factors A and B

$$\mathcal{I}[\bar{q}_1^\mu \bar{q}_1^\nu \bar{q}_2^\rho \bar{q}_2^\sigma] = A \bar{g}^{\mu\nu} \bar{g}^{\rho\sigma} + B(\bar{g}^{\mu\rho} \bar{g}^{\nu\sigma} + \bar{g}^{\mu\sigma} \bar{g}^{\nu\rho}) + \mathcal{O}(k_i)^{\mu\nu\rho\sigma}. \quad (7.98)$$

By contracting this with $4g_{\mu\rho}^{(-2\epsilon)} g_{\nu\sigma}^{(-2\epsilon)} - 4g_{\mu\nu}^{(-2\epsilon)} g_{\rho\sigma}^{(-2\epsilon)}$ we get

$$\mathcal{I}[4\mu_{12}^2 - 4\mu_{11}\mu_{22}] = 8\epsilon(1 + 2\epsilon)(B - A). \quad (7.99)$$

We recall the basic two-loop recurrence relation of Eq. (7.34)

$$4\mathcal{I}[\mu_{11}\mu_{22} - \mu_{12}^2] = 2\epsilon(1 + 2\epsilon)\mathcal{I}^{(d+2)}, \quad (7.100)$$

which combined with Eq. (7.99) gives

$$B - A = -\frac{1}{4}\mathcal{I}^{(d+2)}. \quad (7.101)$$

In order to find an analogous relation for four-dimensional numerators, we use an *antisymmetric, orthogonal tensor* $t_{\perp}^{\mu\nu}$ which satisfies

$$t_{\perp}^{\mu\nu} = -t_{\perp}^{\nu\mu}, \quad t_{\perp\mu\nu} k_i^{\mu} = t_{\perp\mu\nu} k_i^{\nu} = 0, \quad (7.102)$$

for every external leg k_i of the diagram. A particularly convenient choice of $t_{\perp}^{\mu\nu}$ is the one based on the Shouten polynomials (see Section 7.1) defined in Eq. (7.10) which at two-loops reads

$$t_{\perp\mu\nu} = \epsilon_{\mu\nu\rho_1\dots\rho_{n-1}}^{(4)} k_1^{\rho_1} \dots k_{n-1}^{\rho_{n-1}}, \quad (7.103)$$

or its d -dimensional version

$$t_{(d)\perp\mu\nu} = \epsilon_{\mu\nu\rho_1\dots\rho_{n-1}}^{(d)} k_1^{\rho_1} \dots k_{n-1}^{\rho_{n-1}}. \quad (7.104)$$

If we contract Eq. (7.98) with $t_{\perp\mu\sigma}t_{\perp\nu\rho}$ we obtain

$$\mathcal{I}[t_{\perp}(q_1, q_2)^2] = -t_{\perp}^2 (B - A), \quad (7.105)$$

where

$$t_{\perp}^2 \equiv t_{\perp}^{\mu\nu} t_{\perp\mu\nu} = -t_{\perp}^{\mu\nu} t_{\perp\nu\mu}. \quad (7.106)$$

Hence, by combining Eq. (7.105) and (7.101), we get the following recurrence relation

$$\mathcal{I}[t_{\perp}(q_1, q_2)^2] = \frac{t_{\perp}^2}{4}\mathcal{I}^{(d+2)}, \quad (7.107)$$

which with the choices of Eq. (7.103) and Eq. (7.104) reads

$$\mathcal{I}[S(4; q_1, q_2, k_1, \dots, k_{n-1})] = \frac{t_{\perp}^2}{4}\mathcal{I}^{(d+2)} \quad (7.108)$$

and

$$\mathcal{I}[S(d; \bar{q}_1, \bar{q}_2, k_1, \dots, k_{n-1})] = \frac{t_{(d)\perp}^2}{4}\mathcal{I}^{(d+2)}. \quad (7.109)$$

respectively.

Similarly, starting from

$$\mathcal{I}[\bar{q}_i^{\mu}\bar{q}_j^{\nu}] = A_{00}^{(ij)}\bar{g}^{\mu\nu} + \mathcal{O}(k_l)^{\mu\nu} \quad (7.110)$$

and using a vector v_{\perp}^{μ} orthogonal to the external legs k_i , we get

$$\mathcal{I}[\mu_{ij}] = \frac{2\epsilon}{v_{\perp}^2}\mathcal{I}[(q_i \cdot v_{\perp})(q_j \cdot v_{\perp})]. \quad (7.111)$$

Notice that with the (Shouten) choice of Eq. (7.8) for v_{\perp}^{μ} , the integrand on the r.h.s. of Eq. (7.111) can also be worked out by means of Eq. (7.6).

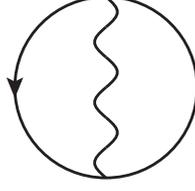


Figure 7.2: Two-loop vacuum diagram in QED, which can be reduced to a product of one-loop tadpoles.

7.4.1 A simple example

As a simple example, we consider the two-loop vacuum topology of \mathcal{I}_{123} with

$$\mathcal{I}_{123}[\mathcal{N}] = \frac{\mathcal{N}}{D_1 D_2 D_3}$$

and loop denominators

$$\begin{aligned} D_1 &= \bar{q}_1^2 - m^2 = q_1^2 - m^2 - \mu_{11} \\ D_2 &= \bar{q}_2^2 - m^2 = q_2^2 - m^2 - \mu_{22} \\ D_3 &= (\bar{q}_1 - \bar{q}_2)^2 = (q_1 - q_2)^2 - \mu_{11} - \mu_{22} + 2\mu_{12}, \end{aligned} \quad (7.112)$$

which is represented by the diagram in Fig. 7.2.

Since in this case $t_{\perp}^2 = 12$ and $v_{\perp}^2 = 4$, Eq. (7.108), (7.100) and (7.111) become

$$\mathcal{I}[S(4; q_1, q_2)] = 3\mathcal{I}^{(d+2)} \quad (7.113)$$

$$\mathcal{I}[4\mu_{12}^2 - 4\mu_{11}\mu_{22}] = -(d-4)(d-5)\mathcal{I}^{(d+2)} \quad (7.114)$$

$$\mathcal{I}[\mu_{ij}] = \frac{4-d}{4}\mathcal{I}[(q_i \cdot v_{\perp})(q_j \cdot v_{\perp})], \quad (7.115)$$

respectively, with

$$S(4; q_1, q_2) = q_1^2 q_2^2 - (q_1 \cdot q_2)^2. \quad (7.116)$$

We start from the integrand reduction of $\mathcal{I}[S(4; q_1, q_2)]$, which integrated gives

$$3\mathcal{I}_{123}^{(d+2)} = \mathcal{I}_{123}[S(4; q_1, q_2)] \quad (7.117)$$

$$= -\frac{1}{4}\mathcal{I}_{123}[4\mu_{12}^2 - 4\mu_{11}\mu_{22}] - m^2\mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}] + \frac{m^2}{2}\mathcal{I}_{12}. \quad (7.118)$$

Since \mathcal{I}_{12} is the product of two one-loop tadpoles, we can use Eq. (7.57) twice which yields

$$d^2\mathcal{I}_{12}^{(d+2)} = 4m^4\mathcal{I}_{12}^{(d)}. \quad (7.119)$$

Hence, the only missing recurrence relation needed to bring all the integrals in Eq. (7.118) in $d + 2$ dimensions is the one of $\mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}]$. Using Eq. (7.115) we have

$$\begin{aligned}\mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}] &= \frac{4-d}{4} \mathcal{I}_{123}[2(q_1 \cdot v_\perp)(q_2 \cdot v_\perp) - (q_1 \cdot v_\perp)^2 - (q_2 \cdot v_\perp)^2] \\ &= \frac{4-d}{4} \mathcal{I}_{123}[2(q_1 \cdot q_2) - q_1^2 - q_2^2].\end{aligned}\quad (7.120)$$

After an (easy) integrand reduction of the r.h.s., we get

$$\mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}] = \frac{4-d}{4} \left(\mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}] - \mathcal{I}_{12} \right).\quad (7.121)$$

hence

$$-d \mathcal{I}_{123}[2\mu_{12} - \mu_{11} - \mu_{22}] = (4-d) \mathcal{I}_{12}.\quad (7.122)$$

Plugging all the recursion relations in Eq. (7.118) we finally get

$$\mathcal{I}_{123}^{(d+2)} = \frac{d}{2m^2(d-1)} \mathcal{I}_{12}^{(d+2)}.\quad (7.123)$$

which after the shift $d \rightarrow d - 2$ becomes

$$\mathcal{I}_{123} = \frac{d-2}{2m^2(d-3)} \mathcal{I}_{12}.\quad (7.124)$$

This is a two-loop IBP identity, which we calculated here with our new method.

Alternatively, one can use $\mathcal{I}[S(d; \bar{q}_1, \bar{q}_2)]$ which satisfies Eq. (7.109), that in this case reads

$$\mathcal{I}[S(d; \bar{q}_1, \bar{q}_2)] = \frac{d(d-1)}{4} \mathcal{I}_{123}^{(d+2)}\quad (7.125)$$

and with a similar computation one gets, after integrand reduction and integration of the l.h.s.

$$\frac{d(d-1)}{4} \mathcal{I}_{123}^{(d+2)} = \mathcal{I}_{123}[S(d; \bar{q}_1, \bar{q}_2)] = \frac{m^2}{2} \mathcal{I}_{12} = \frac{d^2}{8m^2} \mathcal{I}_{12}^{(d+2)},\quad (7.126)$$

which after $d \rightarrow d - 2$ yields again Eq. (7.124). Using $\mathcal{I}[S(d; \bar{q}_1, \bar{q}_2)]$ is therefore more convenient in this case, since quadratic terms in μ_{ij} do not appear in the reduction.

Chapter 8

Conclusions

Scattering amplitudes describe the probability of the fundamental interactions between elementary particles involved in physical processes, and they can be regarded as the main point of contact between a theoretical model and the related phenomenology. They are an essential ingredient for obtaining theoretical predictions needed by modern experiments in particle physics, especially the ones at colliders such as LHC. For this reason, the development of general methods for their computation at higher-orders in perturbation theory has become increasingly important in recent years.

In this thesis, we presented several new developments on the topic of the *integrand reduction* of loop amplitudes, focusing on semi-analytic and algebraic techniques. The main goal of this techniques is the computation of loop integrals contributing to scattering amplitudes by reducing the respective integrands to a linear combination of fundamental, irreducible contributions. We dealt with the improvement and the extension of one-loop integrand reduction techniques, providing several phenomenological applications, as well as with their generalization to higher loops.

We built a general framework for the integrand decomposition of Feynman diagrams which is based on simple concepts of algebraic geometry and can be applied at all orders in perturbation theory. We derived a recursive formula which allows to find the integrand decomposition of any Feynman integral, by iterating the application of the *multivariate polynomial division* algorithm to integrands with a lower and lower number of loop denominators. This formula correctly reproduces the known one-loop integrand decomposition and provides a recipe for extending the result to any higher-loop topology. It also allows to extend one- and higher-loop results to theories allowing higher-rank integrands, such as non-renormalizable and effective theories. We showed how the *fit-on-the-cut* approach for the reduction, which consists in performing the reduction by cutting – i.e. putting on-shell – loop propagators, can also be systematically extended to higher-loop amplitudes, assessing its advantages and limits. We also developed an alternative and purely algebraic reduction technique, called *divide-and-conquer*, which consists in the recursive application of the polynomial division to

the integrand of the diagram and its sub-topologies. This method, being based on the same concepts used to prove the generalization of the integrand decomposition to all loop orders, can be applied to any integrand in any theory and it always yields the integrand decomposition in a finite number of well-defined algebraic operations.

In the one-loop case, we developed a novel method for the integrand reduction, based on the systematic application of the *Laurent series expansion* to the integrand. The algorithm has been implemented in a C++ library called NINJA, which we made publicly available. The library has been interfaced with the one-loop package GOSAM and has been used for the computation of several complex one-loop amplitudes with up to eight external legs, and for producing phenomenological results, such as NLO corrections to Higgs boson production in association with a top quark pair and a jet, and more recently for new analysis on Higgs production plus two and three jets in gluon fusion. We expect that NINJA, interfaced with either GOSAM or other one-loop packages, will be useful for future phenomenological computations at one-loop involving complex processes.

In the higher-loop case, we showed examples of the application of both the fit-on-the-cut and the divide-and-conquer techniques, discussing also a preliminary semi-automated implementation of the latter. We also showed how to use the integrand reduction combined with identities on dimensionally shifted integrals in order to find additional relations among integrals which are independent at the integrand level, such as identities traditionally found via Integration-by-Parts or Passarino-Veltman methods.

The one-loop techniques proposed in this thesis have already been, and are likely to be in the future, very useful for phenomenological studies in particle physics. Our study on the possibility to generalize integrand reduction techniques at higher-loops produced very promising results which could allow in the future to extend the possibilities of making phenomenological predictions with high-precision for more complex processes.

Appendix A

Spinor-helicity formalism

In this appendix we give a brief description of the *spinor-helicity formalism* which has been used in several points of this thesis. This is particularly useful, among many other things, for the construction of four-dimensional bases of momenta or polarization vectors. Here we will focus on massless spinors. Further details on the topic can be found in pedagogical reviews such as Ref.s [118,119] and references therein. A simple numerical library for massless spinors is distributed with the library NINJA (see e.g. the example described in Section C.5.2).

The Dirac equation for a massless spinor u with momentum k is

$$\not{k} u(k) = 0. \quad (\text{A.1})$$

This equation has two solutions u_- and u_+ , for left- and right-handed fermions respectively. These satisfy

$$P_{\pm} u_{\pm}(k) \equiv \frac{1}{2} (1 \pm \gamma_5) u_{\pm}(k) = u_{\pm}(k) \quad (\text{A.2})$$

The spinors v for anti-particles, in the massless case, obey the same equation. One can thus choose them to be equal to the ones for particles, namely $v_{\pm}(k) = u_{\mp}(k)$.

From these one defines the square and angle brackets

$$\begin{aligned} |k\rangle &\equiv u_+(k) = v_-(k), & [k] &\equiv u_-(k) = v_+(k), \\ \langle k| &\equiv \bar{u}_-(k) = \bar{v}_+(k), & [k| &\equiv \bar{u}_+(k) = \bar{v}_-(k). \end{aligned} \quad (\text{A.3})$$

These allow to define, for any set of momenta k_i the *anti-symmetric* spinor products

$$\langle i j \rangle \equiv \langle k_i k_j \rangle, \quad [i j] \equiv [k_i k_j] \quad (\text{A.4})$$

and the Lorentz vectors

$$\frac{\langle i \gamma^{\mu} j \rangle}{2} \equiv \frac{\langle k_i \gamma^{\mu} k_j \rangle}{2}. \quad (\text{A.5})$$

Notice that, as a consequence of the Dirac equation, the vector in Eq. (A.5) is orthogonal to the momenta k_i and k_j which defined them. Eq. (A.5) can also be read as a way to define a

Lorentz vector from its square and angle brackets. As a special case, a massless momentum k can be rewritten as

$$k^\mu = \frac{\langle k \gamma^\mu k \rangle}{2}. \quad (\text{A.6})$$

One can prove the following relations

$$\langle k_i k_j \rangle [k_j k_i] = 2 (k_i \cdot k_j) \quad (\text{A.7})$$

$$\frac{\langle k_i \gamma^\mu k_j \rangle}{2} \frac{\langle k_l \gamma_\mu k_m \rangle}{2} = \frac{1}{2} \langle k_i k_l \rangle [k_m k_j] \quad (\text{A.8})$$

and the so-called *Shouten identity*

$$\begin{aligned} \langle k_i k_j \rangle \langle k_l k_m \rangle + \langle k_i k_l \rangle \langle k_m k_j \rangle + \langle k_i k_m \rangle \langle k_j k_l \rangle &= 0 \\ [k_i k_j] [k_l k_m] + [k_i k_l] [k_m k_j] + [k_i k_m] [k_j k_l] &= 0. \end{aligned} \quad (\text{A.9})$$

One can also define *polarization vectors* for right-handed (R) and left-handed (L) massless gauge bosons as

$$\epsilon_R^\mu(k, r) = \frac{1}{\sqrt{2}} \frac{\langle r \gamma^\mu k \rangle}{\langle r k \rangle}, \quad \epsilon_L^\mu(k, r) = \frac{1}{\sqrt{2}} \frac{\langle k \gamma^\mu r \rangle}{[k r]}, \quad (\text{A.10})$$

where k is the momentum of the *incoming* boson and r is an arbitrary reference vector. For *outgoing* particles, these formulas still hold but the helicity should be reversed.

If k_1 and k_2 are two independent massless momenta, the following defines a four-dimensional basis \mathcal{E} of massless Lorentz vectors

$$\mathcal{E} = \{e_1, e_2, e_3, e_4\}, \quad e_1^\mu = k_1^\mu, \quad e_2^\mu = k_2^\mu, \quad e_3^\mu = \frac{\langle e_1 \gamma^\mu e_2 \rangle}{2}, \quad e_4^\mu = \frac{\langle e_2 \gamma^\mu e_1 \rangle}{2}, \quad (\text{A.11})$$

which satisfies the following normalization relations

$$e_i^2 = e_1 \cdot e_3 = e_1 \cdot e_4 = e_2 \cdot e_3 = e_2 \cdot e_4 = 0, \quad (e_3 \cdot e_4) = -(e_1 \cdot e_2), \quad (\text{A.12})$$

where the last identity follows from Eq. (A.7) and (A.8). Any four-dimensional (real or complex) vector k can be decomposed in terms of this basis as

$$k^\mu = x_1 e_1^\mu + x_2 e_2^\mu + x_3 e_3^\mu + x_4 e_4^\mu, \quad (\text{A.13})$$

where

$$x_1 = \frac{k \cdot e_2}{e_1 \cdot e_2}, \quad x_2 = \frac{k \cdot e_1}{e_1 \cdot e_2}, \quad x_3 = \frac{k \cdot e_4}{e_3 \cdot e_4}, \quad x_4 = \frac{k \cdot e_3}{e_3 \cdot e_4}. \quad (\text{A.14})$$

Moreover, if k is massless one can define its angle and square brackets from the ones of the vectors e_1 and e_2 . If, for instance, $x_3 \neq 0$, we can define

$$|k\rangle = x_3 |e_1\rangle + x_2 |e_2\rangle, \quad |k] = \frac{x_1}{x_3} |e_1] + |e_2]. \quad (\text{A.15})$$

If, instead, $x_4 \neq 0$, we can use

$$|k\rangle = x_1 |e_1\rangle + x_4 |e_2\rangle, \quad |k] = |e_1] + \frac{x_2}{x_4} |e_2]. \quad (\text{A.16})$$

If $x_3 = x_4 = 0$, the massless condition implies that either $k = x_1 e_1$ or $k = x_2 e_2$ and one can simply choose

$$|k\rangle = x_1 |e_1\rangle, \quad |k] = |e_1] \quad \text{or} \quad |k\rangle = x_2 |e_2\rangle, \quad |k] = |e_2] \quad (\text{A.17})$$

respectively. In all these cases, one can check that Eq. (A.6) reproduces the correct expression for k^μ (notice that the massless condition $k^2 = 0$ implies $x_1 x_2 = x_3 x_4$).

For numerical evaluations, one can use an explicit representation of the Dirac matrices, e.g.

$$\gamma^\mu = \begin{pmatrix} 0 & \sigma^\mu \\ \bar{\sigma}^\mu & 0 \end{pmatrix}, \quad (\text{A.18})$$

with

$$(\sigma^\mu) = (\mathbf{1}_{2 \times 2}, \sigma_1, \sigma_2, \sigma_3), \quad (\bar{\sigma}^\mu) = (\mathbf{1}_{2 \times 2}, -\sigma_1, -\sigma_2, -\sigma_3), \quad (\text{A.19})$$

where σ_i are the Pauli matrices. In this representation, the matrix γ_5 reads

$$\gamma_5 = \begin{pmatrix} -\mathbf{1}_{2 \times 2} & 0 \\ 0 & \mathbf{1}_{2 \times 2} \end{pmatrix}. \quad (\text{A.20})$$

One can choose

$$|k\rangle = u_+(k) = v_-(k) = \begin{pmatrix} 0 \\ 0 \\ k^+ \\ k^- e^{+i\phi_k} \end{pmatrix}, \quad |k] = u_-(k) = v_+(k) = \begin{pmatrix} k^- e^{-i\phi_k} \\ -k^+ \\ 0 \\ 0 \end{pmatrix}, \quad (\text{A.21})$$

where

$$k_\pm = \sqrt{k^0 \pm k^3}, \quad k^- e^{\pm i\phi_k} = \frac{k^1 \pm i k^2}{k^+}.$$

These formulas are strictly valid for positive-energy and real momenta. Using this assumption, one can work out the analytic expressions for spinor products, as well as the ones for the orthogonal momenta defined by Eq.(A.5) and the polarization vectors of Eq. (A.10). The results can thus be extended by analytic continuation to negative-energy or complex momenta. As we stated, numeric formulas for spinors, spinor products, orthogonal momenta and polarization vectors have been implemented in the NINJA library. They are used internally for the computation of four-dimensional bases, but they can also be useful for the numerical definition of spinors and polarization vectors appearing in the expression of scattering amplitudes.

Appendix B

Basic concepts of algebraic geometry

In this appendix we collect some basic concepts of algebraic geometry we used in Chapters 3 and 6. We make no attempt to be completely general or mathematically rigorous, but we limit ourselves to give a brief review of some notions needed for understanding the contents of this thesis. A more comprehensive treatment of these subjects can be found on several textbooks, such as Ref.s [200, 201].

B.1 Polynomial ideals

With $P[\mathbf{z}] = P[z_1, \dots, z_n]$ we denote the *ring* of all polynomials in the variables $\mathbf{z} \equiv (z_i)$ over a properly chosen field. In this thesis, the latter is typically assumed to be the field of complex numbers or, in algebraic computations, the field of rational functions of a set of kinematic invariants.

For any set of polynomials $\{p_1, \dots, p_m\}$ in the variables \mathbf{z} , one can define the *ideal* \mathcal{J} generated by $\{p_1, \dots, p_m\}$ as

$$\mathcal{J} \equiv \langle p_1, \dots, p_m \rangle = \left\{ \sum_{k=1}^m h_k(\mathbf{z}) p_k(\mathbf{z}) : h_k(\mathbf{z}) \in P[\mathbf{z}] \right\}, \quad (\text{B.1})$$

i.e. as the set of polynomials which can be written as a combination of the generators p_k .

One can observe that all the elements of \mathcal{J} vanish on the solutions of the system of equations defined by its generators p_k , namely $p_1(\mathbf{z}) = \dots = p_m(\mathbf{z}) = 0$. On the other hand, if a polynomial vanishes on all solutions of this system of equations, it does *not* necessarily belong to the ideal \mathcal{J} . More formally, for every ideal \mathcal{J} one can define the *algebraic variety* $\mathcal{V}(\mathcal{J})$

$$\mathcal{V}(\mathcal{J}) = \{ \mathbf{z} : p(\mathbf{z}) = 0 \forall p \in \mathcal{J} \}. \quad (\text{B.2})$$

One can prove that the set of polynomials which identically vanish on an algebraic variety is an ideal, which we denote as $\mathcal{I}(\mathcal{V})$,

$$\mathcal{I}(\mathcal{V}) = \{p \in P[\mathbf{z}] : p(\mathbf{z}) = 0 \forall \mathbf{z} \in \mathcal{V}\}. \quad (\text{B.3})$$

It is straightforward to see that, for any ideal \mathcal{J} ,

$$\mathcal{J} \subseteq \mathcal{I}(\mathcal{V}(\mathcal{J})), \quad (\text{B.4})$$

while the equality between the two sides of the previous equation is in general not true.

The *radical* of an ideal \mathcal{J} is denoted as $\sqrt{\mathcal{J}}$ and defined as

$$\sqrt{\mathcal{J}} = \{p \in P[\mathbf{z}] : p^k \in \mathcal{J} \text{ for some integer } k > 0\} \quad (\text{B.5})$$

An ideal \mathcal{J} is said to be radical if $\sqrt{\mathcal{J}} = \mathcal{J}$. One can prove the following

Theorem. Hilbert's Nullstellensatz. *Let \mathcal{J} be a polynomial ideal of the ring $P[\mathbf{z}]$. If p is a polynomial in $P[\mathbf{z}]$ such that $p(\mathbf{z}) = 0$ for all \mathbf{z} in the variety $\mathcal{V}(\mathcal{J})$, then $p^k \in \mathcal{J}$ for some positive integer k . In other words*

$$\mathcal{I}(\mathcal{V}(\mathcal{J})) = \sqrt{\mathcal{J}}.$$

For the special case where \mathcal{J} is a radical ideal, we have the following

Corollary. *Let \mathcal{J} be a radical ideal (i.e. $\sqrt{\mathcal{J}} = \mathcal{J}$) of $P[\mathbf{z}]$. If $p \in P[\mathbf{z}]$ is a polynomial such that $p(\mathbf{z}) = 0$ for all $\mathbf{z} \in \mathcal{V}(\mathcal{J})$, then $p \in \mathcal{J}$.*

Another special case is the one where $\mathcal{V}(\mathcal{J}) = \emptyset$, i.e. when the system of equations $p_1(\mathbf{z}) = \dots = p_m(\mathbf{z}) = 0$ defined by the ideal \mathcal{J} has no solution. In that case, one can prove a *weak* version of Hilbert's Nullstellensatz, namely

Corollary. Hilbert's weak Nullstellensatz. *The system of polynomial equations defined by the generators of an ideal \mathcal{J} is impossible if and only if $1 \in \mathcal{J}$. In other words,*

$$\mathcal{V}(\mathcal{J}) = \emptyset \quad \Leftrightarrow \quad \exists h_1, \dots, h_m \in P[\mathbf{z}] : 1 = \sum_{k=1}^m h_k(\mathbf{z})p_k(\mathbf{z}).$$

It is worth observing that if the conditions of Hilbert's weak Nullstellensatz are satisfied, then $\mathcal{J} = P[\mathbf{z}]$, i.e. the ideal coincides with the whole ring and any polynomial can be written as a combination of the generators of \mathcal{J} .

B.2 Polynomial division, Gröbner bases and quotient rings

Given a ring of polynomials $P[\mathbf{z}]$, a *monomial order* (or *monomial ordering*) is a total order in the set of the monomials of the ring, which satisfies

- if $m_1 < m_2$ then $m_1 m_3 < m_2 m_3$, for m_1, m_2, m_3 monomials in $P[\mathbf{z}]$,
- for any monomial m_1 , if $m_1 \neq 1$ then $m_1 > 1$.

In the univariate case, the only monomial order satisfying these conditions is $1 < z < z^2 < z^3 < \dots$, but in the multivariate case several orderings can be defined. For the purposes of this thesis, we only need to mention two of them, namely the *lexicographical* and the *degree reverse lexicographical* orderings. Given an ordered set of coordinates $z_1 > z_2 > \dots > z_n$ and two monomials $m_1 = z_1^{\alpha_1} \dots z_n^{\alpha_n}$ and $m_2 = z_1^{\beta_1} \dots z_n^{\beta_n}$, we build the vectors of exponents $\vec{\alpha} = (\alpha_1, \dots, \alpha_n)$ and $\vec{\beta} = (\beta_1, \dots, \beta_n)$. Then we can define the two monomial orders as

lexicographical order $m_1 > m_2$ if the leftmost nonzero entry of the vector $\vec{\alpha} - \vec{\beta}$ is positive,

degree reverse lexicographical order $m_1 > m_2$ if $\sum_k \alpha_k > \sum_k \beta_k$, or if $\sum_k \alpha_k = \sum_k \beta_k$ and the rightmost non nonzero entry of the vector $\vec{\alpha} - \vec{\beta}$ is negative.

Unless specified otherwise, in the computations presented in this thesis we use the *degree reverse lexicographical order*. This has the advantages of being more practical and computationally more efficient. Moreover, when this ordering is used, a reduction via polynomial division (see below) will either lower or leave invariant the total degree of the polynomials.

Given a set of polynomials p_1, \dots, p_m in the variables \mathbf{z} and a monomial order, the *polynomial-division algorithm* allows to decompose any polynomial $f \in P[\mathbf{z}]$ as

$$f(\mathbf{z}) = \mathcal{Q}(\mathbf{z}) + \mathcal{R}(\mathbf{z}), \quad \mathcal{Q}(\mathbf{z}) = \sum_{k=1}^m q_k(\mathbf{z}) p_k(\mathbf{z}), \quad (\text{B.6})$$

where the polynomials \mathcal{Q} and \mathcal{R} are the *quotient* and *remainder* of the division respectively. In this case we say that f is *reduced* to \mathcal{R} modulo the polynomials p_k , or

$$f(\mathbf{z}) = \mathcal{R}(\mathbf{z}) \pmod{\{p_k\}}. \quad (\text{B.7})$$

The main downside of such a decomposition is that it is generally not unique. In particular, \mathcal{R} and \mathcal{Q} can depend on the order in which the polynomials p_k are taken, and \mathcal{R} could be different from zero even if f belongs to the ideal $\mathcal{J} = \langle p_1, \dots, p_m \rangle$. These issues are however not present when *Gröbner bases* are used.

Let \mathcal{J} be a polynomial ideal such as the one defined in Eq. (B.1). A *Gröbner basis* is a set of polynomials $\mathcal{G}_{\mathcal{J}} = \{g_1, \dots, g_l\}$ which generates \mathcal{J} , i.e.

$$\langle g_1, \dots, g_l \rangle = \langle p_1, \dots, p_m \rangle = \mathcal{J}, \quad (\text{B.8})$$

such that the remainder of a polynomial division modulo the g_k is always unique, once the monomial order has been fixed. More in detail, the remainder $\mathcal{R}_{\mathcal{G}_{\mathcal{J}}}$ of a polynomial division modulo $\mathcal{G}_{\mathcal{J}}$ only depends on the ideal \mathcal{J} and the dividend f . In particular, $\mathcal{R}_{\mathcal{G}_{\mathcal{J}}}(\mathbf{z}) = 0$ if

and only if $f \in \mathcal{J}$. One can show that every ideal has a Gröbner basis¹. Notice that in general $l \neq m$, i.e. the number of polynomials in a Gröbner basis can differ from the one of the original generators which defined the ideal.

The properties of Gröbner bases allow us to define the *reduction modulo a polynomial ideal*. Given a polynomial ring $P[\mathbf{z}]$, an ideal \mathcal{J} and a Gröbner basis $\mathcal{G}_{\mathcal{J}}$ of this ideal, a polynomial $f(\mathbf{z})$ is said to be reduced to $\mathcal{R}(\mathbf{z})$ modulo \mathcal{J} , if \mathcal{R} is the remainder of a polynomial division modulo a Gröbner basis of \mathcal{J} . Since for Gröbner bases the remainder of a polynomial division is unique, this definition is unambiguous. We use the notation $\lfloor f \rfloor_{\mathcal{J}}$ to denote the reduced polynomial \mathcal{R} , i.e.

$$\lfloor f \rfloor_{\mathcal{J}} = f \pmod{\mathcal{G}_{\mathcal{J}}}. \quad (\text{B.9})$$

In this case we say that $\lfloor f \rfloor_{\mathcal{J}}$ is the *normal form* of f with respect to the ideal \mathcal{J} .

One can also define, for any ideal \mathcal{J} , an equivalence relation $\sim_{\mathcal{J}}$ in $P[\mathbf{z}]$ as

$$p \sim_{\mathcal{J}} q \quad \text{iff} \quad p - q \in \mathcal{J}, \quad (\text{B.10})$$

i.e. two polynomials are equivalent if their difference belongs to the ideal \mathcal{J} . This is the case when the difference $p - q$ is reduced to zero modulo (the Gröbner basis $\mathcal{G}_{\mathcal{J}}$ of) \mathcal{J} . The equivalence classes $\lfloor p \rfloor_{\mathcal{J}}$ contain polynomials whose difference is an element of \mathcal{J} . One can easily show that the set of all equivalence classes defined by such equivalence relation is a polynomial ring, called *quotient ring* and denoted by $P[\mathbf{z}]/\mathcal{J}$. More explicitly

$$P[\mathbf{z}]/\mathcal{J} \equiv \{\lfloor p \rfloor_{\mathcal{J}} : p \in P[\mathbf{z}]\}. \quad (\text{B.11})$$

Algebraic operations between equivalence classes can be defined from algebraic operations in the original ring $P[\mathbf{z}]$ by choosing a representative for each class. One can check that these definitions are well formed, being independent of the choice of representative in each equivalence class.

In general, one can identify in a natural way any equivalence class in $P[\mathbf{z}]/\mathcal{J}$ with one of its representatives in $P[\mathbf{z}]$, thanks to the properties of Gröbner bases, namely with the assignment

$$\lfloor p \rfloor_{\mathcal{J}} \leftrightarrow p. \quad (\text{B.12})$$

More verbosely, each equivalence class is identified with the remainder of a representative p modulo a Gröbner basis of the ideal \mathcal{J} . Since $\lfloor p \rfloor_{\mathcal{J}} = 0$ if and only if $p \in \mathcal{J}$ (or equivalently $\lfloor p \rfloor_{\mathcal{J}} = \lfloor q \rfloor_{\mathcal{J}}$ if and only if $p - q \in \mathcal{J}$), this definition is independent of the choice of the

¹With the definition we gave, a *Gröbner basis* for an ideal is not unique. Indeed, one could easily check that if $\{g_k\}$ is a Gröbner basis, then $\{\lambda g_k\}$ is one as well, for any constant $\lambda \neq 0$. One can introduce the concept of *reduced Gröbner basis* and show that every ideal has a unique reduced Gröbner basis. However we will omit this definition, since it is not needed for understanding the contents of this thesis, although the algorithms implemented in computer algebra systems generally return *reduced* Gröbner bases, and in the literature one often understands Gröbner bases to be reduced and therefore unique.

representative p in $[p]$. Hence, we can think of any element of $P[\mathbf{z}]/\mathcal{J}$ as an element of $P[\mathbf{z}]$ by performing a reduction modulo \mathcal{J} on one of its representatives. Gröbner bases thus allow to perform algebraic operations in quotient rings following the same principles of modular arithmetic. Indeed, one can perform arithmetic operations in \mathbb{Z}/n by mapping them into operations in \mathbb{Z} combined with the operation “mod n ”. Likewise, one can perform algebraic operations in quotient rings $P[\mathbf{z}]/\mathcal{J}$ by mapping them to operations in $P[\mathbf{z}]$ on their representatives and combining them with the operation “mod $\mathcal{G}_{\mathcal{J}}$ ”.

B.3 Zero-dimensional ideals

In this section we collect some results about *zero-dimensional ideals* that we used to prove the *maximum-cut theorem* in Section 3.2. Here we will assume to work with polynomials over the field of complex numbers.

Before giving the definition of a *zero-dimensional ideal*, we enunciate the following

Theorem. Finiteness. *Let \mathcal{J} be an ideal in $P[\mathbf{z}]$. The following conditions are equivalent*

- $P[\mathbf{z}]/\mathcal{J}$ is a finite-dimensional space,
- $\mathcal{V}(\mathcal{J})$ is a finite set.

If the conditions of the *Finiteness Theorem* are satisfied, the ideal \mathcal{J} is said to be *zero-dimensional*.

A second result we used in Section 3.2, is the following Proposition, which can be found e.g. in Ref. [200] (Corollary 2.6, Chapter 4),

Proposition. *Let \mathcal{J} be a zero-dimensional ideal. The ideal is radical ($\sqrt{\mathcal{J}} = \mathcal{J}$) if and only if every solution in $\mathcal{V}(\mathcal{J})$ has multiplicity 1.*

Finally, we recall the *Shape Lemma*, which is the most important ingredient for the proof of the *maximum-cut theorem* given in Section 3.2.

Theorem. Shape Lemma. *Let \mathcal{J} be a zero-dimensional radical ideal in $P[\mathbf{z}]$ such that the last coordinates z_n of every $\mathbf{z} \in \mathcal{V}(\mathcal{J})$ are distinct from each other. Let $\mathcal{G}_{\mathcal{J}}$ be the Gröbner basis of this ideal with respect to lexicographical order with z_n as last variable. Then, the following conditions are satisfied*

- if $\mathcal{V}(\mathcal{J})$ has n_s points, then the monomials $1, z_n, z_n^2, \dots, z_n^{n_s-1}$ are linearly independent and are a basis of $P[\mathbf{z}]/\mathcal{J}$,

- $\mathcal{G}_{\mathcal{J}}$ consists in n monomials g_1, \dots, g_n of the form

$$\begin{aligned}g_1 &= z_1 + h_1(z_n) \\ &\vdots \\ g_{n-1} &= z_{n-1} + h_{n-1}(z_n) \\ g_n &= z_n^{n_s} + h_n(z_n)\end{aligned}$$

where the maximum degree of the polynomials h_k is n_s .

Appendix C

Usage of the C++ library NINJA

In this appendix we describe the usage of the public C++ library NINJA, which implements the integrand reduction via Laurent expansion method for the computation of one-loop integrals (described in Chapter 4).

C.1 Installation

NINJA can be obtained at the url <http://ninja.hepforge.org>. The library is distributed with its source code using the GNU build system (also known as AUTOTOOLS). It can be compiled and installed with the shell commands

```
./configure
make
make install
```

This will typically install the library and the header files in sub-directories of `/usr/local`. The `--prefix` option can be used in order to specify a different installation path. In this case, you might need to update your `LD_LIBRARY_PATH` (or `DYLD_LIBRARY_PATH` on MAC OS) environment variable accordingly. In order to use NINJA for the production of phenomenological results, one must interface it with a library of Master Integrals. Interfaces to the ONELOOP and LOOPTOOLS libraries are already provided with the distribution (see Section C.7 for interfacing a different library). These can be enabled by passing to the `configure` script the options `--with-avholo[=FLAGS]` and `--with-looptools[=FLAGS]`. For instance, the following commands

```
./configure --prefix=$HOME/ninja \  
  --with-avholo='-L/path/to/avh_olo/lib -lavh_olo' \  
  FCINCLUDE=-I/path/to/avh_olo/include  
make install
```

will install all the files in sub-directories of `$HOME/ninja` and build the interface with the ONELOOP library, which will be supposed to be already installed and linkable with the flags specified with the `--with-avholo` option. We also specified the `FCINCLUDE` variable with the flags which are needed to find FORTRAN-90 modules when they are not installed in a default directory. A full list of optional arguments for the `configure` script can be obtained with the command `./configure --help`. While most of them are common to every package distributed with the GNU build system, some are instead specific to the NINJA library and they are described in Table C.1. In most of the cases, only the options for interfacing the integral libraries should be needed.

The user can also optionally install the PYTHON package NINJANUMGEN, which allows one to easily generate the input needed by NINJA from an analytic expression of the numerator of the integrand. The package can be used both as a script and as a PYTHON module, and it could also be useful for interfacing NINJA to existing one-loop packages. In order to install the package, move in the `utils` folder and type

```
python setup.py install
```

where, as usual, an installation prefix can be specified using `--prefix`. In this case one might need to update the `PATH` and `PYTHONPATH` environment variables accordingly. The package needs FORM-4 [136, 137] in order to compute the expansions which are needed and for the generation of the corresponding source code.

Further information about the installation procedure can be found in the `README` file of the distribution.

Option	Description
<code>--with-avholo [=FLAGS]</code>	Include an interface with the ONELOOP integral library [45, 129], specifying the corresponding flags for <i>dynamic</i> linking. If the FORTRAN module <code>avh_olo</code> is not in a standard path, one should add its directory to the <code>FCINCLUDE</code> variable when using this option.
<code>--with-looptools [=FLAGS]</code>	Include an interface with the LOOPTOOLS library [44] (needs LOOPTOOLS version 2.9 or higher), specifying the corresponding flags for <i>static</i> linking. If the header file <code>clooptools.h</code> is not in a standard path, one should add its directory to the <code>CPPFLAGS</code> variable when using this option.
<code>--with-quadruple [=FLAGS]</code>	Compile the library in quadruple precision. This requires the GCC <code>LIBQUADMATH</code> library and one can specify the corresponding flags for the linker (<code>-lquadmth</code> by default). With this option, the types <code>ninja::Real</code> and <code>ninja::Complex</code> will be quadruple precision floating point numbers, and including any public header file of the library will define the macro <code>NINJA_QUADRUPLE</code> to 1 (it will not be defined otherwise). The user should also make sure, when using this option, that the libraries of Master Integrals used by NINJA are compiled in quadruple precision, with floating point types compatible with the ones of GCC.
<code>--enable-higher_rank</code>	Enable support for higher-rank numerators. This is not needed for renormalizable theories.
<code>--disable-gosam</code>	Do not include GOSAM interface.
<code>--disable-avholo_cache</code>	Do not include a cache of Master Integrals in the interface with the ONELOOP library.

Table C.1: Options and environment flags for the `configure` script. Only the options which modify the default behaviour of NINJA are listed.

C.2 Basic types and namespaces

All the types, classes and functions provided by the NINJA library are defined inside the `ninja` namespace. In particular, the types `Real` and `Complex` are aliases for `double` and `std::complex<double>`, unless the library was compiled in quadruple precision. Classes for real and complex momenta are defined as `RealMomentum` and `ComplexMomentum` respectively. They are wrappers of four-dimensional arrays of the corresponding floating-point types, which overload arithmetic and subscript operators. More in detail, an instance `p` of one of these classes represents a momentum p according to the representation

$$\mathbf{p} = \{p[0], p[1], p[2], p[3]\} = \{E_p, x_p, y_p, z_p\},$$

i.e. with the energy in the zeroth component, followed by the spatial components.

C.3 Writing the integrand

The inputs needed by the reduction algorithm implemented in NINJA are the momenta p_i and the masses m_i of the loop denominators defined in Eq. (3.14), and the numerator $\mathcal{N}(q, \mu^2)$ of the integrand. The latter must be cast in four different forms (one of which is optional). The C++ implementation requires the numerator to be an instance of a class inherited from the abstract class `ninja::Numerator`. The latter is defined as

```
class Numerator {
public:
    virtual Complex evaluate(const ninja::ComplexMomentum & q,
                           const ninja::Complex & mu2,
                           int cut,
                           const ninja::PartitionInt partition[])
        = 0;

    virtual void muExpansion(const ninja::ComplexMomentum v[],
                            const ninja::PartitionInt partition[],
                            ninja::Complex c[]) {}

    virtual void t3Expansion(const ninja::ComplexMomentum & v0,
                            const ninja::ComplexMomentum & v3,
                            const ninja::ComplexMomentum & v4,
                            const ninja::Complex & beta,
                            int mindeg,
                            int cut,
                            const ninja::PartitionInt partition[],
```

```

        ninja::Complex c[]) = 0;

    virtual void t2Expansion(const ninja::ComplexMomentum & v1,
                           const ninja::ComplexMomentum & v2,
                           const ninja::ComplexMomentum & v3,
                           const ninja::ComplexMomentum & v4,
                           const ninja::Complex beta[],
                           int mindeg,
                           int cut,
                           const ninja::PartitionInt partition[],
                           ninja::Complex c[]) = 0;

    virtual ~Numerator() {}
};

```

The input parameters `cut` and `partition` are common to more methods and give information about the multiple cut where NINJA is currently evaluating the numerator. Although this information is not always necessary, there might be occasions where it could be useful for an efficient evaluation of the numerator. The integer `cut` is equal to k if the numerator is being evaluated on a k -ple cut, with $k = 1, 2, 3, 4$. This parameter is not given in the method `muExpansion` because the latter is always evaluated on quadruple cuts (i.e. `cut = 4`). The parameter `partition` points to an array of integers (namely of integer type `ninja::PartitionInt`), with length equal to `cut`, containing the indexes of the cut numerators. If the user asks to perform a global test (see Section C.4), the numerator will also be evaluated outside the solutions of the multiple cuts, in which case the parameter `cut` will be set to zero. As an example, if the method `t3Expansion` is evaluated on the 3-ple cut $D_0 = D_2 = D_5 = 0$ for the determination of the 3-point residue Δ_{025} , we will have `cut = 3`, `partition[0] = 0`, `partition[1] = 2`, and `partition[2] = 5`. The returned expansion only needs to be valid for the cut specified (e.g. $D_0 = D_2 = D_5 = 0$ in the previous example).

In Section C.3.1 we give a detailed description of each method of the class `ninja::Numerator`, for a generic numerator of an n -point integrand of rank r . If the analytic expression of the integrand is available, all these methods can be more easily generated with the help of the simple PYTHON package NINJANUMGEN, which is distributed with the library and whose usage is described in Section C.3.2 and C.6.

C.3.1 Definition of the input

In this subsection we give a detailed description of each method of the class `ninja::Numerator`, for a generic numerator of an n -point integrand of rank r . As already mentioned, this is only needed to those who prefer to provide an alternative implementation of the required methods without using the script NINJANUMGEN, which otherwise can automatically generate the

input for NINJA.

The method `Numerator::evaluate(q, μ2, cut, partition)`

It must return the value of the numerator $\mathcal{N}(q, \mu^2)$ evaluated at the (complex) values of q and μ^2 given as input.

The method `Numerator::muExpansion(v, partition, c)`

This is used for the computation of the Laurent expansion in μ^2 required to obtain the coefficient c_4 of the boxes. In the renormalizable case, this method should compute the leading term of a parametric expansion in t of the integrand defined by

$$q^\nu \rightarrow t v_\perp^\nu, \quad \mu^2 \rightarrow t^2 v_\perp^2 \quad (\text{C.1})$$

where v_\perp is given by $v[0]$, i.e. by the zeroth entry of the array of momenta v . For renormalizable theories this array will therefore only contain at most one element. The method should write the leading coefficient of the expansion in the zeroth entry of the array pointed by the parameter c , i.e.

$$c[0] = \mathcal{N}[t^r];$$

In the higher-rank case $r = n + 1$, which might appear in non-renormalizable and effective theories, this method should instead compute both the leading and the next-to-leading terms of the expansion in t of the numerator, defined by

$$q^\nu \rightarrow t v_0^\nu t + v_1^\nu, \quad \mu^2 \rightarrow t^2 v_0^2 \quad (\text{C.2})$$

with $v_i \equiv v[i]$, where v is the array of momenta passed as input parameter. The two leading terms of the expansion should be written in the first two entries of the array pointed by the parameter c , ordered by decreasing powers of t , i.e.

$$\begin{aligned} c[0] &= \mathcal{N}[t^r]; \\ c[1] &= \mathcal{N}[t^{r-1}]; \end{aligned}$$

The implementation of this method is only required when $r \geq n$. It is also not needed if the user chooses to disable the μ^2 -expansion method for the boxes, but in that case more evaluations of the numerator will be needed and the computation of the pentagons will not be skipped.

The method `Numerator::t3Expansion(v0, v3, v4, β, mindeg, partition, c)`

This method is used for the computation of the coefficients of the residues of both the triangles and the tadpoles. It is supposed to compute the coefficients of the terms $t^j \mu^{2k}$ for $j \in$

$\{r, r-1, r-2, \dots, r-\text{mindeg}\}$, given by substituting into the numerator the parametric expansion of the loop momentum defined by

$$q^\nu \rightarrow v_0^\nu + t v_3^\nu + \frac{\beta + \mu^2}{t} v_4^\nu, \quad v_3^2 = v_4^2 = 0, \quad (v_3 \cdot v_4) = \frac{1}{2}, \quad (\text{C.3})$$

as a function of the momenta v_i^ν and the constant β which are passed as parameters. The maximum value of the parameter `mindeg` is $r-n+3$. Since in a renormalizable theory $r \leq n$, and by definition of rank we have $j+2k \leq r$, in this case at most 6 terms can be non-vanishing in the specified range of j . Similarly, one can check that in the higher-rank case with $r = n+1$ there will be at most 9 terms. The method should write these terms in the entries of the array pointed by `c`, ordered by decreasing powers of t . Terms with the same power of t should be ordered by increasing powers of μ^2 . A pseudo-implementation will therefore look like

```
int idx = 0;
for (int j=r; j>=r-mindeg; --j)
  for (int k=0; 2*k<=r-j; ++k)
    c[idx++] =  $\mathcal{N}[t^j \mu^{2k}]$ ;
```

The method `Numerator::t2Expansion($v_1, v_2, v_3, v_4, \beta, \text{mindeg}, \text{partition}, c$)`

In the current version of NINJA, this method is called during the computation of the coefficients of the bubbles. It is supposed to compute the coefficients of the terms $t^j x^l \mu^{2k}$ for $j \in \{r, r-1, \dots, r-\text{mindeg}\}$, given by the expansion

$$q^\nu \rightarrow v_1^\nu + x v_2^\nu + t v_3^\nu + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{t} v_4^\nu, \\ v_2 \cdot v_3 = v_2 \cdot v_4 = v_3^2 = v_4^2 = 0, \quad (v_3 \cdot v_4) = \frac{1}{2} \quad (\text{C.4})$$

as a function of the momenta v_i^ν and the constants $\beta_i \equiv \beta[\text{i}]$, which are passed as parameters to the method. The maximum value of `mindeg` is $r-n+2$. In a renormalizable theory, this implies that one can have at most 7 non-vanishing terms in this range of j . In the higher-rank case with $r = n+1$ we can have instead up to 13 non-vanishing terms in that range. It is worth observing that the expansion in Eq. (C.4) can be obtained from the previous one in Eq. (C.3) by means of the substitutions

$$v_0^\nu \rightarrow v_1^\nu + x v_2^\nu, \quad \beta \rightarrow \beta_0 + \beta_1 x + \beta_2 x^2, \quad v_2 \cdot v_3 = v_2 \cdot v_4 = 0.$$

The terms of the expansion must be stored in the entries of the array pointed by `c`, ordered by decreasing powers of t . Terms with the same power of t should be ordered from the lowest to the highest with respect to the lexicographical order in the variables (x, μ^2) . A pseudo-implementation will have the form

```

int idx = 0;
for (int j=r; j>=r-mindeg; --j)
  for (int l=0; l<=r-j; ++l)
    for (int k=0; 2*k<=r-j-l; ++k)
      c[idx++] =  $\mathcal{N}[t^j x^l \mu^{2k}]$ ;

```

C.3.2 Using NINJANUMGEN

In this subsection we briefly describe, with the help of a simple example, how to generate the numerator methods, with the help of the PYTHON package NINJANUMGEN. NINJANUMGEN can be both used as a command-line script and within PYTHON by importing the `ninjanumgen` module. Here we describe its usage as a script, with a simple example. A more detailed list of options, allowing to fine-tune the output according to the user's needs, as well as the usage of the package as a PYTHON module, are described in Appendix C.6.

Let us define, as an example, the following 4-point one-loop integrand, with kinematics $k_0, k_1 \rightarrow k_2, k_3$

$$\begin{aligned}
\mathcal{I} &= \frac{\mathcal{N}(q, \mu^2)}{D_0 D_1 D_2 D_3} = \frac{(q \cdot v_1)(q \cdot v_2)(q \cdot v_3)(q \cdot v_4) + \mu^4}{D_0 D_1 D_2 D_3} \\
D_0 &= \bar{q}^2 - m_0^2 \\
D_1 &= (\bar{q} + k_0)^2 - m_1^2 \\
D_2 &= (\bar{q} + k_0 + k_1)^2 - m_2^2 \\
D_3 &= (\bar{q} + k_3)^2 - m_3^2.
\end{aligned} \tag{C.5}$$

where v_i are arbitrary reference vectors. In order to generate the methods declared in the `ninja::Numerator` class, we first create a file `mynum.frm` containing a FORM [124] expression for the numerator

```

* mynum.frm
V v1, v2, v3, v4;
V Q;
S Mu2;

L Diagram = (Q.v1)*(Q.v2)*(Q.v3)*(Q.v4) + Mu2^2;

```

and we run the script with the command

```

ninjanumgen mynum.frm --nlegs 4 --rank 4 -o mynum.cc

```

which creates the source file `mynum.cc` with the definition of the methods, optimized for fast numerical evaluation using the recent features of FORM-4. The command also creates an

header file `mynum.hh`, unless already present, which is supposed to contain the declaration of the numerator class. The latter will have the following form (where for brevity we replaced the parameters of each method with ellipses),

```
class Diagram : public ninja::Numerator {

public:
    virtual ninja::Complex evaluate(...);
    virtual void muExpansion(...);
    virtual void t3Expansion(...);
    virtual void t2Expansion(...);

public:
    // Add other public methods and data here

private:
    // Add other private methods and data here
};
```

If the numerator expression depends on other momenta or parameters, these should be visible inside the definitions of the methods. In our example, the numerator depends on the reference vectors v_i which appear in its FORM expression. One possibility would be declaring these vectors as global variables, but a better alternative could be defining them as data members of the numerator class. In this example we will declare them as public data members by inserting the following code inside the class definition

```
public:
    ninja::ComplexMomentum v1, v2, v3, v4;
```

which defines the vectors as complex Lorentz momenta. This completes the generation of the input, which will allow NINJA to compute the integral.

C.4 Running the reduction

In this subsection we describe the usage of NINJA for the reduction of a generated integrand, such as the one in the example of the previous subsection. With the help of simple examples, we show how to specify the input and how to control the run-time behavior of the procedures of the library. All the public header files of the library are installed in the sub-directory `ninja` of the `include` path in the installation directory.

C.4.1 A simple example

Here we show the contents of a file `simple_test.cc` which illustrates the basic usage of NINJA for computing the integral defined in Eq. (C.5).

```
// simple_test.cc

#include <iostream>
#include <ninja/ninja.hh>
#include <ninja/rambo.hh>
#include "mynum.hh"
using namespace ninja;

int main()
{
    // External legs of the loop
    const int N_LEGS = 4;

    // Center of mass energy
    const Real ENERGY_CM = 50;

    // Invariant s
    const Real S = ENERGY_CM*ENERGY_CM;

    // Rank of the numerator
    const int RANK = 4;

    // Declare an instance of the numerator
    Diagram num;

    // Assign numerical values to the reference vectors
    num.v1 = ComplexMomentum(1.0,1.1,1.2,1.3);
    num.v2 = ComplexMomentum(1.4,1.5,1.6,1.7);
    num.v3 = ComplexMomentum(1.8,1.9,2.0,2.1);
    num.v4 = ComplexMomentum(2.2,2.3,2.4,2.5);

    // Define external momenta
    RealMomentum k[N_LEGS];

    // Get a random phase-space point
    Rambo phase_space(S,N_LEGS);
    phase_space.getMomenta(k);

    // Define the internal momenta of the loop
    RealMomentum pi[N_LEGS];
    pi[0] = RealMomentum(0,0,0,0);
    pi[1] = k[0];
    pi[2] = k[0]+k[1];
}
```

```

pi[3] = k[3];

// Define the square of the internal masses
Real msq[N_LEGS] = {1.,2.,3.,4.};

// Create an amplitude object
Amplitude<RealMasses> amp(N_LEGS,RANK,pi,msq);

// Evaluate the integral
amp.evaluate(num);

// Print the result
std::cout << amp[0] // or amp.eps0(), finite part
           << amp[1] // or amp.epsm1(), single-pole
           << amp[2] // or amp.epsm2(), double-pole
           << std::endl;

return 0;
}

```

In the example above, after specifying some constants, we declare an instance `num` of the user-defined class `Diagram`, which we constructed in Section C.3.2. Then we give numerical values to the reference vectors v_i appearing in the numerator definition of Eq. (C.5) as well as in the analytic expression given in the corresponding FORM file. This defines our numerator.

Next we randomly generate a phase-space point, by creating a `Rambo` object and calling its method `getMomenta` which fills the array `k` of external momenta. The phase space generation is a translation in C++ of the corresponding GOSAM implementation, which in turn is based on the one of Ref. [143]. It is meant to provide an easy way to generate phase-space points in tests which use the library NINJA. Every call of the method `getMomenta` on the same `Rambo` object randomly generates a different phase-space point. The code above assumes the external legs to be massless. If the external legs were massive, with masses `{MASS_0, MASS_1, MASS_2, MASS_3}`, we should have generated the phase-space point by passing an array of external masses as third argument to the constructor of the `Rambo` object, i.e.

```

Real external_masses[N_LEGS] = {MASS_0, MASS_1,
                               MASS_2, MASS_3};

Rambo phase_space(S, N_LEGS, external_masses);
phase_space.getMomenta(k);

```

The method `getMomenta` can optionally take a second parameter, namely a pointer to a `Real`, into which the weight of the generated phase-space point will be written. In the output, the momenta `k[0]` and `k[1]` are taken as incoming, while all the others are taken as outgoing. In order to get reproducible results, one can set the random seed with the class method `setSeed`,

which takes an integer as input.

The momenta p_i are then defined according to Eq. (C.5). Arithmetic operations between momentum types work as one would expect. After specifying the *square* of the internal masses, we create an `Amplitude<RealMasses>` object `amp`, whose method `evaluate` computes the integral with the numerator specified in its argument. This adds the computed integral to the total result stored internally by `amp`, which can then be accessed either with the methods `eps0`, `epsm1`, `epsm2` or with the subscript operator “`[]`” as the example shows.

C.4.2 The Amplitude class

The `Amplitude` template class is the main class of the NINJA library and its method `evaluate` computes a one-loop integral. The method takes as input an object of a class derived from `ninja::Numerator`, which provides a generic interface to the methods defined by each Laurent expansion. The template parameter of the `Amplitude` class is the type of the internal masses. Allowed types are: `RealMasses`, `ComplexMasses` and `Massless`. The methods needed for the evaluation of the amplitude are instantiated for all these three types in the compiled code of the library.

Instantiation In the example above, we showed how to instantiate an `Amplitude` object passing to its constructor the number of external legs, the rank of the numerator, the momenta p_i and the squared masses m_i^2 of the loop denominators (Eq. (3.14)). There we assumed the internal masses to be real. For the complex-masses case and the massless case, the relevant part of the source would have looked like

```
// Complex masses
Complex msq[N_LEGS] = {...};
Amplitude<ComplexMasses> amp(N_LEGS, RANK, pi, msq);
amp.evaluate(num);
```

and

```
// Massless (msq does not need to be specified here)
Amplitude<Massless> amp(N_LEGS, RANK, pi);
amp.evaluate(num);
```

respectively. The `Amplitude` class also has a default constructor, as well as methods which allow to set or change the kinematics, the (squared) internal masses, the number of legs or the rank of the numerator to be evaluated, as in the following

```
Amplitude<RealMasses> amp;
amp.setN(N_LEGS);
amp.setRank(RANK);
```

```
amp.setKinematics(pi);
amp.setInternalMasses(msq);
```

More in detail, the methods `setKinematics` and `setInternalMasses` set a data member which is a pointer to the array of momenta (`pi`) and internal masses (`msq`) to be used respectively. A call of one of these methods copies the pointer given as input into the corresponding data member. The user should thus make sure that the pointed data will exist in memory until the end of the execution of the `evaluate` method.

Renormalization scale Another important setting is the renormalization scale μ_R^2 to be used. This is equal to 1 by default, and it only affects the computation of the Master Integrals. It can be set as in the following example

```
// takes the square of the scale
amp.setRenormalizationScale(50*50);
```

The S-matrix An optional parameter which can be set by the user is a matrix of kinematic invariants, which we call S-matrix. This is defined in NINJA by

$$s_{ij} = (p_i - p_j)^2 \quad (\text{C.6})$$

where p_i are the momenta appearing in Eq. (3.14). When this is specified by the user, the computation of the Master Integrals might be more accurate. This can be particularly useful in the presence of infrared singularities, which otherwise might not be detected by the integral library in use. The S-matrix can be declared either by an `SMatrix` object or by a simple n^2 -dimensional array, where n is the number of loop denominators. In the simple example given in Section C.4.1, using the definitions of Eq. (C.5) in Eq. (C.6), for massless external momenta k_i we could have specified the following S-matrix

$$(s_{ij}) = \begin{pmatrix} 0 & 0 & 2(k_0 \cdot k_1) & 0 \\ 0 & 0 & 0 & -2(k_0 \cdot k_3) \\ 2(k_0 \cdot k_1) & 0 & 0 & 0 \\ 0 & -2(k_0 \cdot k_3) & 0 & 0 \end{pmatrix} \quad (\text{C.7})$$

either with

```
SMatrix s_mat;
s_mat.allocate(N_LEGS); // allocate the matrix
s_mat.fill(0); // fill the entries with zeros
s_mat(0,2) = s_mat(2,0) = 2*mp(k[0],k[1]);
s_mat(1,3) = s_mat(3,1) = -2*mp(k[0],k[3]);
amp.setSMatrix(s_mat);
```

or with

```
Real s_mat[N_LEGS*N_LEGS] = {0, 0, 2*mp(k[0],k[1]), 0,
                             0, 0, 0, -2*mp(k[0],k[3]),
                             2*mp(k[0],k[1]), 0, 0, 0,
                             0, -2*mp(k[0],k[3]), 0, 0};
amp.setSMatrix(s_mat);
```

We recommend to specify the S-Matrix whenever possible, and in particular when infrared singularities are present. As an alternative to writing it explicitly, one could use the method

```
SMatrix &
SMatrix::fillFromKinematics(const RealMomentum pi[],
                           Real ir_threshold = 0);
```

before each call of `evaluate`. This will automatically compute the matrix from the momenta p_i , but it will set to zero all the matrix elements which are smaller than `ir_threshold`.

It can be worth pointing out that an `Amplitude` object, similarly to the case of the internal masses and the kinematics, only stores as data member a pointer to the data of the S-matrix (`s_mat`) to be used.

Stopping the reduction earlier There are cases where lower-point residues do not contribute to the final result for the integral (see e.g. the example in Section C.5.3). If the user knows that k -point residues with $k < \text{MIN_CUT}$ will not contribute to an amplitude, this information can be passed to NINJA through the `setCutStop` method, as in this example

```
amp.setCutStop(MIN_CUT);
```

which will tell NINJA to stop the reduction right after the evaluation of the residues of k -ple cuts with $k = \text{MIN_CUT}$.

Master Integrals Each instance of an `Amplitude` object can in principle use a different library of Master Integrals. The library to be used can be specified with the method `setIntegralLibrary` as in the following example

```
Amplitude<RealMasses> amp;
amp.setIntegralLibrary(loop_tools);
```

If an integral library is not set explicitly for an amplitude object, the instance will use the one which is the *default* at the time of its creation. The default library will be `ONELOOP` if enabled during configuration, otherwise it will be `LOPTOOLS`. If none of the two is enabled, NINJA can still be used, but a different library of Master Integrals should be specified at run-time (more details about the implementation of the corresponding interface are given

in Appendix C.7). The function `setDefaultIntegralLibrary` can be used to change the default integral library. Assuming both `ONELOOP` and `LOOPTOOLS` have been enabled, the user can change the default as in the following example

```
#include <ninja/ninja.hh>
#include <ninja/avholo.hh>
#include <ninja/looptools.hh>
using namespace ninja;

int main()
{
    setDefaultIntegralLibrary(loop_tools);
    // Amplitude objects defined here will use LoopTools
    setDefaultIntegralLibrary(avh_olo);
    // Amplitude objects defined here will use OneLoop
    return 0;
}
```

Disabling the μ^2 expansion The user can choose to avoid using the μ^2 expansion for the boxes with

```
amp.useMuExpansion(false);
```

In this case the method `muExpansion` of the numerator class does not need to be provided. However, this would increase the number calls of the numerator method `evaluate` and it would also require the computation of pentagons. Given the simplicity of the μ^2 expansion, disabling it is therefore not recommended, unless it is done for debugging purposes.

Evaluation of the integrals As already explained, integrals are computed by calling the method `evaluate`. This has the following prototype

```
template <typename MassType>
int Amplitude<MassType>::evaluate(Numerator & num);
```

and takes as input the numerator of the integrand, which must be an instance of a class inherited from `ninja::Numerator`. It returns an integer value which depends on the results of the internal tests which NINJA can optionally perform during the computation (see Section C.4.3). By default no test is performed and the return value can be safely ignored. In general, the return value will be equal to

`Amplitude<MassType>::TEST_FAILED` if any of the performed tests failed

`Amplitude<MassType>::SUCCESS` otherwise.

Each call of the method `evaluate` adds the computed integral to the total result stored internally by the instance. It can then be accessed either with the methods `eps0`, `epsm1`, `epsm2` or with the subscript operator “`[]`” as we illustrated in the simple test described in this section. The result can be quickly reset to zero, by calling the `reset` method,

```
amp.reset();
```

The finite term is given by the sum of the *cut-constructible part* and the *rational part*, but the two can also be accessed separately with

```
Complex cut_constr_part = amp.getCutConstructiblePart();
Complex rational_part = amp.getRationalPart();
```

C.4.3 Global settings

By default NINJA tries to compute a minimal set of coefficients during the reduction, i.e. those which are needed for the determination of the final integrated result. These are only a subset of the ones which are required for the full reconstruction of the integrand decomposition of Eq. (3.17). Indeed the computation of spurious coefficients is skipped whenever possible, i.e. whenever these do not enter the coefficient-level subtractions needed for lower-point residues, as in the case of spurious coefficients of pentagons, boxes and tadpoles. NINJA also entirely skips the computation of residues whose non-spurious coefficients would multiply scaleless integrals.

For debugging purposes, the user can however ask NINJA to perform some tests on the quality of the reconstruction of the integrand, or print some information about the ongoing computation. These operations might require the computation of a larger set of coefficients.

There are two kinds of tests which NINJA can perform: *global tests* and *local tests*. The global $\mathcal{N} = \mathcal{N}$ test [40, 202] checks that the following equality, which follows from Eq. (3.17), is valid

$$\mathcal{N}(q, \mu^2) = \sum_{k=1}^5 \sum_{\{j_1, \dots, j_k\}} \Delta_{i_1 \dots i_k} \prod_{h \neq i_1, \dots, i_k} D_h. \quad (\text{C.8})$$

Another global test, which can be performed when the rank r of the numerator is equal to the number of external legs n , is the so-called *power-test* introduced in Ref. [43]. This consists in checking that the sum of all the spurious tadpole coefficients is vanishing,

$$\sum_{i_1=0}^{n-1} \sum_{k=0}^4 c_k^{(i_1)} = 0. \quad (\text{C.9})$$

Finally, NINJA can perform local $\mathcal{N} = \mathcal{N}$ tests on quadruple, triple, double and single cuts. These will check the validity of Eq. (C.8) on values of the loop momenta corresponding to a

given k -ple cut, and they can be useful in order to pinpoint the multiple cut where an error or an instability has occurred.

By default NINJA does not perform any internal test. The tests to be performed during the execution of the method `evaluate` can be set using the function

```
void setTest(unsigned flag);
```

where the parameter `flag` can be any of the following:

`Test::NONE` no test is performed

`Test::ALL` all tests are performed

`Test::GLOBAL` global tests are performed

`Test::LOCAL_` k with $k \in \{1, 2, 3, 4\}$, local tests on k -ple cuts are performed

`Test::LOCAL` all local tests are performed

or any combination of these. Different flags can be combined with the bitwise OR operator “|”. For instance, the following command will ask NINJA to perform global tests and local tests on double cuts

```
setTest(Test::GLOBAL | Test::LOCAL_2);
```

The $\mathcal{N} = \mathcal{N}$ tests will check whether the numerator \mathcal{N}_{rec} reconstructed by evaluating the r.h.s. of Eq. (C.8) is equal to the numerator \mathcal{N}_{eva} , obtained by a direct evaluation through the `evaluate` method of the numerator class, up to a given tolerance. More explicitly, it checks if

$$\left| \frac{\mathcal{N}_{rec} - \mathcal{N}_{eva}}{\mathcal{N}_{eva}} \right| < \delta_{tol} \quad (\text{C.10})$$

where the threshold δ_{tol} is 10^{-5} by default, but it can be specified by the user through the function

```
void setTestTolerance(Real test_tolerance);
```

As explained in Section C.4.2, the return value of the method `evaluate` of the `Amplitude` class can be used in order to check whether any performed test has failed. These tests have been implemented for debugging purposes, and they are not meant as an estimate of the accuracy of the total result. Indeed, there are cases where a numerical instability might cause a test to fail while having negligible effects on the total amplitude. The accuracy of the result can be better estimated by means of the scaling test proposed in Ref. [51] or the rotation test described in Ref. [99] and Section 5.1.3 of this thesis.

Another global option which can be set is the *verbosity*, i.e. the amount of information printed during the evaluation of an integral. By default nothing is printed during a computation. The setting can be controlled by calling the function

```
void setVerbosity(unsigned flag);
```

where possible values of the parameter `flag` can be

`Verbose::NONE` nothing is printed

`Verbose::ALL` everything is printed (equivalent to the combination of all the other options)

`Verbose::GLOBAL_TEST` the result of global tests are printed, when performed

`Verbose::LOCAL_TEST_k` with $k \in \{1, 2, 3, 4\}$, the result of local tests on k -ple cuts are printed when performed

`Verbose::LOCAL_TESTS` the result of all performed local tests is printed

`Verbose::TESTS` the result of all performed tests is printed

`Verbose::Ck` with $k \in \{1, 2, 3, 4, 5\}$, the value of the coefficients of the computed k -point residues is printed

`Verbose::COEFFS` the value of all the computed coefficients is printed

`Verbose::RESULT` the partial result of every call of the `evaluate` method is printed

`Verbose::INTEGRALS` the value of the Master Integrals is printed.

Similarly to the options controlling the performed tests, any combination of the flags above can be specified using the bitwise OR operator “|”. As an example, the following instruction will ask NINJA to print the value of the triangle coefficients, and the result of the current integral

```
setVerbosity(Verbose::C3 | Verbose::RESULT);
```

When not specified otherwise, NINJA will print everything to standard output. Any other output stream can be set, as in the following example

```
#include <fstream>
#include <ninja/ninja.hh>
using namespace ninja;

int main()
{
    std::ofstream f;
    f.open("my_file.out");
    setOutputStream(f);
}
```

```
// from now on everything will be printed  
// by Ninja on the file "myfile.out"  
  
// ...  
  
return 0;  
}
```

C.4.4 Controlling the settings from GOSAM

As explained in Section 5.1, NINJA is the default reduction library used by the one-loop package GOSAM (since version 2.0 of the package) for the computation of one-loop amplitudes. Most of the settings of NINJA can be controlled from the FORTRAN90 code generated by GOSAM as well. All the routines and the flags defined in the interface between the two, which is included in the NINJA distribution, can be used by importing the FORTRAN90 module `ninjago_module`.

A first option which can be set is the library of Master Integrals to be used by NINJA when called by GOSAM. This can be specified with the call

```
call ninja_set_integral_library(libflag)
```

where `libflag` is an integer flag associated to the library and can be equal to

`NINJA_ONELOOP` for the ONELOOP library

`NINJA_LOOPTOOLS` for the LOOPTOOLS library.

If the ONELOOP library is chosen, NINJA will add to it (if configured with the default options) a cache of integrals, as explained in Section 4.2.2 and C.7.1. The cache of integrals can be cleared with the command

```
call ninja_clear_integral_cache
```

which will reset the cache while keeping some memory allocated for efficient use in later calls (see also Section C.7.1 for more details). If one wishes to completely free all the allocated memory, this can be done with the command

```
call ninja_free_integral_cache
```

although the former should in general be preferred. The command `ninja_clear_integral_cache` also clears the internal cache of the LOOPTOOLS library, when the latter is used. When GOSAM is used as One Loop Provider for Monte Carlo programs, it will automatically call `ninja_clear_integral_cache` once per phase-space point.

As we described in Section C.4.3, for debugging purposes NINJA can optionally perform some internal test or print some information on the ongoing computation. The *local* and *global* tests to be performed (which have been described in Section C.4.3) can be selected with the command

```
call ninja_set_test(val)
```

where the parameter `val` is an integer which can have the following values

NINJA_TEST_NONE no test is performed

NINJA_TEST_ALL all tests are performed

NINJA_TEST_GLOBAL global tests are performed

NINJA_TEST_LOCAL_ k with $k \in \{1, 2, 3, 4\}$, local tests on k -ple cuts are performed

NINJA_TEST_LOCAL all local tests are performed

or any combination of these. Different flags can be combined using the bitwise logical “OR” operation, which in FORTRAN is available as the function `IOR`. For instance, the command

```
call ninja_set_test(IOR(NINJA_TEST_GLOBAL, &
                        & NINJA_TEST_LOCAL_2));
```

asks NINJA to perform global tests and local tests on double cuts. The tolerance of the tests can be set with

```
call ninja_set_test_tolerance(val)
```

where `val` is the maximum relative error.

Another option which can be set is the verbosity of NINJA. By default nothing is printed during a computation, but this setting can be changed with the command

```
call ninja_set_verbosity(verbosity)
```

The information which is asked to be printed with this command is appended to the file `ninja_gosam.out`. Possible values of the parameter `verbosity` are

NINJA_OUTPUT_NONE nothing is printed

NINJA_OUTPUT_ALL everything is printed (equivalent to the combination of all the other options)

NINJA_OUTPUT_GLOBAL_TEST the result of global tests are printed, when performed

NINJA_OUTPUT_LOCAL_TEST_ k with $k \in \{1, 2, 3, 4\}$, the result of local tests on k -ple cuts are printed when performed

`NINJA_OUTPUT_LOCAL_TESTS` the result of all performed local tests is printed

`NINJA_OUTPUT_TESTS` the result of all performed tests is printed

`NINJA_OUTPUT_Ck` with $k \in \{1, 2, 3, 4, 5\}$, the value of the coefficients of the computed k -point residues is printed

`NINJA_OUTPUT_COEFFS` the value of all the computed coefficients is printed

`NINJA_OUTPUT_RESULT` the partial result of every call of the `evaluate` method is printed

`NINJA_OUTPUT_INTEGRALS` the value of the Master Integrals is printed,

where, as in previous case, different flags can be combined with the bitwise “OR” operation. The precision of the printed floating point numbers can be modified with the command

```
call ninja_set_output_precision(val)
```

The other subroutines defined in the module are not supposed to be called directly by the users, but they are called by the code automatically generated by GOSAM during the computation of the amplitudes.

C.5 Built-in examples

In this Section we give a description of the examples which are distributed with the library. In order to compile the corresponding executables, one can use the command

```
make examples
```

either in the root directory or in the `examples` directory. These examples are meant to provide a more detailed description of the usage of the library in several kinds of problems which involve the computation of one-loop integrals. More involved computations performed with the help of NINJA have been presented in Ref.s [99, 100] and in Chapter 5 of this thesis.

Every example presented in this section has been generated with the help of the package NINJANUMGEN, which is distributed with NINJA and is described in Appendix C.6. The package can be used both as a script and as a PYTHON module. For each example, we include in the distribution

- the FORM file (with extension `.frm`) containing the analytic expression of the numerator which is used as input
- a SHELL script (with extension `.sh`) with the command we used for the generation of the numerator class methods
- a PYTHON script (with extension `.py`) which achieves the same by importing the `ninjanumgen` module

- the C++ source files (with extension `.cc`) and headers files (with extension `.hh`) defining the numerator and its methods, as well as a test program.

C.5.1 Simple Test

The first simple example has already been extensively described above, in order to illustrate the basic usage of the library. The numerator class is defined in the header file `mynum.hh`, the source file generated by NINJANUMGEN is `mynum.cc`, while the source file with the `main` function is `simple_test.cc`.

C.5.2 Four-photon helicity amplitudes

In this example we consider a four-photon amplitude [203, 204] and we describe the usage of NINJA for the definition of polarization vectors and other spinor objects which are needed for the evaluation of the numerator.

The integrand of a diagram contributing to a four-photon amplitude is given by

$$\begin{aligned} \mathcal{I} &= \frac{\mathcal{N}(q, \mu^2)}{D_0 D_1 D_2 D_3} \\ \mathcal{N}(q, \mu^2) &= -\text{Tr}\left((\bar{l}_1 + m_f)\epsilon_1(\bar{l}_2 + m_f)\epsilon_2(\bar{l}_3 + m_f)\epsilon_3(\bar{l}_0 + m_f)\epsilon_0\right) \\ D_i &= \bar{l}_i^2 - m_f^2 \end{aligned} \quad (\text{C.11})$$

where m_f is the mass of the fermion propagating in the loop, and the momenta \bar{l}_i are defined by

$$\bar{l}_0 = \bar{q}, \quad \bar{l}_1 = \bar{q} + k_0, \quad \bar{l}_2 = \bar{q} + k_0 + k_1, \quad \bar{l}_3 = \bar{q} - k_4. \quad (\text{C.12})$$

For simplicity we have assumed the four photons to be all incoming, i.e. $k_0, k_1, k_2, k_3 \rightarrow 0$. The extra-dimensional components $\vec{\mu}$ of the loop momentum satisfy the (anti-)commutation relations

$$\{\not{p}, \not{\mu}\} = 0, \quad \{\not{\mu}, \not{\mu}\} = -\mu^2, \quad (\text{C.13})$$

for any four-dimensional momentum p . This allows to work out the extra-dimensional algebra and rewrite the numerator in terms of four-dimensional spinor products between the polarization vectors ϵ_i , such as $\langle \epsilon_i \epsilon_j \rangle$ and $[\epsilon_i \epsilon_j]$, and scalar products involving the four-dimensional momenta q , k_i , and momenta e_{ij} defined by

$$e_{ij} \equiv \frac{\langle \epsilon_i \gamma^\mu \epsilon_j \rangle}{2}. \quad (\text{C.14})$$

The FORM package SPINNEY [134] can help in this kind of algebraic operations. The final expression can be found in the FORM file `4photons.frm` of the directory `examples` of the distribution.

NINJA includes a small library for massless spinors, which is used internally for building the bases of momenta corresponding to each residue. This can also be useful for defining polarization vectors and other spinor-related objects. The spinor library can be used by including the header file `ninja/spinors.hh` in the source and linking the program with the NINJA library. Polarization vectors can be defined with

```
// Positive helicity (right-handed)
ComplexMomentum epsilon_r = polarizationVectorR(r,k);

// Negative helicity (left-handed)
ComplexMomentum epsilon_l = polarizationVectorL(r,k);
```

where \mathbf{r} is an arbitrary reference momentum and \mathbf{k} is the momentum of the corresponding photon (or gluon). These functions assume the momenta to be incoming, while for outgoing momenta the helicity should be reversed. Angle-bracket and square-bracket spinor products can be computed with the functions `spaa` and `spbb` respectively. If \mathbf{k} is a (real or complex) massless momentum, the corresponding spinor `spinor_k` can be defined as

```
Spinor spinor_k = Spinor(k);
```

and can be used as input parameter for the functions described above. This turns out to be more efficient when several spinor-related operations need to be performed on the same momentum. Instances of the class `Spinor` can also be used in order to define vectors as in Eq. (C.14), using the following function

```
// this returns  $\langle p\gamma^\mu q \rangle / 2$ 
ComplexMomentum
momentumFromSpinors(const Spinor & p, const Spinor & q);
```

In the header file `4photons_num.hh` we define a numerator class `FourPhotons` containing, as private data members, the values of all the momenta and spinor products appearing in the integrand. The numerator methods have been generated with NINJANUMGEN and written in the file `4photons_num.cc`. The file `4photons_init.cc` contains the implementation of an `init` method which initializes the data members using the spinor-related operations described above, while `4photons.cc` contains a simple test. In this test, the mass of the fermion is complex, thus it can have a width. The results have been compared with the ones in Ref [204] as well as with a similar computation performed with SAMURAI for several choices of the external helicity states and the fermion mass. In Figure C.1 we show a typical output for this example.

C.5.3 Six-photon helicity amplitudes

In this example we consider six incoming photons [202, 205–210]. This is a non-trivial case where the `setCutStop` method of an `Amplitude` class can make the computation more efficient when lower point cuts do not contribute to the total result.

A generic six-photon diagram has an integrand of the form

$$\begin{aligned} \mathcal{I} &= \frac{\mathcal{N}(q, \mu^2)}{D_0 D_1 D_2 D_3 D_4 D_5} \\ \mathcal{N}(q, \mu^2) &= -\text{Tr}\left(\bar{l}_1 \epsilon_1 \bar{l}_2 \epsilon_2 \bar{l}_3 \epsilon_3 \bar{l}_4 \epsilon_4 \bar{l}_5 \epsilon_5 \bar{l}_6 \epsilon_6\right) \\ D_i &= \bar{l}_i^2 \end{aligned} \quad (\text{C.15})$$

where we assumed the fermion running in the loop to be massless. The momenta \bar{l}_i are defined by

$$\begin{aligned} \bar{l}_0 &= \bar{q}, & \bar{l}_1 &= \bar{q} + k_0, & \bar{l}_2 &= \bar{q} + k_0 + k_1, \\ \bar{l}_3 &= \bar{q} + k_0 + k_1 + k_2, & \bar{l}_4 &= \bar{q} - k_4 - k_5, & \bar{l}_5 &= \bar{q} - k_5. \end{aligned} \quad (\text{C.16})$$

One can work out the algebra, define the corresponding spinor products and vectors, and generate the input for NINJA in the same way as for the four-photons case. One can also check that the terms proportional to μ^2 in the final expression for the integral vanish upon integration. Therefore, we can perform the simplification $\bar{l}_i \rightarrow l_i$, or equivalently $\mu^2 \rightarrow 0$, in the numerator. Moreover, one can exploit the knowledge that only the cut-constructible contributions of boxes and triangles contribute to the total result, hence we can ask NINJA to stop the reduction at triple cuts with

```
amp.setCutStop(3);
```

and remove the rational part from the result with

```
amp.onlyCutConstructible();
```

which will make the computation more efficient (in the example implemented here, the runtime is reduced by about 33%).

In the file `6photons.cc` we call the method `evaluate` on all the independent permutations of the external legs, generated at run-time with the function `std::next_permutation` of the C++ standard library. The results have been compared with the ones in Ref.s [209, 210] as well as with a similar computation performed with SAMURAI for several helicity choices.

C.5.4 Five-point diagram of $gg \rightarrow Ht\bar{t}$

With this example, we discuss a possible strategy for the generation of the input needed by NINJA which can be suited for more complex computations where an efficient evaluation of the numerator methods at run-time can be important.

We consider the one-loop integral defined by the diagram depicted in Figure C.2, contributing to the 5-point helicity amplitude $g(k_1, -), g(k_2, -) \rightarrow H(k_3), t(k_4, +), \bar{t}(k_5, -)$. The analytic expression for the integrand of this example, which can be worked out from the Feynman rules of the Standard Model, has been generated with the help of the package GOSAM and can be found in the FORM file `ttbarh.frm`. This already contains some abbreviations which are independent of the loop momentum \bar{q} of the diagram. At run-time, these \bar{q} -independent abbreviations are computed only once per phase space point, making thus the evaluation of the numerator and its expansions more efficient. This analytic expression is processed by NINJANUMGEN which produces the numerator expansions. We also add to the numerator class `TTbarHDiagram` an `init` method which uses the spinor library described in Section C.5.2 in order to compute the relevant spinor products and polarization vectors, as well as the abbreviations which do not depend on the loop momentum. These are stored as private data members of the class. For simplicity, our result neglects the coupling constants and an overall color factor.

Even though we considered a single diagram and a specific helicity choice, this example illustrates a general strategy for the generation of an analytic numerator expression which is suited for the numerical evaluations performed by integrand-reduction algorithms like the one implemented in the library NINJA. The full amplitude for this process has been computed in Ref.s [147–151], while an additional jet has recently been added to the final state in Ref.s [99, 100] and Section 5.2 of this thesis, where NINJA has been used for the reduction of the corresponding integrands generated by GOSAM.

C.5.5 Higher-rank example

In this example we show how NINJA can be used in order to compute integrals whose rank is higher than the number of loop denominators. This simple test is similar to the example presented in Section C.3.2 and C.4.1, hence we will describe each step as in the previous case. We define a 5-point amplitude of rank 6, with kinematics $k_0, k_1 \rightarrow k_2, k_3, k_4, k_5$ and integrand

$$\begin{aligned} \mathcal{I} &= \frac{\mathcal{N}(q, \mu^2)}{D_0 D_1 D_2 D_3 D_4} \\ \mathcal{N}(q, \mu^2) &= \prod_{i=0}^2 \left((q \cdot v_{2i})(q \cdot v_{2i+1}) + \mu^2 (v_{2i} \cdot v_{2i+1}) \right) \\ D_i &= \bar{l}_i^2 - m_i^2 \end{aligned} \tag{C.17}$$

in terms of the reference vectors v_i ($i = 0, \dots, 5$) and the momenta \bar{l}_i running into the loop

$$\bar{l}_0 = \bar{q}, \quad \bar{l}_1 = \bar{q} + k_0, \quad \bar{l}_2 = \bar{q} + k_0 + k_1, \quad \bar{l}_3 = \bar{q} + k_3 + k_4, \quad \bar{l}_4 = \bar{q} + k_4. \tag{C.18}$$

We follow the same steps outlined in Section C.3.2. With NINJANUMGEN we generate the methods for NINJA. After writing the integrand in the FORM file `mynumhr.frm` we call the script with the command

```
ninjanumgen mynumhr.frm --nlegs 5 --rank 6 -o mynumhr.cc
```

which creates the file `mynumhr.cc` and a template for the header `mynumhr.hh`. Once again, we define the vectors v_i as public members of the numerator class `Diagram`, by inserting

```
public:
    ninja::ComplexMomentum v0, v1, v2, v3, v4, v5;
```

in the class definition. A possible test program can be almost identical to the one we showed in Section C.4.1, with obvious changes in the definition of the rank, the number of external legs and the reference vectors v_i . This is implemented in the file `simple_higher_rank_test.cc`. In order to run this example, the user must compile the library with the `--enable-higher_rank` option, otherwise a call to the `evaluate` method of an `Amplitude` object will cause a run-time error.

As one can see, when `NINJANUMGEN` is used for the generation of the expansions, the higher-rank case is handled automatically without any intervention by the user. Besides, the internal higher-rank routines of `NINJA` will be automatically called whenever the rank r is equal to $n + 1$ (where n is the number of loop propagators), while in the public programming interface there is no difference with respect to the normal-rank case.

C.5.6 Usage in multi-threaded applications

In the last examples, we wish to illustrate the possibility of using `NINJA` in a multi-threaded application. These examples are implemented using POSIX threads, which are a standard in Unix-like operating systems, but adapting them to different programming interfaces for threads (such as `OPENMP`) should be straightforward.

In order to implement a thread-safe application, one should avoid race conditions which might occur if different threads try to write on the same variables. In particular, one should avoid accessing global variables for writing from different threads. The only global variables used directly by `NINJA` are those controlling the global options described in Section C.4.3. As explained in that section, these options are only meant to change the general behavior of the library for debugging purposes (e.g. for checking that the provided numerator methods are correct), while in general the default options should not be changed during a phase-space integration, especially when performance is important. Hence, on the side of the `NINJA` library, there should be no issue and one can safely call the `evaluate` method from different `Amplitude` objects in different threads.

During a call of the `evaluate` method on an `Amplitude` object, issues might however arise from global variables used by the chosen library of Master Integrals or the numerator methods. As for the numerator methods, all the examples distributed with `NINJA` define a thread-safe numerator class (more specifically, one can safely call numerator methods from different

instances of the class in different threads). This is simply done by using data members of the class instead of global variables, making thus different instances of the same class completely independent.

If the procedures implemented by libraries of Master Integrals are thread-safe, one can therefore use NINJA in multi-threaded applications. As an example, one can use the class `AvHOneLoop` which, as explained in Section C.7, wraps routines of the `ONELOOP` library and adds a cache of computed integrals. The cache is a non-static data member of the class. One can therefore create one instance of this class per thread and assign it accordingly to the `Amplitude` objects to be evaluated in the same thread. As an example, with

```
avh_olo.init(1);
AvHOneLoop my_lib[N_THREADS];
Amplitude<RealMasses> amp[N_THREADS];
for (int i=0; i<N_THREADS; ++i)
    amp[i].setIntegralLibrary(my_lib[i]);
```

we create `N_THREADS` amplitude objects whose `evaluate` method can be safely called in a separate thread (in the first line, we called the `init` method on the global instance `avh_olo` defined in the library, in order to allow `ONELOOP` to perform its global initialization). In this way, different threads will also have an independent cache of Master Integrals. This strategy allows to build a multi-threaded application which uses NINJA for the reduction of one-loop integrals. Recent versions of `LOOPTOOLS` (namely `LOOPTOOLS-2.10` or later) can also be used in threaded applications, since they have a mutex regulating writing access to the internal cache of integrals.

In the following we discuss the possibility to build a multi-threaded application with NINJA and any other (not necessarily thread-safe) library of Master Integrals. Indeed, even though NINJA has obviously no control over possible issues arising from routines of external libraries, we offer an easy way to work around any potential problem. In this case, there is no general way to ensure that calling routines of the same integral library from different threads will not cause conflicts. However, one can avoid these conflicts by scheduling the calls of the external procedures in such a way that they are never evaluated at the same time from two or more threads. If the computation of the Master Integrals takes only a small fraction of the total run time (which is usually the case when a cache of integrals is present), the effects of this on the performance will in general be reasonably small.

Within NINJA, implementing a scheduled access on the routines used by a library of Master Integrals is straightforward. As explained more in detail in Appendix C.7, the generic interface used by NINJA in order to call Master Integral procedures has two methods called `init` and `exit` which are evaluated exactly once in each call of the `evaluate` method, immediately before the computation of the first Master Integral and after the computation of the last Master Integral respectively. Therefore we can use mutexes (such as the ones present the

POSIX standard for threads) in order to *lock* the calls to the Master Integrals in the `init` method and *unlock* them in the `exit` method. This makes sure that, between the calls of the `init` and `exit` methods, no other thread will use the Master Integral routines, hence avoiding any possible conflict.

In order to make a library of Master Integrals thread-safe, we use the template class `ThreadSafeIntegralLibrary`, which is included in the distribution. This automatically wraps an existing class derived from `IntegralLibrary` and adds to it a mutex that schedules the calls to the Master Integrals as explained above. As an example, defining a thread-safe version of a generic library `BaseLibrary` can be simply achieved with

```
#include <ninja/thread_safe_integral_library.hh>
using namespace ninja;
ThreadSafeIntegralLibrary<BaseLibrary> my_lib;
```

which defines a new interface `my_lib` that can be made the default by calling

```
setDefaultIntegralLibrary(my_lib);
```

before any thread is created (alternatively, we could call the `setIntegralLibrary` method on each `Amplitude` object, either outside or inside the threads).

In the files `thread_4photons.cc` and `thread_6photons.cc` we repeat the examples of the four- and six-photons amplitudes, but this time we compute several phase-space points in parallel on different threads. As mentioned before, we do not need to implement other numerator classes, since the ones described in Sections C.5.2 and C.5.3 can be safely used in multi-threaded applications. In the source files, we implement both the approaches described in this section. The preprocessor will select the former if the `ONELOOP` interface has been enabled and the latter otherwise. The multi-threaded examples can be compiled with

```
make thread-examples
```

if at least one between the `ONELOOP` and `LOOPTOOLS` libraries was enabled during configuration and your system supports POSIX threads.

A complete discussion on the implementation of multi-threaded applications for doing phenomenology at one-loop is beyond the purposes of this thesis. Moreover, a detailed assessment of possible advantages of this approach would generally depend on the generator of the integrands and the phase space integration. In these examples we showed that the methods implementing the reduction via Laurent expansion in NINJA can be safely used in multi-threaded programs.

PARTICLE	E	p_x	p_y	p_z
k_0	7.0000000000000000	0.0000000000000000	0.0000000000000000	7.0000000000000000
k_1	7.0000000000000000	0.0000000000000000	0.0000000000000000	-7.0000000000000000
k_2	-6.9999999999999964	-6.1126608202785198	0.8284979592001092	-3.3089226083172685
k_3	-7.0000000000000027	6.1126608202785278	-0.8284979592001093	3.3089226083172703

```

+-----+
|
|  Ninja - version 1.0.0
|
|  Author: Tiziano Peraro
|
|  Based on:
|
|    P. Mastrolia, E. Mirabella and T. Peraro,
|    "Integrand reduction of one-loop scattering amplitudes
|    through Laurent series expansion,"
|    JHEP 1206 (2012) 095 [arXiv:1203.0291 [hep-ph]].
|
|    T. Peraro,
|    "Ninja: Automated Integrand Reduction via Laurent
|    Expansion for One-Loop Amplitudes,"
|    arXiv:1403.1229 [hep-ph]
|
+-----+

Finite:      (-0.184034,-0.16765)
Abs. val.:  0.248948
Single pole: (-3.73035e-13,3.98348e-13)
Double pole: (0,0)

```

Figure C.1: Phase space point and output for the example in `4photons.cc`. It shows the computed finite part and poles of an all-plus four-photon helicity amplitude, using a complex fermion mass $m_f = 10.0 - 1.0i$.

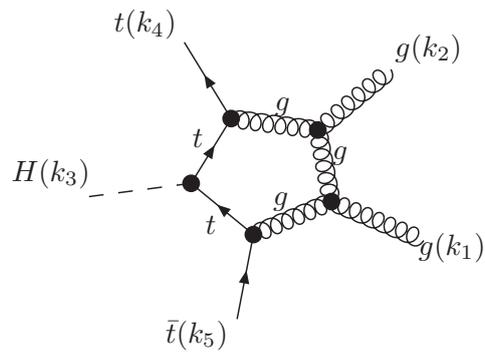


Figure C.2: Diagram contributing to $gg \rightarrow Ht\bar{t}$. This picture has been generated using GoSAM.

C.6 The PYTHON package NINJANUMGEN

The reduction procedures implemented in NINJA take as input a class derived from the abstract class `ninja::Numerator`. This must implement the required expansions in the corresponding methods. If the analytic expression of the numerator can be provided by the user, the source code for the methods can be automatically generated with the help of the simple PYTHON package NINJANUMGEN, which is distributed with the library and can be installed as explained in Section C.1. The package can be used both as a script or as a module within PYTHON.

In Section C.3.2 we already gave a simple example of its usage as a script. As explained there, the user is supposed to create a file containing a FORM expression of the numerator of the integrand. The package uses FORM-4 in order to generate the expansions which are needed and produce a C++ source file with the definitions of the corresponding methods. If not already present, an header file with a sketch of the definition of the class will also be created. The user can complete it by adding data members and methods which are specific of this class. FORM allows one to define symbols between square brackets (e.g. `[symbol_name]`), containing characters which otherwise would not be permitted in a declaration. NINJANUMGEN also allows the usage of such symbols in the expression of the numerator, and it will remove the brackets (which would produce illegal C++ code) when writing the final source files. This gives the user a wider range of possibilities, for instance using symbols which correspond to variable names containing underscores or data members of structures (e.g. with `[structure_instance.data_member]`).

We first give a few more details about the usage of the package as a script. It is invoked with the command

```
ninjanumgen --nlegs NLEGS optional-arguments file
```

where `file` is the name of the file which contains the numerator expression and `NLEGS` is the number of external legs of the loop, which is equal to the number of loop denominators. A description of all the allowed arguments can be obtained with the command

```
ninjanumgen --help
```

and the most important ones are:

`--rank RANK, -r RANK` rank of the numerator, by default it will be assumed to be equal to the number of external legs of the loop

`--diagname DIAGNAME, -d DIAGNAME` name of the numerator expression in the FORM file, by default it will be assumed to be `Diagram`

`--cdiagname CDIAGNAME` name of the numerator class in the generated C++ files, by default it will be the same as the FORM expression

`--formexec FORMEXEC` the FORM executable, the default is `form`
`--qvar QVAR` name of the loop momentum variable q defined in Eq. (3.16), the default is `Q`
`--mu2var MU2VAR` name of the loop variable μ^2 defined in Eq. (3.16), the default is `Mu2`
`--output OUTPUT, -o OUTPUT` name of the output source file, the default is `ninjanumgen.cc`
`--header HEADER` C++ header file containing the definition of the numerator class: if the file does not exist, one will be created. By default it will have the same name as the output but with `.hh` extension.

As mentioned, one can also use the package as a PYTHON module (`ninjanumgen`). This contains a class `DiagramExpansion` which can be used for the generation of the source code which implements the numerator methods. The input parameters of the constructor of this class roughly correspond to the arguments which can be used in the script. A detailed description can be obtained, after installation, by invoking PYTHON in interactive mode (usually done with the command `python`) and typing

```
import ninjanumgen
help(ninjanumgen.DiagramExpansion)
```

The method `writeSource` generates the source files. As a simple example, the source for the integrand we defined in Section C.3.2 could have been generated within PYTHON with the commands

```
# import the module
import ninjanumgen

# define the mandatory arguments for the constructor
n_legs = 4
input_file = 'mynum.frm'
output_file = 'mynum.cc'

# define an instance of the class DiagramExpansion
mynum = ninjanumgen.DiagramExpansion(input_file,
                                     output_file,
                                     n_legs, rank=4)

# generate the source
mynum.writeSource()
```

We suggest to look at the PYTHON files in the `examples` directory for other basic examples.

C.7 Interfaces to Integral Libraries

NINJA already implements interfaces for the ONELOOP and the LOOPTOOLS integral libraries. These libraries have been used in a large number of computations and provide very reliable results, hence they should suffice for most purposes. However, NINJA has been designed considering the possibility of using any other library of Master Integrals.

The Master Integrals are computed by calling virtual methods of the abstract class `ninja::IntegralLibrary`, which is defined in the header file `ninja/integral_library.hh`. Therefore, any library of Master Integrals can be interfaced by implementing a class derived from `IntegralLibrary`. Each method of the library corresponds to a different Master Integral appearing in Eq. (3.29), which should be implemented for both real and complex internal masses (and optionally for the massless case). An implementation of higher-rank integrals can also be provided but it is not needed, since NINJA has a default implementation of them in terms of lower rank integrals. There are two further methods, namely `init` and `exit`. The former is called inside the method `Amplitude::evaluate` just before the computation of the first needed Master Integral, while the latter is called after the last Master Integral has been computed. The method `init`

```
virtual void init(Real muRsq) = 0;
```

takes as input the square of the renormalization scale to be used in the subsequent calls of the methods implementing the Master Integrals. It can also be used in order to perform any other initialization the library might need before computing the integrals. The `exit` method instead, does not need to be implemented and by default it will not perform any action. In Section C.5.6 we gave an example of a case where a non-trivial implementation of the `exit` method could be useful.

The other methods should compute the finite part and the poles of the corresponding Master Integrals. As an example, the methods for the box integrals have the following declarations

```
// - real masses
virtual void
getBoxIntegralRM(Complex rslt[3],
                 Real s21, Real s32, Real s43,
                 Real s14, Real s31, Real s42,
                 Real m1sq, Real m2sq,
                 Real m3sq, Real m4sq) = 0;

// - complex masses
virtual void
getBoxIntegralCM(Complex rslt[3],
```

```

Real s21, Real s32, Real s43,
Real s14, Real s31, Real s42,
const Complex & m1sq,
const Complex & m2sq,
const Complex & m3sq,
const Complex & m4sq) = 0;

```

and they are supposed to write the $\mathcal{O}(\epsilon^{-i})$ term of the result in the i -th entry of the array `rs1t`, for $i \in \{0, 1, 2\}$. The arguments are the invariants s_{ij} and the squared masses m_i^2 . Similar methods need to be provided for 3-point, 2-point and 1-point Master Integrals, as described in detail in the comments inside the header file `ninja/integral_library.hh`.

C.7.1 Built-in interfaces

Examples of implementation of this interface for the libraries `ONELOOP` and `LOOPTOOLS` can be found in the source code. More in detail, we define the instances `ninja::avh_olo` and `ninja::loop_tools` of the classes `ninja::AvHOneLoop` and `ninja::LoopTools` respectively, which implement the methods described above as wrappers of the corresponding routines in each integral library.

The `ONELOOP` interface also implements a cache of Master Integrals on top of these routines. The cache is implemented similarly to a hash table, which allows constant-time look-up of each computed integral from its arguments. Hence, the methods of the `AvHOneLoop` class will call the routines of the `ONELOOP` library only if a Master Integral is not found in the cache. The cache can be cleared with the class method `AvHOneLoop::clearIntegralCache`. During a phase-space integration, we suggest calling this method once per phase space point, especially for more complex processes. This method does not completely free the allocated memory, but keeps the buckets of the hash table available in order to store the integrals more efficiently in subsequent calls of the respective methods. If the user wishes to completely free the allocated memory, the method `AvHOneLoop::freeIntegralCache` can be used, although in general `clearIntegralCache` should be preferred. As already mentioned, every instance of `AvHOneLoop` has a cache of Master Integrals as data member. This can be useful for building multi-threaded applications, as discussed in the examples of Section [C.5.6](#).

Since `LOOPTOOLS` already has an internal cache of Master Integrals, the implementation of its interface is much simpler and only consists in a wrapper of its routines. We implemented a `clearIntegralCache` method in the `LoopTools` class as well, which in this case simply calls the routine which clears the cache of integrals in `LOOPTOOLS`.

Appendix D

Benchmarks of NINJA and GOSAM

In this appendix we collect some benchmarks for several processes computed with the computational framework GOSAM+NINJA, as described in Section 5.1. For each process, we include the kinematics and a detailed list of the input parameters.

The coefficients a_i which appear in the various tables are defined as follows:

$$\frac{a_{-2}}{\epsilon^2} + \frac{a_{-1}}{\epsilon} + a_0 \equiv \frac{2\Re\{\mathcal{M}^{\text{tree-level}}*\mathcal{M}^{\text{one-loop}}\}}{(\alpha_s/2\pi)|\mathcal{M}^{\text{tree-level}}|^2},$$

where the finite part a_0 is computed in the dimensional reduction scheme if not stated otherwise. The reconstruction of the renormalized pole has been checked against the value of a_{-1} and a_{-2} obtained by the universal singular behavior of the dimensionally regularized one-loop amplitudes [142], while the precision of the finite parts is estimated by re-evaluating the amplitudes for a set of momenta rotated by an arbitrary angle about the axis of collision, as described in Section 5.1.3. The accuracy of the results obtained with GOSAM+NINJA is indicated by the underlined digits.

D.1 $pp \rightarrow W + 3 \text{ jets}$

Partonic process: $d\bar{u} \rightarrow \bar{\nu}_e e^- ggg$

The finite part for this process is given in the conventional dimensional regularization (CDR) scheme and was compared with NJET [52]. We also find agreement in the phase space point given by the BLACKHAT collaboration [211].

PARTICLE	E	p_x	p_y	p_z
p_1	500.0000000000000000	0.0000000000000000	0.0000000000000000	500.0000000000000000
p_2	500.0000000000000000	0.0000000000000000	0.0000000000000000	-500.0000000000000000
p_3	414.1300683745429865	232.1455649459389861	332.7544367808189918	-82.9857518524426041
p_4	91.8751521026383955	-43.3570226791010995	-24.0058236140056991	77.3623460793434958
p_5	86.3540681437814044	-15.2133893202618005	37.6335512949163018	-76.2187226821854011
p_6	280.1181818093759830	-83.1261116505822031	-263.2038567586509998	47.7490851160265990
p_7	127.5225295696610033	-90.4490412959934957	-83.1783077030789002	34.0930433392580028

PARAMETER	VALUE
m_W	80.419 GeV
w_W	2.0476 GeV
N_f	5
μ	1000.0 GeV

	$d\bar{u} \rightarrow \bar{\nu}_e e^- ggg$		Ref. [52]
a_0	-91.1772093904611438	-91.17720939055536	
a_{-1}	-57.6891244440692361	-57.68912444409629	
a_{-2}	-11.6666666666668277	-11.66666666666660	

Table D.1: Benchmark point for the subprocess $d(p_1)\bar{u}(p_2) \rightarrow \bar{\nu}_e(p_3)e^-(p_4)g(p_5)g(p_6)g(p_7)$.

D.2 $pp \rightarrow Z + 3$ jets

Partonic process: $d\bar{d} \rightarrow e^+e^-ggg$

The finite part for this process is given in CDR and was compared with NJET [52]. We also find agreement in the phase space point given by the BLACKHAT Collaboration [212].

PARTICLE	E	p_x	p_y	p_z
p_1	500.0000000000000000	0.0000000000000000	0.0000000000000000	500.0000000000000000
p_2	500.0000000000000000	0.0000000000000000	0.0000000000000000	-500.0000000000000000
p_3	414.1300683745429865	232.1455649459389861	332.7544367808189918	-82.9857518524426041
p_4	91.8751521026383955	-43.3570226791010995	-24.0058236140056991	77.3623460793434958
p_5	86.3540681437814044	-15.2133893202618005	37.6335512949163018	-76.2187226821854011
p_6	280.1181818093759830	-83.1261116505822031	-263.2038567586509998	47.7490851160265990
p_7	127.5225295696610033	-90.4490412959934957	-83.1783077030789002	34.0930433392580028

PARAMETER	VALUE
m_W	80.419 GeV
m_Z	91.188 GeV
w_Z	2.4414 GeV
N_f	5
μ	1000.0 GeV

	$d\bar{d} \rightarrow e^+e^-ggg$	Ref. [52]
a_0	-91.0463291277814761	-91.04632919538757
a_{-1}	-57.6876717480941892	-57.68767174883881
a_{-2}	-11.6666666666669485	-11.66666666666667

Table D.2: Benchmark point for the subprocess $d(p_1)\bar{d}(p_2) \rightarrow e^+(p_3)e^-(p_4)g(p_5)g(p_6)g(p_7)$.

D.3 $pp \rightarrow ZZZ + 1 \text{ jet}$

Partonic process: $u\bar{u} \rightarrow ZZZg$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	98.2984277476074197	25.7992145382080409	-12.3240263933454042	23.0200218627010820
p_4	178.4558180449861595	-120.4664227955374400	-73.7638561773535599	-59.8166791207833768
p_5	142.0905125919531145	50.7287365223860434	91.2424257480069656	-31.2402050144644221
p_6	81.1552416154533205	43.9384717349430645	-5.1545431773078745	68.0368622725466565

PARAMETER	VALUE	
m_W	80.376 GeV	
m_Z	91.1876 GeV	
N_f	5	
μ	500.0 GeV	
		$u\bar{u} \rightarrow ZZZg$
a_0	-18.2274687669522883	
a_{-1}	-22.3832348831861125	
a_{-2}	-5.6666666666670755	

Table D.3: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow Z(p_3)Z(p_4)Z(p_5)g(p_6)$.

D.4 $pp \rightarrow WWZ + 1 \text{ jet}$ Partonic process: $u\bar{u} \rightarrow W^+W^-Zg$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	89.1058761118447791	27.0348198946258513	-12.9142626969235721	24.1225229592474193
p_4	179.7207629659015140	-126.2359378733789299	-77.2966387614043384	-62.6814876216509802
p_5	146.1313400758695593	53.1582949292336409	95.6123118862003167	-32.7363964816230890
p_6	85.0420208463841476	46.0428230495191357	-5.4014104278722739	71.2953611440265860

PARAMETER	VALUE
m_W	80.376 GeV
m_Z	91.1876 GeV
N_f	5
μ	500.0 GeV

$u\bar{u} \rightarrow W^+W^-Zg$	
a_0	-18.0050305906438837
a_{-1}	-22.1025452011781987
a_{-2}	-5.6666666666666661

Table D.4: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow W^+(p_3)W^-(p_4)Z(p_5)g(p_6)$.

D.5 $pp \rightarrow WZZ + 1 \text{ jet}$

Partonic process: $u\bar{d} \rightarrow W^+ZZg$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	88.8514948513947331	26.6180234830689777	-12.7151632255396141	23.7506254934495686
p_4	182.6577199093957518	-124.2897556491134168	-76.1049547854528043	-61.7151257515301381
p_5	144.7598590943426586	52.3387523298249846	94.1382547757718982	-32.2316987387113372
p_6	83.7309261448668849	45.3329798362191525	-5.3181367647793465	70.1961989967918498

PARAMETER	VALUE	$u\bar{d} \rightarrow W^+ZZg$	
m_W	80.376 GeV	a_0	-16.6719638614981740
m_Z	91.1876 GeV	a_{-1}	-22.1957678011010735
N_f	5	a_{-2}	-5.6666666666670213
μ	500.0 GeV		

Table D.5: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow W^+(p_3)Z(p_4)Z(p_5)g(p_6)$.

D.6 $pp \rightarrow W W W + 1 \text{ jet}$ Partonic process: $u\bar{d} \rightarrow W^+W^-W^+g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	89.4258007278425993	27.5517039326065500	-13.1611730250244197	24.5837262194154818
p_4	182.4747913234621421	-128.6494675939613614	-78.7744883983191500	-63.8799073098456347
p_5	141.4314519821789986	54.1746388235997784	97.4403424793774207	-33.3622900835894285
p_6	86.6679559665162316	46.9231248377547274	-5.5046810560337240	72.6584711740195104

PARAMETER	VALUE	$u\bar{d} \rightarrow W^+W^-W^+g$
m_W	80.376 GeV	a_0 -15.8859769338002099
m_Z	91.1876 GeV	a_{-1} -21.9889128719035618
N_f	5	a_{-2} -5.6666666666864467
μ	500.0 GeV	

Table D.6: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow W^+(p_3)W^-(p_4)W^+(p_5)g(p_6)$.

D.7 $pp \rightarrow ZZZZ$

Partonic process: $u\bar{u} \rightarrow ZZZZ$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	96.3863867610220666	21.9480082963376795	-10.4843437432888980	19.5836826767025762
p_4	159.2027435644542095	-102.4836644941604078	-62.7526750844588079	-50.8874782857443790
p_5	130.0351856078737001	43.1561483551038094	77.6221118797800074	-26.5767889104262487
p_6	114.3756840666499812	37.3795078427186667	-4.3850930520321931	57.8805845194680018

PARAMETER	VALUE
m_W	80.376 GeV
m_Z	91.1876 GeV
N_f	5
μ	500.0 GeV

$u\bar{u} \rightarrow ZZZZ$	
a_0	10.0010268560339206
a_{-1}	-3.9999999999613696
a_{-2}	-2.6666666666665884

Table D.7: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow Z(p_3)Z(p_4)Z(p_5)Z(p_6)$.

D.8 $pp \rightarrow W W W W$ Partonic process: $u\bar{u} \rightarrow W^+W^-W^+W^-$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	87.7920438094441096	24.8207950462383629	-11.8566452013738353	22.1470015595954024
p_4	168.0494737136866092	-115.8978071108833205	-70.9664068759461202	-57.5481680112689915
p_5	132.1031656516532848	48.8048800986569518	87.7821123452778238	-30.0554392738635485
p_6	112.0553168252159821	42.2721319659877182	-4.9590602679577351	65.4566057255370879

PARAMETER	VALUE	
m_W	80.376 GeV	$u\bar{u} \rightarrow W^+W^-W^+W^-$
m_Z	91.1876 GeV	a_0
N_f	5	a_{-1}
μ	500.0 GeV	a_{-2}

Table D.8: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow W^+(p_3)W^-(p_4)W^+(p_5)W^-(p_6)$.

D.9 $pp \rightarrow t\bar{t}b\bar{b}$

Partonic process: $d\bar{d} \rightarrow t\bar{t}b\bar{b}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	213.1527786548405459	-26.8458616570582116	-117.8628562648380296	-38.8884799556846303
p_4	220.5509350311838546	61.9684848664477599	92.5635212096267281	83.2176445698946168
p_5	42.2714703981682263	-16.8075489037092431	24.9923105627744704	-29.3620149548096769
p_6	24.0248159158073982	-18.3150743056803300	0.3070244924368429	-14.9671496594002438

PARAMETER	VALUE	$d\bar{d} \rightarrow t\bar{t}b\bar{b}$		
m_t	171.2 GeV	a_0	154.6475673006605973	
m_b	4.2 GeV	a_{-1}	2.7409050914577211	
N_f	4	a_{-2}	-2.6666666666666683	
μ	500.0 GeV			

Table D.9: Benchmark point for the subprocess $d(p_1)\bar{d}(p_2) \rightarrow t(p_3)\bar{t}(p_4)b(p_5)\bar{b}(p_6)$.

Partonic process: $gg \rightarrow t\bar{t}b\bar{b}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	194.0670578462199387	-60.6594324624948058	-47.3641590248774236	-49.2915067154802884
p_4	172.4499944124030151	-15.6689752760792977	-7.1446619651393677	-11.5324581958636152
p_5	61.9093840678718479	12.0715208460656545	23.6785835371366993	55.7560301833003820
p_6	71.5735636735052054	64.2568868925084331	30.8302374528801408	5.0679347280435243

PARAMETER	VALUE	$gg \rightarrow t\bar{t}b\bar{b}$		
m_t	171.2 GeV	a_0	165.1250038552732349	
m_b	4.2 GeV	a_{-1}	-3.4472725930136989	
N_f	4	a_{-2}	-6.0000000000001563	
μ	500.0 GeV			

Table D.10: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow t(p_3)\bar{t}(p_4)b(p_5)\bar{b}(p_6)$.

D.10 $pp \rightarrow t\bar{t} + 2$ jetsPartonic process: $gg \rightarrow t\bar{t}gg$

The finite part for this process is given in CDR and was compared with Ref. [213].

PARTICLE	E	p_x	p_y	p_z
p_1	2424.7465026975200999	0.0000000000000000	0.0000000000000000	2424.7465026975200999
p_2	2424.7465026975200999	0.0000000000000000	0.0000000000000000	-2424.7465026975200999
p_3	881.9042263139403985	-715.3340594013137661	-475.1625187999429158	101.1925816377931966
p_4	343.4841188963524132	-24.1478321960174789	-6.3283366075706340	295.5085181344487069
p_5	1673.4634600426329598	21.8782679485017297	1000.4115637957629588	1341.3344089052341133
p_6	1950.6412001421140303	717.6036236488295117	-518.9207083882492952	-1738.0355086774759457

PARAMETER	VALUE
m_t	173.3 GeV
N_f	5
μ	173.3 GeV

	$gg \rightarrow t\bar{t}gg$	Ref. [213]
a_0	-93.9825428068626394	-93.98254280655584
a_{-1}	47.0991735298819236	47.0991735300671
a_{-2}	-11.999999999947171	-11.9999999999874

Table D.11: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow t(p_3)\bar{t}(p_4)g(p_5)g(p_6)$.

D.11 $pp \rightarrow Z b \bar{b} + 1 \text{ jet}$

Partonic process: $ug \rightarrow ue^+e^-b\bar{b}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	34.5908278264605187	21.2091909896352142	-9.4555401980608202	25.6376353599131122
p_4	166.7223525775519022	-156.7623134542972991	-48.7827423195217946	-29.0200617028515602
p_5	111.6942046513332798	30.9750523871488710	106.1302756897373314	-15.8904394000814051
p_6	84.2714416207739418	35.0918815486497024	1.4231216042880672	76.4890217424595988
p_7	102.7211733238803646	69.4861885288632521	-49.3151147764427051	-57.2161559994397777

PARAMETER	VALUE		
m_W	80.376 GeV		
w_W	2.124 GeV		
m_b	4.2 GeV		
N_f	4		
μ	500.0 GeV		
			$ug \rightarrow ue^+e^-b\bar{b}$
		a_0	145.3708954680396630
		a_{-1}	-8.3679512693708471
		a_{-2}	-5.6666666666675392

Table D.12: Benchmark point for the subprocess $u(p_1)g(p_2) \rightarrow u(p_3)e^+(p_4)e^-(p_5)b(p_6)\bar{b}(p_7)$.

D.12 $pp \rightarrow W b \bar{b} + 1 \text{ jet}$ Partonic process: $u\bar{d} \rightarrow e^+ \nu_e b \bar{b} g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	93.4300963683620580	-16.6492363753179085	-37.4579803162897420	-83.9576413803095818
p_4	119.9994454272237192	4.7053605726301706	-100.6826015733804809	65.1209660949429150
p_5	57.9859979994296282	9.1381348721238638	-4.6735873988293006	56.9156220722767259
p_6	104.9559149323387999	87.3260122226470514	54.3049824548373437	-20.5728109201014071
p_7	123.6285452726457805	-84.5202712920831658	88.5091868336622554	-17.5061358668087337

PARAMETER	VALUE	$u\bar{d} \rightarrow e^+ \nu_e b \bar{b} g$	
m_W	80.376 GeV	a_0	129.9538554864771243
w_W	2.124 GeV	a_{-1}	-5.3385683701189715
m_b	4.2 GeV	a_{-2}	-5.6666666666668695
N_f	4		
μ	500.0 GeV		

Table D.13: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow e^+(p_3)\nu_e(p_4)b(p_5)\bar{b}(p_6)g(p_7)$.

D.13 $pp \rightarrow W b \bar{b} + 2 \text{ jets}$

Partonic process: $u\bar{d} \rightarrow e^+ \nu_e b \bar{b} d \bar{d}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	125.6965187314100234	-17.0943040170214537	-113.3597903541534748	51.5456838370753374
p_4	72.1993434263444129	25.0205029393394440	42.4573677065765622	52.7644913865970722
p_5	130.5494441454287085	-59.2470822889473183	116.2307891878907924	-2.3883575291830641
p_6	52.9433261580396106	45.1283306603629413	-14.5296908876010313	-23.1878769876905828
p_7	99.1517346871049057	-5.9484899951818377	-31.1690273964595583	-93.9370730297576273
p_8	19.4596328516722785	12.1410427014482281	0.3703517437466808	15.2031323229588882

PARAMETER	VALUE	$u\bar{d} \rightarrow e^+ \nu_e b \bar{b} d \bar{d}$	
m_W	80.376 GeV	a_0	148.2499564138260837
w_W	2.124 GeV	a_{-1}	-7.7272995122163879
m_b	4.2 GeV	a_{-2}	-5.3333333333331892
N_f	4		
μ	500.0 GeV		

Table D.14: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow e^+(p_3)\nu_e(p_4)b(p_5)\bar{b}(p_6)d(p_7)\bar{d}(p_8)$.

Partonic process: $u\bar{d} \rightarrow e^+ \nu_e b \bar{b} s \bar{s}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	1.8312125180161738	1.2481975210878733	-0.3016228634824342	-1.3055136470806736
p_4	132.0663362603577298	-16.6607423420677527	94.8432229336175965	-90.3808602603960196
p_5	121.6089450080674226	-63.6699664483805989	-102.6429343577717077	13.4780898076629523
p_6	51.8272368161295987	-33.3143054988286877	-36.6615737768635270	-14.6461098360448521
p_7	124.2305315458477253	116.2047315474863467	-2.8413585495376918	43.8361952698163435
p_8	68.4357378515813224	-3.8079147792972257	47.6042666140377335	49.0181986660422169

PARAMETER	VALUE	$u\bar{d} \rightarrow e^+ \nu_e b \bar{b} s \bar{s}$	
m_W	80.376 GeV	a_0	161.1656361677729592
w_W	2.124 GeV	a_{-1}	-1.3276262260753209
m_b	4.2 GeV	a_{-2}	-5.3333333333332735
N_f	4		
μ	500.0 GeV		

Table D.15: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow e^+(p_3)\nu_e(p_4)b(p_5)\bar{b}(p_6)s(p_7)\bar{s}(p_8)$.

Partonic process: $u\bar{d} \rightarrow e^+ \nu_e b \bar{b} g g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	118.4116290336479551	49.7302976872290259	-105.1432905838027665	-22.2058512007951201
p_4	9.9876328289379046	7.4336546931814604	4.5250271451401858	4.9007873616352553
p_5	113.2724870353399353	-8.8682847027615033	110.4231681698792471	23.2614225044082019
p_6	50.3648230617062964	-38.8899369624844553	-31.0689344437605435	-6.4241355543198972
p_7	105.2876197360011901	-28.2113391151385819	10.7916624062129962	100.8620009592996638
p_8	102.6758083043666687	18.8056083999740657	10.4723673063308507	-100.3942240702281055

PARAMETER	VALUE	$u\bar{d} \rightarrow e^+ \nu_e b \bar{b} g g$	
m_W	80.376 GeV	a_0	64.8935770569783301
w_W	2.124 GeV	a_{-1}	-35.9610562256753425
m_b	4.2 GeV	a_{-2}	-8.6666666666670285
N_f	4		
μ	500.0 GeV		

Table D.16: Benchmark point for the subprocess $u(p_1)\bar{d}(p_2) \rightarrow e^+(p_3)\nu_e(p_4)b(p_5)\bar{b}(p_6)g(p_7)g(p_8)$.

D.14 $pp \rightarrow WWb\bar{b}$

Partonic process: $d\bar{d} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	25.6649135887983881	15.7119189648809563	-7.6256852614994211	18.8061776088085644
p_4	154.5064489409292605	-136.9768135536888565	-46.2042967903236104	-54.5413446032057010
p_5	96.1603526761926730	22.3143342117606984	88.8440473781699325	-29.2509685474019605
p_6	61.0731578730670606	23.2528867537403734	2.3717946919774544	56.4234743714476039
p_7	93.6199892430268648	58.9397665353324101	-43.8226058551382351	-57.9028973600646211
p_8	68.9751376779857708	16.7579070879743455	6.4367458368138966	66.4655585304161036

PARAMETER	VALUE
m_W	80.376 GeV
m_Z	91.1876 GeV
m_t	171.2 GeV
m_b	4.2 GeV
N_f	4
μ	500.0 GeV

$d\bar{d} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}$	
a_0	118.3990066409585751
a_{-1}	8.5574384926230991
a_{-2}	-2.6666666666666492

Table D.17: Benchmark point for the subprocess $d(p_1)\bar{d}(p_2) \rightarrow \nu_e(p_3)e^+(p_4)\bar{\nu}_\mu(p_5)\mu^-(p_6)b(p_7)\bar{b}(p_8)$.

Partonic process: $gg \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b \bar{b}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	25.6649135887983881	15.7119189648809563	-7.6256852614994211	18.8061776088085644
p_4	154.5064489409292605	-136.9768135536888565	-46.2042967903236104	-54.5413446032057010
p_5	96.1603526761926730	22.3143342117606984	88.8440473781699325	-29.2509685474019605
p_6	61.0731578730670606	23.2528867537403734	2.3717946919774544	56.4234743714476039
p_7	93.6199892430268648	58.9397665353324101	-43.8226058551382351	-57.9028973600646211
p_8	68.9751376779857708	16.7579070879743455	6.4367458368138966	66.4655585304161036

PARAMETER	VALUE
m_W	80.376 GeV
m_Z	91.1876 GeV
m_t	171.2 GeV
m_b	4.2 GeV
N_f	4
μ	500.0 GeV

$gg \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b \bar{b}$	
a_0	<u>27.4387494492212056</u>
a_{-1}	<u>-7.9555523940773458</u>
a_{-2}	<u>-5.9999999999999885</u>

Table D.18: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow \nu_e(p_3)e^+(p_4)\bar{\nu}_\mu(p_5)\mu^-(p_6)b(p_7)\bar{b}(p_8)$.

D.15 $pp \rightarrow WWb\bar{b} + 1 \text{ jet}$

Partonic process: $u\bar{u} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	18.1435357791203664	10.6794732394995044	-3.3652659935279523	14.2762644514536543
p_4	132.6867460765342059	-121.1435093670056773	-5.2844663256145745	-53.8711159925087628
p_5	92.9820970636741606	17.2946496492191066	87.0024013115393018	-27.8773677719967203
p_6	46.6813566690488102	14.7875289839330328	9.6031823094958089	43.2233378690595345
p_7	69.5794003816978091	41.1619960772432023	-24.4397402957503331	-50.4943772185567497
p_8	53.6926389028252160	9.1521313351918430	14.4565277002773502	50.8934845655482206
p_9	86.2342251270994211	28.0677300819189206	-77.9726387064195876	23.8497740970007897

PARAMETER	VALUE
m_W	80.376 GeV
m_Z	91.1876 GeV
m_t	171.2 GeV
m_b	0.0 GeV
N_f	5
μ^2	500.0 GeV

$u\bar{u} \rightarrow \nu_e e^+ \bar{\nu}_\mu \mu^- b\bar{b}g$	
a_0	-38.5591246673060795
a_{-1}	-32.4828496060584442
a_{-2}	-8.3333333333334405

Table D.19: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow \nu_e(p_3)e^+(p_4)\bar{\nu}_\mu(p_5)\mu^-(p_6)b(p_7)\bar{b}(p_8)g(p_9)$.

D.16 $pp \rightarrow H + 3 \text{ jets (GF)}$ Partonic process: $dd \rightarrow Hgdd$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	144.2726812297522656	26.1871426153409317	-66.9976759414063139	3.8924965402436307
p_4	135.5632052379070274	5.8338195550562180	26.5129120011233681	-132.8172227574218596
p_5	75.6651361325424006	-19.2292152047334604	-65.8505932559228597	31.9241206051040152
p_6	144.4989773997982923	-12.7917469656637373	106.3353571962057913	97.0006056120742102

PARAMETER	VALUE	$dd \rightarrow Hgdd$	
m_H	125.0 GeV	a_0	45.0406751589844419
N_f	5	a_{-1}	-38.5795131098704047
μ	500 GeV	a_{-2}	-8.3333333333337265

Table D.20: Benchmark point for the subprocess $d(p_1)d(p_2) \rightarrow H(p_3)g(p_4)d(p_5)d(p_6)$.Partonic process: $d\bar{d} \rightarrow Hgu\bar{u}$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	144.2726812297522656	26.1871426153409317	-66.9976759414063139	3.8924965402436307
p_4	135.5632052379070274	5.8338195550562180	26.5129120011233681	-132.8172227574218596
p_5	75.6651361325424006	-19.2292152047334604	-65.8505932559228597	31.9241206051040152
p_6	144.4989773997982923	-12.7917469656637373	106.3353571962057913	97.0006056120742102

PARAMETER	VALUE	$d\bar{d} \rightarrow Hgu\bar{u}$	
m_H	125.0 GeV	a_0	-4.0311565363569581
N_f	5	a_{-1}	-38.5799895715348669
μ	500 GeV	a_{-2}	-8.3333333333335418

Table D.21: Benchmark point for the subprocess $d(p_1)\bar{d}(p_2) \rightarrow H(p_3)g(p_4)u(p_5)\bar{u}(p_6)$.

Partonic process: $d\bar{d} \rightarrow Hgg$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	144.2726812297522656	26.1871426153409317	-66.9976759414063139	3.8924965402436307
p_4	135.5632052379070274	5.8338195550562180	26.5129120011233681	-132.8172227574218596
p_5	75.6651361325424006	-19.2292152047334604	-65.8505932559228597	31.9241206051040152
p_6	144.4989773997982923	-12.7917469656637373	106.3353571962057913	97.0006056120742102

PARAMETER	VALUE	$d\bar{d} \rightarrow Hgg$	
m_H	125.0	a_0	-8.8224441195131735
N_f	5	a_{-1}	-48.7670278783515414
μ^2	500 GeV	a_{-2}	-11.6666666666666785

Table D.22: Benchmark point for the subprocess $d(p_1)\bar{d}(p_2) \rightarrow H(p_3)g(p_4)g(p_5)g(p_6)$.**Partonic process: $gg \rightarrow Hgg$**

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	144.2726812297522656	26.1871426153409317	-66.9976759414063139	3.8924965402436307
p_4	135.5632052379070274	5.8338195550562180	26.5129120011233681	-132.8172227574218596
p_5	75.6651361325424006	-19.2292152047334604	-65.8505932559228597	31.9241206051040152
p_6	144.4989773997982923	-12.7917469656637373	106.3353571962057913	97.0006056120742102

PARAMETER	VALUE	$gg \rightarrow Hgg$	
m_H	125.0 GeV	a_0	23.4307927578718953
N_f	5	a_{-1}	-56.3734964424517315
μ	500.0 GeV	a_{-2}	-15.0000000000008757

Table D.23: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow H(p_3)g(p_4)g(p_5)g(p_6)$.

D.17 $pp \rightarrow Z t \bar{t} + 1 \text{ jet}$ Partonic process: $u\bar{u} \rightarrow t\bar{t}e^+e^-g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	183.2414081421947287	-30.2337217736484156	32.1314578860740667	48.1815850690226029
p_4	199.0327070603159996	74.5268539046026035	-40.9270527537185629	-55.4554134393922311
p_5	70.1181125436057044	-63.0760999348447697	21.5315800178266556	21.7794946135846281
p_6	20.7607087314536756	-7.2430664321972609	-7.1983324871256098	-18.0756472939650585
p_7	26.8470635224299627	26.0260342360878454	-5.5376526630565506	3.5699810507501222

PARAMETER	VALUE
m_Z	91.1876 GeV
m_t	171.2 GeV
N_f	5
μ	500.0 GeV

$u\bar{u} \rightarrow t\bar{t}e^+e^-g$	
a_0	-20.4367763710913373
a_{-1}	-25.9078542815554513
a_{-2}	-5.6666666665792098

Table D.24: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow t(p_3)\bar{t}(p_4)e^+(p_5)e^-(p_6)g(p_7)$.Partonic process: $gg \rightarrow t\bar{t}e^+e^-g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	174.2203895522303014	-25.0977827305029138	-19.5610151031829993	5.5472629175473589
p_4	186.7123996976260685	-14.0800163072181022	56.3619207264196902	-46.6601246640355427
p_5	60.3016377245591073	38.1795332240129639	22.1553968884492853	41.0822241824339116
p_6	18.6184873501163182	5.2347824612577458	-1.6661313271933778	-17.7895792583830961
p_7	60.1470856754682259	-4.2365166475497116	-57.2901711844925998	17.8202168224373914

PARAMETER	VALUE
m_Z	91.1876 GeV
m_t	171.2 GeV
N_f	5
μ	500.0 GeV

$gg \rightarrow t\bar{t}e^+e^-g$	
a_0	9.2826425323344441
a_{-1}	-26.2816094048822784
a_{-2}	-9.0000000000005702

Table D.25: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow t(p_3)\bar{t}(p_4)e^+(p_5)e^-(p_6)g(p_7)$.

D.18 $pp \rightarrow Ht\bar{t} + 1 \text{ jet}$

Partonic process: $u\bar{u} \rightarrow Ht\bar{t}g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	126.3747891763443505	6.8355633672742222	-3.2652801590882752	6.0992096455298030
p_4	177.2234233286846745	-31.9178657717747534	-19.5439094615872051	-15.8485716665707326
p_5	174.8995128490773538	13.4406996200208031	24.1748981179500326	-8.2771667589629576
p_6	21.5022746458936318	11.6416027844796517	-1.3657084972745175	18.0265287800038720

PARAMETER	VALUE	$u\bar{u} \rightarrow Ht\bar{t}g$	
m_H	126.0 GeV	a_0	-80.4023340755848750
m_t	172.5 GeV	a_{-1}	-32.6902910205892141
N_f	5	a_{-2}	-5.6666666666679379
μ^2	500 GeV		

Table D.26: Benchmark point for the subprocess $u(p_1)\bar{u}(p_2) \rightarrow H(p_3)t(p_4)\bar{t}(p_5)g(p_6)$.

Partonic process: $gg \rightarrow Ht\bar{t}g$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	126.3747891763443505	6.8355633672742222	-3.2652801590882752	6.0992096455298030
p_4	177.2234233286846745	-31.9178657717747534	-19.5439094615872051	-15.8485716665707326
p_5	174.8995128490773538	13.4406996200208031	24.1748981179500326	-8.2771667589629576
p_6	21.5022746458936318	11.6416027844796517	-1.3657084972745175	18.0265287800038720

PARAMETER	VALUE	$gg \rightarrow Ht\bar{t}g$	
m_H	126.0 GeV	a_0	-45.6979334407767297
m_t	172.5 GeV	a_{-1}	-35.9217497445515619
N_f	5	a_{-2}	-8.9999999999990887
μ	500.0 GeV		

Table D.27: Benchmark point for the subprocess $g(p_1)g(p_2) \rightarrow H(p_3)t(p_4)\bar{t}(p_5)g(p_6)$.

D.19 $pp \rightarrow H + 3 \text{ jets (VBF)}$ **Partonic process:** $uu \rightarrow gHuu$ (VBF)

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	40.1603071333660182	-14.0012702315405289	0.4395613413056457	37.6380324509251736
p_4	127.7084583092647421	-23.4171446211731009	-10.8486559189339324	4.2888607196847408
p_5	145.0573545181100599	-109.8833468186949176	-94.5094823127907233	5.9366610719649477
p_6	187.0738800392591656	147.3017616714085705	104.9185768904190468	-47.8635542425748710

PARAMETER	VALUE	$uu \rightarrow gHuu$	
m_H	125.0 GeV	a_0	-94.6818287259862359
m_Z	91.1876 GeV	a_{-1}	-40.8904107779583796
w_Z	2.4952 GeV	a_{-2}	-8.3333333333336821
N_f	5		
μ	500.0 GeV		

Table D.28: Benchmark point for the subprocess $u(p_1)u(p_2) \rightarrow g(p_3)H(p_4)u(p_5)u(p_6)$.

D.20 $pp \rightarrow H + 4 \text{ jets (VBF)}$

Partonic process: $uu \rightarrow ggHuu$ (VBF)

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	77.8540004794647871	-42.9509851388533761	6.0812524321357140	-64.6488718781321126
p_4	179.6868495846460405	66.2917119993839208	-68.8295351006103004	152.1685510599341171
p_5	140.8511015083574875	-29.7530986170501173	2.6554844463192953	-57.6344889901617350
p_6	81.8035523206006161	2.3404081739038114	78.3533883805051659	-23.3899591949725902
p_7	19.8044961069310155	4.0719635826157594	-18.2605901583498813	-6.4952309966676500

PARAMETER	VALUE	$uu \rightarrow ggHuu$	
m_H	125.0 GeV	a_0	-85.2117220498222565
m_Z	91.1876 GeV	a_{-1}	-54.2263214854450339
w_Z	2.4952 GeV	a_{-2}	-11.33333333333333464
N_f	5		
μ	500.0 GeV		

Table D.29: Benchmark point for the subprocess $u(p_1)u(p_2) \rightarrow g(p_3)g(p_4)H(p_5)u(p_6)u(p_7)$.

Partonic process: $uu \rightarrow u\bar{u}Huu$

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	30.3977507329956786	18.6382269943320686	-8.3093459176077840	22.5298582897146815
p_4	146.5123801227517504	-137.7596903034009301	-42.8693308104768249	-25.5022691658345515
p_5	158.9318930425956466	27.2202771824319179	93.2652344091314376	-13.9642109273819610
p_6	73.9640816926485911	30.8380670569460271	1.2506117519626909	67.2170735086997695
p_7	90.1938944090082799	61.0631190696906998	-43.3371694330094499	-50.2804517051979616

PARAMETER	VALUE	$uu \rightarrow u\bar{u}Huu$	
m_H	125.0 GeV	a_0	-36.9909931379802686
m_Z	91.1876 GeV	a_{-1}	-35.2029797282532968
w_Z	2.4952 GeV	a_{-2}	-8.0000000000000533
N_f	5		
μ	500.0 GeV		

Table D.30: Benchmark point for the subprocess $u(p_1)u(p_2) \rightarrow u(p_3)\bar{u}(p_4)H(p_5)u(p_6)u(p_7)$.

D.21 $pp \rightarrow H + 5 \text{ jets (VBF)}$ Partonic process: $uu \rightarrow gggHuu$ (VBF)

PARTICLE	E	p_x	p_y	p_z
p_1	250.0000000000000000	0.0000000000000000	0.0000000000000000	250.0000000000000000
p_2	250.0000000000000000	0.0000000000000000	0.0000000000000000	-250.0000000000000000
p_3	24.3265597158113103	9.0293044031657743	17.9817135111430346	-13.6715452237514459
p_4	74.2975116145394878	11.5035704263332619	29.2267551012052458	-67.3319009520241991
p_5	97.6019108689663568	-72.1835660496625877	4.7127971252997813	-65.5244636825316746
p_6	136.6365204371290929	23.6483158828115840	-47.1886774719599131	-16.0786999325225715
p_7	120.9034677660200998	23.6253257672831865	-14.6581830826285433	117.6632065216435592
p_8	46.2340295975336915	4.3770495700687935	9.9255948169403965	44.9434032691863266

PARAMETER	VALUE	
m_H	125.0 GeV	
m_Z	91.1876 GeV	
w_Z	2.4952 GeV	
N_f	5	
μ	500.0 GeV	
		$uu \rightarrow gggHuu$
a_0	-164.8823178520154897	
a_{-1}	-81.4472715794169630	
a_{-2}	-14.3333333333333570	

Table D.31: Benchmark point for the subprocess $u(p_1)u(p_2) \rightarrow g(p_3)g(p_4)g(p_5)H(p_6)u(p_7)u(p_8)$.

Bibliography

- [1] G. Aad *et al.*, “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC,” *Phys.Lett.*, vol. B716, pp. 1–29, 2012.
- [2] S. Chatrchyan *et al.*, “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC,” *Phys.Lett.*, vol. B716, pp. 30–61, 2012.
- [3] R. Cutkosky, “Singularities and discontinuities of Feynman amplitudes,” *J.Math.Phys.*, vol. 1, pp. 429–433, 1960.
- [4] Z. Bern, L. J. Dixon, D. C. Dunbar, and D. A. Kosower, “One loop n point gauge theory amplitudes, unitarity and collinear limits,” *Nucl.Phys.*, vol. B425, pp. 217–260, 1994.
- [5] R. Britto, F. Cachazo, and B. Feng, “Generalized unitarity and one-loop amplitudes in N=4 super-Yang-Mills,” *Nucl.Phys.*, vol. B725, pp. 275–305, 2005.
- [6] F. Cachazo, P. Svrcek, and E. Witten, “MHV vertices and tree amplitudes in gauge theory,” *JHEP*, vol. 0409, p. 006, 2004.
- [7] R. Britto, F. Cachazo, and B. Feng, “New recursion relations for tree amplitudes of gluons,” *Nucl.Phys.*, vol. B715, pp. 499–522, 2005.
- [8] E. Witten, “Perturbative gauge theory as a string theory in twistor space,” *Commun.Math.Phys.*, vol. 252, pp. 189–258, 2004.
- [9] J. Drummond, J. Henn, V. Smirnov, and E. Sokatchev, “Magic identities for conformal four-point integrals,” *JHEP*, vol. 0701, p. 064, 2007.
- [10] Z. Bern, M. Czakon, L. J. Dixon, D. A. Kosower, and V. A. Smirnov, “The Four-Loop Planar Amplitude and Cusp Anomalous Dimension in Maximally Supersymmetric Yang-Mills Theory,” *Phys.Rev.*, vol. D75, p. 085010, 2007.
- [11] J. Drummond, J. Henn, G. Korchemsky, and E. Sokatchev, “Dual superconformal symmetry of scattering amplitudes in N=4 super-Yang-Mills theory,” *Nucl.Phys.*, vol. B828, pp. 317–374, 2010.

-
- [12] A. Brandhuber, P. Heslop, and G. Travaglini, “A Note on dual superconformal symmetry of the N=4 super Yang-Mills S-matrix,” *Phys.Rev.*, vol. D78, p. 125005, 2008.
- [13] N. Arkani-Hamed, F. Cachazo, C. Cheung, and J. Kaplan, “A Duality For The S Matrix,” *JHEP*, vol. 1003, p. 020, 2010.
- [14] L. Mason and D. Skinner, “Dual Superconformal Invariance, Momentum Twistors and Grassmannians,” *JHEP*, vol. 0911, p. 045, 2009.
- [15] N. Arkani-Hamed, F. Cachazo, and C. Cheung, “The Grassmannian Origin Of Dual Superconformal Invariance,” *JHEP*, vol. 1003, p. 036, 2010.
- [16] L. F. Alday and J. M. Maldacena, “Gluon scattering amplitudes at strong coupling,” *JHEP*, vol. 0706, p. 064, 2007.
- [17] Z. Bern, J. Carrasco, and H. Johansson, “New Relations for Gauge-Theory Amplitudes,” *Phys.Rev.*, vol. D78, p. 085011, 2008.
- [18] Z. Bern, J. J. M. Carrasco, and H. Johansson, “Perturbative Quantum Gravity as a Double Copy of Gauge Theory,” *Phys.Rev.Lett.*, vol. 105, p. 061602, 2010.
- [19] R. Britto, F. Cachazo, B. Feng, and E. Witten, “Direct proof of tree-level recursion relation in Yang-Mills theory,” *Phys.Rev.Lett.*, vol. 94, p. 181602, 2005.
- [20] S. Badger, E. N. Glover, V. Khoze, and P. Svrcek, “Recursion relations for gauge theory amplitudes with massive particles,” *JHEP*, vol. 0507, p. 025, 2005.
- [21] Z. Bern, L. J. Dixon, and D. A. Kosower, “Bootstrapping multi-parton loop amplitudes in QCD,” *Phys.Rev.*, vol. D73, p. 065013, 2006.
- [22] Z. Bern, L. J. Dixon, D. C. Dunbar, and D. A. Kosower, “Fusing gauge theory tree amplitudes into loop amplitudes,” *Nucl.Phys.*, vol. B435, pp. 59–101, 1995.
- [23] Z. Bern and A. Morgan, “Massive loop amplitudes from unitarity,” *Nucl.Phys.*, vol. B467, pp. 479–509, 1996.
- [24] R. Britto, E. Buchbinder, F. Cachazo, and B. Feng, “One-loop amplitudes of gluons in SQCD,” *Phys.Rev.*, vol. D72, p. 065012, 2005.
- [25] R. Britto, B. Feng, and P. Mastrolia, “The Cut-constructible part of QCD amplitudes,” *Phys.Rev.*, vol. D73, p. 105004, 2006.
- [26] R. Britto, B. Feng, and P. Mastrolia, “Closed-Form Decomposition of One-Loop Massive Amplitudes,” *Phys.Rev.*, vol. D78, p. 025031, 2008.

-
- [27] P. Mastrolia, “Double-Cut of Scattering Amplitudes and Stokes’ Theorem,” *Phys.Lett.*, vol. B678, pp. 246–249, 2009.
- [28] N. Bjerrum-Bohr, D. C. Dunbar, and W. B. Perkins, “Analytic structure of three-mass triangle coefficients,” *JHEP*, vol. 0804, p. 038, 2008.
- [29] P. Mastrolia, “On Triple-cut of scattering amplitudes,” *Phys.Lett.*, vol. B644, pp. 272–283, 2007.
- [30] D. Forde, “Direct extraction of one-loop integral coefficients,” *Phys.Rev.*, vol. D75, p. 125019, 2007.
- [31] E. Nigel Glover and C. Williams, “One-Loop Gluonic Amplitudes from Single Unitarity Cuts,” *JHEP*, vol. 0812, p. 067, 2008.
- [32] R. Britto and B. Feng, “Solving for tadpole coefficients in one-loop amplitudes,” *Phys.Lett.*, vol. B681, pp. 376–381, 2009.
- [33] R. Britto and E. Mirabella, “Single Cut Integration,” *JHEP*, vol. 1101, p. 135, 2011.
- [34] A. Brandhuber, S. McNamara, B. J. Spence, and G. Travaglini, “Loop amplitudes in pure Yang-Mills from generalised unitarity,” *JHEP*, vol. 0510, p. 011, 2005.
- [35] C. Anastasiou, R. Britto, B. Feng, Z. Kunszt, and P. Mastrolia, “D-dimensional unitarity cut method,” *Phys.Lett.*, vol. B645, pp. 213–216, 2007.
- [36] C. Anastasiou, R. Britto, B. Feng, Z. Kunszt, and P. Mastrolia, “Unitarity cuts and Reduction to master integrals in d dimensions for one-loop amplitudes,” *JHEP*, vol. 0703, p. 111, 2007.
- [37] S. Badger, “Direct Extraction Of One Loop Rational Terms,” *JHEP*, vol. 0901, p. 049, 2009.
- [38] G. Passarino and M. Veltman, “One Loop Corrections for $e^+ e^-$ Annihilation Into $\mu^+ \mu^-$ in the Weinberg Model,” *Nucl.Phys.*, vol. B160, p. 151, 1979.
- [39] A. Denner and S. Dittmaier, “Reduction schemes for one-loop tensor integrals,” *Nucl.Phys.*, vol. B734, pp. 62–115, 2006.
- [40] G. Ossola, C. G. Papadopoulos, and R. Pittau, “Reducing full one-loop amplitudes to scalar integrals at the integrand level,” *Nucl.Phys.*, vol. B763, pp. 147–169, 2007.
- [41] R. K. Ellis, W. Giele, and Z. Kunszt, “A Numerical Unitarity Formalism for Evaluating One-Loop Amplitudes,” *JHEP*, vol. 0803, p. 003, 2008.

- [42] G. Ossola, C. G. Papadopoulos, and R. Pittau, “CutTools: A Program implementing the OPP reduction method to compute one-loop amplitudes,” *JHEP*, vol. 0803, p. 042, 2008.
- [43] P. Mastrolia, G. Ossola, T. Reiter, and F. Tramontano, “Scattering AMplitudes from Unitarity-based Reduction Algorithm at the Integrand-level,” *JHEP*, vol. 1008, p. 080, 2010.
- [44] T. Hahn and M. Perez-Victoria, “Automatized one loop calculations in four-dimensions and D-dimensions,” *Comput.Phys.Commun.*, vol. 118, pp. 153–165, 1999.
- [45] A. van Hameren, C. Papadopoulos, and R. Pittau, “Automated one-loop calculations: A Proof of concept,” *JHEP*, vol. 0909, p. 106, 2009.
- [46] G. Bevilacqua, M. Czakon, M. Garzelli, A. van Hameren, A. Kardos, *et al.*, “HELAC-NLO,” *Comput.Phys.Commun.*, vol. 184, pp. 986–997, 2013.
- [47] C. Berger, Z. Bern, L. Dixon, F. Febres Cordero, D. Forde, *et al.*, “An Automated Implementation of On-Shell Methods for One-Loop Amplitudes,” *Phys.Rev.*, vol. D78, p. 036003, 2008.
- [48] V. Hirschi, R. Frederix, S. Frixione, M. V. Garzelli, F. Maltoni, *et al.*, “Automation of one-loop QCD corrections,” *JHEP*, vol. 1105, p. 044, 2011.
- [49] G. Cullen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, *et al.*, “Automated One-Loop Calculations with GoSam,” *Eur.Phys.J.*, vol. C72, p. 1889, 2012.
- [50] F. Cascioli, P. Maierhofer, and S. Pozzorini, “Scattering Amplitudes with Open Loops,” *Phys.Rev.Lett.*, vol. 108, p. 111601, 2012.
- [51] S. Badger, B. Biedermann, and P. Uwer, “NGLuon: A Package to Calculate One-loop Multi-gluon Amplitudes,” *Comput.Phys.Commun.*, vol. 182, pp. 1674–1692, 2011.
- [52] S. Badger, B. Biedermann, P. Uwer, and V. Yundin, “Numerical evaluation of virtual corrections to multi-jet production in massless QCD,” *Comput.Phys.Commun.*, vol. 184, pp. 1981–1998, 2013.
- [53] F. Tkachov, “A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions,” *Phys.Lett.*, vol. B100, pp. 65–68, 1981.
- [54] K. Chetyrkin and F. Tkachov, “Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops,” *Nucl.Phys.*, vol. B192, pp. 159–204, 1981.
- [55] S. Laporta, “High precision calculation of multiloop Feynman integrals by difference equations,” *Int.J.Mod.Phys.*, vol. A15, pp. 5087–5159, 2000.

- [56] S. Laporta, “Calculation of master integrals by difference equations,” *Phys.Lett.*, vol. B504, pp. 188–194, 2001.
- [57] R. N. Lee, A. V. Smirnov, and V. A. Smirnov, “Dimensional recurrence relations: an easy way to evaluate higher orders of expansion in ϵ ,” *Nucl.Phys.Proc.Suppl.*, vol. 205–206, pp. 308–313, 2010.
- [58] A. Kotikov, “Differential equation method: The Calculation of N point Feynman diagrams,” *Phys.Lett.*, vol. B267, pp. 123–127, 1991.
- [59] E. Remiddi, “Differential equations for Feynman graph amplitudes,” *Nuovo Cim.*, vol. A110, pp. 1435–1452, 1997.
- [60] T. Gehrmann and E. Remiddi, “Using differential equations to compute two loop box integrals,” *Nucl.Phys.Proc.Suppl.*, vol. 89, pp. 251–255, 2000.
- [61] T. Gehrmann and E. Remiddi, “Two loop master integrals for $\gamma^* \rightarrow 3$ jets: The Planar topologies,” *Nucl.Phys.*, vol. B601, pp. 248–286, 2001.
- [62] T. Gehrmann and E. Remiddi, “Two loop master integrals for $\gamma^* \rightarrow 3$ jets: The Non-planar topologies,” *Nucl.Phys.*, vol. B601, pp. 287–317, 2001.
- [63] R. Bonciani, P. Mastrolia, and E. Remiddi, “Vertex diagrams for the QED form-factors at the two loop level,” *Nucl.Phys.*, vol. B661, pp. 289–343, 2003.
- [64] M. Argeri and P. Mastrolia, “Feynman Diagrams and Differential Equations,” *Int.J.Mod.Phys.*, vol. A22, pp. 4375–4436, 2007.
- [65] J. M. Henn, “Multiloop integrals in dimensional regularization made simple,” *Phys.Rev.Lett.*, vol. 110, no. 25, p. 251601, 2013.
- [66] J. M. Henn, A. V. Smirnov, and V. A. Smirnov, “Analytic results for planar three-loop four-point integrals from a Knizhnik-Zamolodchikov equation,” *JHEP*, vol. 1307, p. 128, 2013.
- [67] J. M. Henn and V. A. Smirnov, “Analytic results for two-loop master integrals for Bhabha scattering I,” *JHEP*, vol. 1311, p. 041, 2013.
- [68] M. Argeri, S. Di Vita, P. Mastrolia, E. Mirabella, J. Schlenk, *et al.*, “Magnus and Dyson Series for Master Integrals,” *JHEP*, vol. 1403, p. 082, 2014.
- [69] V. A. Smirnov, “Analytical result for dimensionally regularized massless on shell double box,” *Phys.Lett.*, vol. B460, pp. 397–404, 1999.
- [70] J. Tausk, “Nonplanar massless two loop Feynman diagrams with four on-shell legs,” *Phys.Lett.*, vol. B469, pp. 225–234, 1999.

- [71] M. Czakon, “Automatized analytic continuation of Mellin-Barnes integrals,” *Comput.Phys.Commun.*, vol. 175, pp. 559–571, 2006.
- [72] A. Smirnov and V. Smirnov, “On the Resolution of Singularities of Multiple Mellin-Barnes Integrals,” *Eur.Phys.J.*, vol. C62, pp. 445–449, 2009.
- [73] M. Beneke and V. A. Smirnov, “Asymptotic expansion of Feynman integrals near threshold,” *Nucl.Phys.*, vol. B522, pp. 321–344, 1998.
- [74] V. A. Smirnov, “Asymptotic expansions in limits of large momenta and masses,” *Commun.Math.Phys.*, vol. 134, pp. 109–137, 1990.
- [75] V. Smirnov, “Feynman integral calculus,” 2006.
- [76] T. Binoth and G. Heinrich, “An automatized algorithm to compute infrared divergent multiloop integrals,” *Nucl.Phys.*, vol. B585, pp. 741–759, 2000.
- [77] T. Binoth and G. Heinrich, “Numerical evaluation of multiloop integrals by sector decomposition,” *Nucl.Phys.*, vol. B680, pp. 375–388, 2004.
- [78] C. Bogner and S. Weinzierl, “Resolution of singularities for multi-loop integrals,” *Comput.Phys.Commun.*, vol. 178, pp. 596–610, 2008.
- [79] G. Heinrich, “Sector Decomposition,” *Int.J.Mod.Phys.*, vol. A23, pp. 1457–1486, 2008.
- [80] A. Smirnov, V. Smirnov, and M. Tentyukov, “FIESTA 2: Parallelizeable multiloop numerical calculations,” *Comput.Phys.Commun.*, vol. 182, pp. 790–803, 2011.
- [81] J. Carter and G. Heinrich, “SecDec: A general program for sector decomposition,” *Comput.Phys.Commun.*, vol. 182, pp. 1566–1581, 2011.
- [82] S. Borowka, J. Carter, and G. Heinrich, “Numerical Evaluation of Multi-Loop Integrals for Arbitrary Kinematics with SecDec 2.0,” *Comput.Phys.Commun.*, vol. 184, pp. 396–408, 2013.
- [83] S. Borowka and G. Heinrich, “Massive non-planar two-loop four-point integrals with SecDec 2.1,” *Comput.Phys.Commun.*, vol. 184, pp. 2552–2561, 2013.
- [84] C. Anastasiou, S. Beerli, and A. Daleo, “Evaluating multi-loop Feynman diagrams with infrared and threshold singularities numerically,” *JHEP*, vol. 0705, p. 071, 2007.
- [85] Z. Bern, J. Rozowsky, and B. Yan, “Two loop four gluon amplitudes in N=4 superYang-Mills,” *Phys.Lett.*, vol. B401, pp. 273–282, 1997.
- [86] Z. Bern, L. J. Dixon, and D. Kosower, “A Two loop four gluon helicity amplitude in QCD,” *JHEP*, vol. 0001, p. 027, 2000.

-
- [87] D. A. Kosower and K. J. Larsen, “Maximal Unitarity at Two Loops,” *Phys.Rev.*, vol. D85, p. 045017, 2012.
- [88] K. J. Larsen, “Global Poles of the Two-Loop Six-Point N=4 SYM integrand,” *Phys.Rev.*, vol. D86, p. 085032, 2012.
- [89] S. Caron-Huot and K. J. Larsen, “Uniqueness of two-loop master contours,” *JHEP*, vol. 1210, p. 026, 2012.
- [90] H. Johansson, D. A. Kosower, and K. J. Larsen, “Two-Loop Maximal Unitarity with External Masses,” *Phys.Rev.*, vol. D87, p. 025030, 2013.
- [91] P. Mastrolia and G. Ossola, “On the Integrand-Reduction Method for Two-Loop Scattering Amplitudes,” *JHEP*, vol. 1111, p. 014, 2011.
- [92] S. Badger, H. Frellesvig, and Y. Zhang, “Hepta-Cuts of Two-Loop Scattering Amplitudes,” *JHEP*, vol. 1204, p. 055, 2012.
- [93] Y. Zhang, “Integrand-Level Reduction of Loop Amplitudes by Computational Algebraic Geometry Methods,” *JHEP*, vol. 1209, p. 042, 2012.
- [94] P. Mastrolia, E. Mirabella, G. Ossola, and T. Peraro, “Scattering Amplitudes from Multivariate Polynomial Division,” *Phys.Lett.*, vol. B718, pp. 173–177, 2012.
- [95] P. Mastrolia, E. Mirabella, G. Ossola, and T. Peraro, “Multiloop Integrand Reduction for Dimensionally Regulated Amplitudes,” *Phys.Lett.*, vol. B727, pp. 532–535, 2013.
- [96] P. Mastrolia, E. Mirabella, and T. Peraro, “Integrand reduction of one-loop scattering amplitudes through Laurent series expansion,” *JHEP*, vol. 1206, p. 095, 2012.
- [97] T. Peraro, “Ninja: Automated Integrand Reduction via Laurent Expansion for One-Loop Amplitudes,” 2014.
- [98] G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, *et al.*, “GoSam-2.0: a tool for automated one-loop calculations within the Standard Model and beyond,” 2014.
- [99] H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, *et al.*, “Multi-leg One-loop Massive Amplitudes from Integrand Reduction via Laurent Expansion,” 2013.
- [100] H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, *et al.*, “NLO QCD corrections to Higgs boson production in association with a top quark pair and a jet,” *Phys.Rev.Lett.*, vol. 111, p. 171801, 2013.

-
- [101] J. Butterworth, G. Dissertori, S. Dittmaier, D. de Florian, N. Glover, *et al.*, “Les Houches 2013: Physics at TeV Colliders: Standard Model Working Group Report,” 2014.
- [102] H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, *et al.*, “NLO QCD corrections to the production of Higgs plus two jets at the LHC,” *Phys.Lett.*, vol. B721, pp. 74–81, 2013.
- [103] G. Cullen, H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, *et al.*, “NLO QCD corrections to Higgs boson production plus three jets in gluon fusion,” *Phys.Rev.Lett.*, vol. 111, p. 131801, 2013.
- [104] M. E. Peskin and D. V. Schroeder, “An Introduction to quantum field theory,” 1995.
- [105] C. Becchi and G. Ridolfi, “An introduction to relativistic processes and the standard model of electroweak interactions,” 2006.
- [106] R. K. Ellis, W. J. Stirling, and B. Webber, “QCD and collider physics,” *Camb.Monogr.Part.Phys.Nucl.Phys.Cosmol.*, vol. 8, pp. 1–435, 1996.
- [107] T. Cheng and L. Li, “GAUGE THEORY OF ELEMENTARY PARTICLE PHYSICS,” 1984.
- [108] G. Aad *et al.*, “Measurements of Higgs boson production and couplings in diboson final states with the ATLAS detector at the LHC,” *Phys.Lett.*, vol. B726, pp. 88–119, 2013.
- [109] T. Aaltonen *et al.*, “Higgs Boson Studies at the Tevatron,” *Phys.Rev.*, vol. D88, p. 052014, 2013.
- [110] S. Heinemeyer *et al.*, “Handbook of LHC Higgs Cross Sections: 3. Higgs Properties,” 2013.
- [111] S. Catani and M. Seymour, “A General algorithm for calculating jet cross-sections in NLO QCD,” *Nucl.Phys.*, vol. B485, pp. 291–419, 1997.
- [112] J. M. Campbell, M. Cullen, and E. N. Glover, “Four jet event shapes in electron - positron annihilation,” *Eur.Phys.J.*, vol. C9, pp. 245–265, 1999.
- [113] D. A. Kosower, “Antenna factorization of gauge theory amplitudes,” *Phys.Rev.*, vol. D57, pp. 5410–5416, 1998.
- [114] S. Frixione, Z. Kunszt, and A. Signer, “Three jet cross-sections to next-to-leading order,” *Nucl.Phys.*, vol. B467, pp. 399–442, 1996.
- [115] S. Frixione, “A General approach to jet cross-sections in QCD,” *Nucl.Phys.*, vol. B507, pp. 295–314, 1997.

- [116] M. Cacciari, G. P. Salam, and G. Soyez, “The Anti-k(t) jet clustering algorithm,” *JHEP*, vol. 0804, p. 063, 2008.
- [117] F. A. Berends and W. Giele, “Recursive Calculations for Processes with n Gluons,” *Nucl.Phys.*, vol. B306, p. 759, 1988.
- [118] L. J. Dixon, “Calculating scattering amplitudes efficiently,” 1996.
- [119] M. E. Peskin, “Simplifying Multi-Jet QCD Computation,” 2011.
- [120] A. Signer and D. Stockinger, “Using Dimensional Reduction for Hadronic Collisions,” *Nucl.Phys.*, vol. B808, pp. 88–120, 2009.
- [121] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, “SINGULAR 3-1-6 — A computer algebra system for polynomial computations.” <http://www.singular.uni-kl.de>, 2012.
- [122] D. R. Grayson and M. E. Stillman, “Macaulay2, a software system for research in algebraic geometry.” Available at <http://www.math.uiuc.edu/Macaulay2/>.
- [123] M. Sogaard and Y. Zhang, “Unitarity Cuts of Integrals with Doubled Propagators,” 2014.
- [124] J. Vermaseren, “New features of FORM,” 2000.
- [125] R. K. Ellis and G. Zanderighi, “Scalar one-loop integrals for QCD,” *JHEP*, vol. 0802, p. 002, 2008.
- [126] F. del Aguila and R. Pittau, “Recursive numerical calculus of one-loop tensor integrals,” *JHEP*, vol. 0407, p. 017, 2004.
- [127] K. Melnikov and M. Schulze, “NLO QCD corrections to top quark pair production in association with one hard jet at hadron colliders,” *Nucl.Phys.*, vol. B840, pp. 129–159, 2010.
- [128] P. Mastrolia, G. Ossola, C. Papadopoulos, and R. Pittau, “Optimizing the Reduction of One-Loop Amplitudes,” *JHEP*, vol. 0806, p. 030, 2008.
- [129] A. van Hameren, “OneLOop: For the evaluation of one-loop scalar functions,” *Comput.Phys.Commun.*, vol. 182, pp. 2427–2438, 2011.
- [130] R. G. Stuart, “Algebraic Reduction of One Loop Feynman Diagrams to Scalar Integrals,” *Comput.Phys.Commun.*, vol. 48, pp. 367–389, 1988.
- [131] T. Binoth, F. Boudjema, G. Dissertori, A. Lazopoulos, A. Denner, *et al.*, “A Proposal for a standard interface between Monte Carlo tools and one-loop programs,” *Comput.Phys.Commun.*, vol. 181, pp. 1612–1622, 2010.

-
- [132] S. Alioli, S. Badger, J. Bellm, B. Biedermann, F. Boudjema, *et al.*, “Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and one-loop programs,” *Comput.Phys.Commun.*, vol. 185, pp. 560–571, 2014.
- [133] P. Nogueira, “Automatic Feynman graph generation,” *J.Comput.Phys.*, vol. 105, pp. 279–289, 1993.
- [134] G. Cullen, M. Koch-Janusz, and T. Reiter, “Spinney: A Form Library for Helicity Spinors,” *Comput.Phys.Communic.*, vol. 182, pp. 2368–2387, 2011.
- [135] T. Reiter, “Optimising Code Generation with haggies,” *Comput.Phys.Communic.*, vol. 181, pp. 1301–1331, 2010.
- [136] J. Kuipers, T. Ueda, J. Vermaseren, and J. Vollinga, “FORM version 4.0,” *Comput.Phys.Communic.*, vol. 184, pp. 1453–1467, 2013.
- [137] J. Kuipers, T. Ueda, and J. Vermaseren, “Code Optimization in FORM,” 2013.
- [138] H. van Deurzen, “Associated Higgs Production at NLO with GoSam,” *Acta Phys.Polon.*, vol. B44, no. 11, pp. 2223–2230, 2013.
- [139] T. Binoth, J.-P. Guillet, G. Heinrich, E. Pilon, and T. Reiter, “Golem95: A Numerical program to calculate one-loop tensor integrals with up to six external legs,” *Comput.Phys.Communic.*, vol. 180, pp. 2317–2330, 2009.
- [140] G. Cullen, J. P. Guillet, G. Heinrich, T. Kleinschmidt, E. Pilon, *et al.*, “Golem95C: A library for one-loop integrals with complex masses,” *Comput.Phys.Communic.*, vol. 182, pp. 2276–2284, 2011.
- [141] J. P. Guillet, G. Heinrich, and J. von Soden-Fraunhofen, “Tools for NLO automation: extension of the golem95C integral library,” 2013.
- [142] S. Catani, S. Dittmaier, and Z. Trocsanyi, “One loop singular behavior of QCD and SUSY QCD amplitudes with massive partons,” *Phys.Lett.*, vol. B500, pp. 149–160, 2001.
- [143] R. Kleiss, W. J. Stirling, and S. Ellis, “A New Monte Carlo Treatment of Multiparticle Phase Space at High-energies,” *Comput.Phys.Communic.*, vol. 40, p. 359, 1986.
- [144] R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, R. Pittau, *et al.*, “Scalar and pseudoscalar Higgs production in association with a top-antitop pair,” *Phys.Lett.*, vol. B701, pp. 427–433, 2011.
- [145] C. Degrande, J. Gerard, C. Grojean, F. Maltoni, and G. Servant, “Probing Top-Higgs Non-Standard Interactions at the LHC,” *JHEP*, vol. 1207, p. 036, 2012.

- [146] P. Artoisenet, P. de Aquino, F. Maltoni, and O. Mattelaer, “Unravelling $t\bar{t}h$ via the Matrix Element Method,” *Phys.Rev.Lett.*, vol. 111, no. 9, p. 091802, 2013.
- [147] W. Beenakker, S. Dittmaier, M. Kramer, B. Plumper, M. Spira, *et al.*, “Higgs radiation off top quarks at the Tevatron and the LHC,” *Phys.Rev.Lett.*, vol. 87, p. 201805, 2001.
- [148] W. Beenakker, S. Dittmaier, M. Kramer, B. Plumper, M. Spira, *et al.*, “NLO QCD corrections to t anti- t H production in hadron collisions,” *Nucl.Phys.*, vol. B653, pp. 151–203, 2003.
- [149] S. Dawson, L. Orr, L. Reina, and D. Wackerroth, “Associated top quark Higgs boson production at the LHC,” *Phys.Rev.*, vol. D67, p. 071503, 2003.
- [150] S. Dawson, C. Jackson, L. Orr, L. Reina, and D. Wackerroth, “Associated Higgs production with top quarks at the large hadron collider: NLO QCD corrections,” *Phys.Rev.*, vol. D68, p. 034022, 2003.
- [151] S. Dittmaier, M. Kramer, Michael, and M. Spira, “Higgs radiation off bottom quarks at the Tevatron and the CERN LHC,” *Phys.Rev.*, vol. D70, p. 074010, 2004.
- [152] T. Plehn, G. P. Salam, and M. Spannowsky, “Fat Jets for a Light Higgs,” *Phys.Rev.Lett.*, vol. 104, p. 111801, 2010.
- [153] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, *et al.*, “Event generation with SHERPA 1.1,” *JHEP*, vol. 0902, p. 007, 2009.
- [154] F. Krauss, R. Kuhn, and G. Soff, “AMEGIC++ 1.0: A Matrix element generator in C++,” *JHEP*, vol. 0202, p. 044, 2002.
- [155] S. Catani, S. Dittmaier, M. H. Seymour, and Z. Trocsanyi, “The Dipole formalism for next-to-leading order QCD calculations with massive partons,” *Nucl.Phys.*, vol. B627, pp. 189–265, 2002.
- [156] T. Gleisberg and F. Krauss, “Automating dipole subtraction for QCD NLO calculations,” *Eur.Phys.J.*, vol. C53, pp. 501–523, 2008.
- [157] M. Cacciari and G. P. Salam, “Dispelling the N^3 myth for the k_t jet-finder,” *Phys.Lett.*, vol. B641, pp. 57–61, 2006.
- [158] M. Cacciari, G. P. Salam, and G. Soyez, “FastJet User Manual,” *Eur.Phys.J.*, vol. C72, p. 1896, 2012.
- [159] J. Pumplin, D. Stump, J. Huston, H. Lai, P. M. Nadolsky, *et al.*, “New generation of parton distributions with uncertainties from global QCD analysis,” *JHEP*, vol. 0207, p. 012, 2002.

- [160] H.-L. Lai, M. Guzzi, J. Huston, Z. Li, P. M. Nadolsky, *et al.*, “New parton distributions for collider physics,” *Phys.Rev.*, vol. D82, p. 074024, 2010.
- [161] J. M. Butterworth, A. R. Davison, M. Rubin, and G. P. Salam, “Jet substructure as a new Higgs search channel at the LHC,” *Phys.Rev.Lett.*, vol. 100, p. 242001, 2008.
- [162] V. Del Duca, W. Kilgore, C. Oleari, C. Schmidt, and D. Zeppenfeld, “Higgs + 2 jets via gluon fusion,” *Phys.Rev.Lett.*, vol. 87, p. 122001, 2001.
- [163] V. Del Duca, W. Kilgore, C. Oleari, C. Schmidt, and D. Zeppenfeld, “Gluon fusion contributions to H + 2 jet production,” *Nucl.Phys.*, vol. B616, pp. 367–399, 2001.
- [164] F. Campanario and M. Kubocz, “Higgs boson production in association with three jets via gluon fusion at the LHC: Gluonic contributions,” *Phys.Rev.*, vol. D88, no. 5, p. 054021, 2013.
- [165] J. M. Campbell, R. K. Ellis, and G. Zanderighi, “Next-to-Leading order Higgs + 2 jet production via gluon fusion,” *JHEP*, vol. 0610, p. 028, 2006.
- [166] J. M. Campbell, R. K. Ellis, and C. Williams, “Hadronic production of a Higgs boson and two jets at next-to-leading order,” *Phys.Rev.*, vol. D81, p. 074023, 2010.
- [167] V. Del Duca, A. Frizzo, and F. Maltoni, “Higgs boson production in association with three jets,” *JHEP*, vol. 0405, p. 064, 2004.
- [168] L. J. Dixon, E. N. Glover, and V. V. Khoze, “MHV rules for Higgs plus multi-gluon amplitudes,” *JHEP*, vol. 0412, p. 015, 2004.
- [169] S. Badger, E. N. Glover, and V. V. Khoze, “MHV rules for Higgs plus multi-parton amplitudes,” *JHEP*, vol. 0503, p. 023, 2005.
- [170] R. K. Ellis, W. Giele, and G. Zanderighi, “Virtual QCD corrections to Higgs boson plus four parton processes,” *Phys.Rev.*, vol. D72, p. 054018, 2005.
- [171] R. K. Ellis, W. Giele, and G. Zanderighi, “Semi-numerical evaluation of one-loop corrections,” *Phys.Rev.*, vol. D73, p. 014027, 2006.
- [172] C. F. Berger, V. Del Duca, and L. J. Dixon, “Recursive Construction of Higgs-Plus-Multiparton Loop Amplitudes: The Last of the Phi-nite Loop Amplitudes,” *Phys.Rev.*, vol. D74, p. 094021, 2006.
- [173] S. Badger and E. N. Glover, “One-loop helicity amplitudes for H \rightarrow ; gluons: The All-minus configuration,” *Nucl.Phys.Proc.Suppl.*, vol. 160, pp. 71–75, 2006.
- [174] S. Badger, E. N. Glover, and K. Risager, “One-loop phi-MHV amplitudes using the unitarity bootstrap,” *JHEP*, vol. 0707, p. 066, 2007.

- [175] E. N. Glover, P. Mastrolia, and C. Williams, “One-loop phi-MHV amplitudes using the unitarity bootstrap: The General helicity case,” *JHEP*, vol. 0808, p. 017, 2008.
- [176] S. Badger, E. Nigel Glover, P. Mastrolia, and C. Williams, “One-loop Higgs plus four gluon amplitudes: Full analytic results,” *JHEP*, vol. 1001, p. 036, 2010.
- [177] L. J. Dixon and Y. Sofianatos, “Analytic one-loop amplitudes for a Higgs boson plus four partons,” *JHEP*, vol. 0908, p. 058, 2009.
- [178] S. Badger, J. M. Campbell, R. K. Ellis, and C. Williams, “Analytic results for the one-loop NMHV Hqgg amplitude,” *JHEP*, vol. 0912, p. 035, 2009.
- [179] T. Stelzer and W. Long, “Automatic generation of tree level helicity amplitudes,” *Comput.Phys.Commun.*, vol. 81, pp. 357–371, 1994.
- [180] J. Alwall, P. Demin, S. de Visscher, R. Frederix, M. Herquet, *et al.*, “MadGraph/MadEvent v4: The New Web Generation,” *JHEP*, vol. 0709, p. 028, 2007.
- [181] R. Frederix, T. Gehrmann, and N. Greiner, “Automation of the Dipole Subtraction Method in MadGraph/MadEvent,” *JHEP*, vol. 0809, p. 122, 2008.
- [182] R. Frederix, T. Gehrmann, and N. Greiner, “Integrated dipoles with MadDipole in the MadGraph framework,” *JHEP*, vol. 1006, p. 086, 2010.
- [183] F. Maltoni and T. Stelzer, “MadEvent: Automatic event generation with MadGraph,” *JHEP*, vol. 0302, p. 027, 2003.
- [184] F. Wilczek, “Decays of Heavy Vector Mesons Into Higgs Particles,” *Phys.Rev.Lett.*, vol. 39, p. 1304, 1977.
- [185] A. Djouadi, M. Spira, and P. Zerwas, “Production of Higgs bosons in proton colliders: QCD corrections,” *Phys.Lett.*, vol. B264, pp. 440–446, 1991.
- [186] S. Dawson, “Radiative corrections to Higgs boson production,” *Nucl.Phys.*, vol. B359, pp. 283–300, 1991.
- [187] P. Mastrolia, E. Mirabella, G. Ossola, and T. Peraro, “Integrand-Reduction for Two-Loop Scattering Amplitudes through Multivariate Polynomial Division,” *Phys.Rev.*, vol. D87, no. 8, p. 085026, 2013.
- [188] J. J. Carrasco and H. Johansson, “Five-Point Amplitudes in N=4 Super-Yang-Mills Theory and N=8 Supergravity,” *Phys.Rev.*, vol. D85, p. 025006, 2012.
- [189] D. J. Broadhurst, J. Fleischer, and O. Tarasov, “Two loop two point functions with masses: Asymptotic expansions and Taylor series, in any dimension,” *Z.Phys.*, vol. C60, pp. 287–302, 1993.

-
- [190] C. Anastasiou and K. Melnikov, “Higgs boson production at hadron colliders in NNLO QCD,” *Nucl.Phys.*, vol. B646, pp. 220–256, 2002.
- [191] C. Anastasiou and A. Lazopoulos, “Automatic integral reduction for higher order perturbative calculations,” *JHEP*, vol. 0407, p. 046, 2004.
- [192] A. Smirnov, “Algorithm FIRE – Feynman Integral REduction,” *JHEP*, vol. 0810, p. 107, 2008.
- [193] C. Studerus, “Reduze-Feynman Integral Reduction in C++,” *Comput.Phys.Commun.*, vol. 181, pp. 1293–1300, 2010.
- [194] A. von Manteuffel and C. Studerus, “Reduze 2 - Distributed Feynman Integral Reduction,” 2012.
- [195] S. Badger, H. Frellesvig, and Y. Zhang, “An Integrand Reconstruction Method for Three-Loop Amplitudes,” *JHEP*, vol. 1208, p. 065, 2012.
- [196] S. Badger, H. Frellesvig, and Y. Zhang, “A Two-Loop Five-Gluon Helicity Amplitude in QCD,” *JHEP*, vol. 1312, p. 045, 2013.
- [197] E. Remiddi and L. Tancredi, “Schouten identities for Feynman graph amplitudes; The Master Integrals for the two-loop massive sunrise graph,” *Nucl.Phys.*, vol. B880, pp. 343–377, 2014.
- [198] Z. Bern, A. De Freitas, and L. J. Dixon, “Two loop helicity amplitudes for gluon-gluon scattering in QCD and supersymmetric Yang-Mills theory,” *JHEP*, vol. 0203, p. 018, 2002.
- [199] Z. Bern and G. Chalmers, “Factorization in one loop gauge theory,” *Nucl.Phys.*, vol. B447, pp. 465–518, 1995.
- [200] D. Cox, J. Little, and D. O’Shea, *Using Algebraic Geometry*. Graduate texts in mathematics, Springer, 1998.
- [201] D. A. Cox, J. Little, and D. O’Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.
- [202] G. Ossola, C. G. Papadopoulos, and R. Pittau, “Numerical evaluation of six-photon amplitudes,” *JHEP*, vol. 0707, p. 085, 2007.
- [203] G. Gounaris, P. Porfyriadis, and F. Renard, “The gamma gamma \rightarrow gamma gamma process in the standard and SUSY models at high-energies,” *Eur.Phys.J.*, vol. C9, pp. 673–686, 1999.

-
- [204] C. Bernicot, “Light-light amplitude from generalized unitarity in massive QED,” 2008.
- [205] G. Mahlon, “One loop multi - photon helicity amplitudes,” *Phys.Rev.*, vol. D49, pp. 2197–2210, 1994.
- [206] Z. Nagy and D. E. Soper, “Numerical integration of one-loop Feynman diagrams for N-photon amplitudes,” *Phys.Rev.*, vol. D74, p. 093006, 2006.
- [207] T. Binoth, G. Heinrich, T. Gehrmann, and P. Mastrolia, “Six-Photon Amplitudes,” *Phys.Lett.*, vol. B649, pp. 422–426, 2007.
- [208] W. Gong, Z. Nagy, and D. E. Soper, “Direct numerical integration of one-loop Feynman diagrams for N-photon amplitudes,” *Phys.Rev.*, vol. D79, p. 033005, 2009.
- [209] C. Bernicot and J.-P. Guillet, “Six-Photon Amplitudes in Scalar QED,” *JHEP*, vol. 0801, p. 059, 2008.
- [210] C. Bernicot, “The six-photon amplitude,” 2008.
- [211] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, *et al.*, “Next-to-Leading Order QCD Predictions for W+3-Jet Distributions at Hadron Colliders,” *Phys.Rev.*, vol. D80, p. 074036, 2009.
- [212] C. Berger, Z. Bern, L. J. Dixon, F. Febres Cordero, D. Forde, *et al.*, “Next-to-Leading Order QCD Predictions for $Z, \gamma^* + 3$ -Jet Distributions at the Tevatron,” *Phys.Rev.*, vol. D82, p. 074002, 2010.
- [213] G. Bevilacqua, M. Czakon, C. Papadopoulos, and M. Worek, “Hadronic top-quark pair production in association with two jets at Next-to-Leading Order QCD,” *Phys.Rev.*, vol. D84, p. 114017, 2011.

Acknowledgments

I wish to thank my supervisor Pierpaolo Mastrolia, which gave me the opportunity to work with him, as well as Edoardo Mirabella and Giovanni Ossola with whom I collaborated for a large part of the work presented in this thesis. I also thank all the other members of the GOSAM collaboration for the common development of a one-loop package which could be interfaced with the NINJA library, and Thomas Hahn for his support with the library LOOPTOOLS and several suggestions.

I am grateful to all the members of the Max-Planck-Institut für Physik in München, where I had a very pleasant time during my Ph.D. studies. I am indebted to my office-mates Davide Pagani, Sophia Borowka and Johannes Schlenk, for innumerable useful discussions and suggestions.

Finally, I wish to thank my family for giving me the opportunity to study and for all their support.

The work presented in this thesis was supported by the Alexander von Humboldt Foundation, in the framework of the Sofja Kovalevskaja Award Project “Advanced Mathematical Methods for Particle Physics”, endowed by the German Federal Ministry of Education and Research.