Higher Order Corrections in Perturbative Quantum Field Theory via Sector Decomposition

Jonathon Carter

A Thesis presented for the degree of
Doctor of Philosophy

Institute for Particle Physics Phenomenology
University of Durham
England

October 2011

Higher Order Corrections in Perturbative Quantum Field Theory via Sector Decomposition

Jonathon Carter

Submitted for the degree of Doctor of Philosophy
October 2011

Abstract

The calculation of higher order corrections in perturbative quantum field theories is a particularly important subject. Our current model for particle physics is the standard model; a quantum field theory which has served to describe a huge amount of observed data very well. As the Large Hadron Collider is collecting more and more high energy data with smaller and smaller experimental errors, the accuracy of theoretical calculations must keep up with experiment in order to discriminate between physics arising from our current standard model, and beyond standard model physics.

In chapter 2 we give a brief introduction to the fundamentals of perturbative quantum field theories, with particular emphasis on Quantum ChromoDynamics, where higher order calculations are particularly important due to the fact that $\alpha_s(M_Z) \gg \alpha$. In chapter 3 we present a review of methods for calculations within perturbative quantum field theories, both for real and virtual corrections. In chapter 4 we give a detailed explanation of the method of sector decomposition, and highlight how it can be applied to the calculation of multi-parameter polynomial integrals, which appear widely in high energy physics, and in particular within the higher order calculations of perturbative quantum field theories. In chapter 5 we present SecDec - a publicly available computer code which implements sector decomposition. We give a range of examples to demonstrate its power in calculating various integrals appearing in higher order calculations in perturbative quantum field theories.

Declaration

The work in this thesis is based on research carried out at the Institute for Particle Physics Phenomenology, the Department of Physics, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

Firstly I would like to thank my supervisor Gudrun Heinrich for her support and collaboration over the past four years. Even from her new position at the Max Planck Institute in Munich she always had time to give invaluable guidance. Thanks also to Nigel Glover, who became my supervisor after gudrun left the IPPP, and has had to deal with the tiresome paperwork a PhD student generates...

I would also like to thank everybody at the IPPP for providing such a great atmosphere to work in. Particular thanks go to Linda Wilkinson and Trudy Forster for their help in organising many research trips throughout my time here, and to the various IT staff we have had over my time here - Phil Roffe, David Ambrose-Griffith and Mike Johnson, as well as David Grellscheid and Daniel Maître who provided expert advise on computing issues.

Thanks to my family, who have always encouraged me to do what makes me happy.

Finally, thanks to Katherine, who has been there for me throughout my time here, and has helped to keep me sane whilst I have been writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For millennia people have wondered what the world around us is made of. From the atomic theory of Democritus, through the discovery of the electron[1] by Thomson, to the vast leaps of understanding of the 20$^{\text{th}}$ century of quantum mechanics and general relativity. The 'particle zoo'[2] has gone through a huge upheaval over the last century, starting with just the electron, with the proton added by Rutherford, then the neutron, positron, (anti-)muon, followed by a string of mesons and baryons (pions, kaons, deltas,...), and the electron and muon neutrinos. In 1954, Yang and Mills developed the concept of local gauge theories, which were to become the basis of the Standard Model (SM) of particle physics. The zoo went through a cull in the 1960s, when Gell-Mann and Zweig put forward the quark model to explain the proliferation of hadrons, and also the fact that deep inelastic scattering experiments had hinted that these particles had internal structure. In 1964 Higgs proposed his eponymous mechanism of spontaneous symmetry breaking. In 1967 Weinberg and Salam independently put forward the Electro-Weak theory, which required the existence of both a neutral and a charged, weakly interacting vector boson, and the massive, scalar Higgs boson. The $W^{\pm}$, Z vector bosons of this theory were discovered in the 1980s at CERN, and the last of the known quarks, the top quark, was

[1] Thomson toasted the discovery with the words,"To the electron - may it never be of any use to anybody." A classic example of how advances in our understanding of the world can lead to applications far beyond our imagination.

[2] List of all known fundamental particles.

discovered at the Tevatron in 1995.

Our current description of particle physics explains a vast amount of experimental data very well, but there are still many questions which need answering. For example, astrophysical observations suggest the existence of *dark matter*[3], that is matter which does not emit or scatter photons but does interact via gravity. The SM has no explanation for this. Nor can the SM describe the gravitational interaction. The matter/antimatter asymmetry is also not predicted by the SM. Moreover, the postulated Higgs boson which completes the theory has not yet been discovered.

Now it is the turn of the Large Hadron Collider to attempt to tackle these problems. The LHC is colliding protons at a centre of mass energy $\sqrt{s} = 7$ TeV, with plans to increase this to ~ 14 TeV in a few years time. This will allow us to either discover physics beyond the standard model (BSM physics), or rule out particular BSM models. The most well known goal of the LHC is to discover the origin of Electro-Weak symmetry breaking, be it the SM Higgs, or something from a BSM theory (SuperSymmetric Higgs particle(s), Technicolour particles,...). The idea of SuperSymmetry in its various forms is being probed, and certain models are already being ruled out by LHC data. Further theories which postulate extra dimensions predict the creation and subsequent decay of microscopic black holes at the LHC.

No matter what the BSM theory being probed, without a good understanding of the SM processes which provide a background for BSM signals we can infer very little from the data. The difficulty of the calculations behind this understanding of the background increases dramatically with the level of accuracy required. Moreover, if the BSM model is described as a Quantum Field Theory then all the mechanisms used to calculate SM backgrounds can be employed to calculate BSM processes. Thus the development and, ideally, automation of higher order calculations within perturbative QFTs is crucial for us to advance our understanding of high energy physics.

[3]With about 4 times as much dark matter than regular matter

1.1 Outline

The structure of this thesis is as follows: In chapter 2 we will discuss the background of how to perform calculations within a perturbative QFT, with reference in particular to the strong interaction (Quantum ChromoDynamics) as it appears in the SM. We will outline how these theories contain certain divergences, and how these divergences can be treated in order to obtain finite results for physical observables. We will consider the validity of perturbative calculations, and how non-perturbative effects are dealt with for hadronic initial/final states in a scattering process.
In chapter 3 we will go into more detail about how calculations at higher orders in the perturbative expansion can be evaluated using a number of methods. In chapter 4, we will introduce the powerful method of sector decomposition, and describe how it can be used for both real and virtual radiative corrections to perturbative QFTs. In particular we will show how its application to beyond next-to-leading order calculations is straightforward. In chapter 5 we will present the program SecDec, and illustrate its wide range of applicability through a number of examples. We will then present our conclusions and outlook in chapter 6.

Chapter 2

Perturbative Quantum Field Theories

In this chapter I will discuss a number of features which occur generically in Quantum Field Theories (QFT). I will mainly use Quantum ChromoDynamics (QCD) to illustrate these, as it contains all of the most complicated features that can arise. This section provides a brief overview of a number of topics within perturbative QFT. For more detailed explanations see eg [1–4].

2.1 QCD Particle Content

The fundamental particles of QCD are the quarks (u, d, c, s, t, b), together with their anti-particles, and the gluon (g). Due to confinement these are never observed as free particles in nature. Instead these particles group together as baryons (3 quark bound states) and mesons (quark anti-quark bound states). The theory is invariant under local $SU(3)_{Colour}$ gauge transformations. Each individual quark has one of three colours associated with it (r, b, g), as the quarks transform under the fundamental representation, and only colour-neutral bound states have been observed in nature, so each quark within a baryon has a different colour (this fact helps us overcome the apparent violation of the Pauli Exclusion Principle), and a meson is made of a superposition of $\frac{1}{\sqrt{3}}\left(r\bar{r} + b\bar{b} + g\bar{g}\right)$. The gluon colour index runs from $1 \to 8$, as the gluon transforms under the adjoint representation of $SU(3)_{Colour}$.

2.2 Lie Groups, Algebras and Representations

Let us define a few terms which will be useful in the following discussion.

A *Lie group* $\mathcal{G}$ is a continuous group, where every infinitesimal group element can be written as

$$g(x) = 1 + i\alpha^a T^a + O(\alpha^2)$$

where the α^a are infinitesimal parameters, and T^a are the *generators* of $\mathcal{G}$

The generators span the set of infinitesimal group transformations, and so the commutator of generators can be written as

$$\left[T^a, T^b\right] = if^{abc}T^c$$

where the f^{abc} are called the *structure constants*.

A *Lie algebra* $\mathcal{L}$ is a vector space V, together with a bilinear operator

$$[\cdot, \cdot] : V \times V \to V$$

such that

$$[x, x] = 0,$$

and the *Jacobi identity*:

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0 \quad \forall x, y, z \in V$$

The vector space spanned by the generators of the Lie group, together with the commutator, form a Lie algebra. This is known as the *fundamental representation*. The structure constants also form a representation of the Lie algebra, namely the *adjoint representation*, via

$$\left(t^b\right)_{ac} \equiv if^{abc}$$

since these objects satisfy the commutation relation

$$\begin{aligned} \left(\left[t^b, t^c\right]\right)_{ae} &= -\left(f^{abd}f^{dce} - f^{acd}f^{dbe}\right) = f^{cde}f^{abd} + f^{bde}f^{cad} \\ &= -f^{ade}f^{bcd} = if^{bcd}\left(t^d\right)_{ae} \end{aligned}$$

where we have used the Jacobi identity.

Following on from the above, we consider the Lie group $SU(3)$. The generators

of $SU(3)$ are the 3×3 generalisations of the Pauli σ matrices. They are 8 traceless, Hermitian 3×3 matrices, referred to as the Gell-Mann matrices. The adjoint representation is made up of 8 traceless, Hermitian 8×8 matrices.

2.3 QCD Lagrangian

QCD is described by the Lagrangian density (henceforth referred to as just 'Lagrangian')

$$\mathcal{L}_{QCD} = \sum_f \bar{\psi}_{f,i} \left(i \not{D}_{ij} - m_f \delta_{ij} \right) \psi_{f,j} - \frac{1}{4} F_a^{\mu\nu} F^a_{\mu\nu} + \mathcal{L}_{gauge-fixing} + \mathcal{L}_{ghost} \tag{2.1}$$

where f runs over all flavours, $i = 1, 2, 3$ is the colour index of the quarks and $a = 1, \cdots, 8$ [1] is the colour index of the gluon. That is to say that quarks live in the fundamental representation, and gluons in the adjoint representation of $SU(3)_C$. The usual Dirac slash notation of $\not{D} = \gamma^\mu D_\mu$ is followed, where the gauge covariant derivative is

$$D^\mu_{ij} = \delta_{ij} \partial^\mu - i g A^\mu_a T^a_{ij} \tag{2.2}$$

where g is the strong coupling, and the gamma matrices satisfy the Clifford algebra

$$\{\gamma^\mu, \gamma^\nu\} = 2 g^{\mu\nu} \tag{2.3}$$

and $\bar{\psi} \equiv \psi^\dagger \gamma^0$.

The T^a are the generators of $SU(3)$ in the fundamental representation, and obey commutation relations

$$\left[T^a, T^b \right] = i f_{abc} T^c \tag{2.4}$$

The field strength $F^a_{\mu\nu}$ is defined by

$$[D_\mu, D_\nu] = i g T^a F^a_{\mu\nu} \tag{2.5}$$

which yields

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu \tag{2.6}$$

[1] as colour indices are raised and lowered with δ_{ab}, we make no distinction between raised and lowered indices. Any repeated index is understood to be summed over

Under the action of an element $U(x)$ of the gauge group $SU(3)$, the fields transform as

$$\psi \rightarrow U(x)\psi \tag{2.7}$$

$$A^\mu \rightarrow U(x)A^\mu U^{-1}(x) + \frac{i}{g}U(x)\partial U^{-1}(x) \tag{2.8}$$

which leads to the covariant derivative to have the behaviour

$$\begin{aligned} D^\mu &\rightarrow U(x)U^{-1}(x)\partial^\mu - igU(x)A^\mu U^{-1}(x) + U(x)\partial^\mu U^{-1}(x) \\ &= U(x)D^\mu U^{-1}(x) \end{aligned} \tag{2.9}$$

and trivially

$$F^{\mu\nu} \rightarrow U(x)F^{\mu\nu}U^{-1}(x) \tag{2.10}$$

and the part of the Lagrangian given explicitly is invariant under this transformation.

2.4 Gauge-fixing and Ghost fields

If two field configurations are related by a gauge transformation, then they are physically indistinguishable. This ambiguity in the theory needs to be dealt with carefully, by means of *gauge-fixing*. To this end, we choose a condition for A^μ to satisfy, so all physically indistinguishable field configurations are represented by exactly one configuration satisfying this condition. The standard way this is done for internal gluons in QCD is to set

$$F(A) \equiv \partial_\mu A^\mu = 0 \tag{2.11}$$

which is referred to[2] as the *Lorentz condition* .

Without an extra term in the Lagrangian, the definition of the gluon propagator is impossible, since the inverse propagator (in momentum space) is given by

$$D^{-1}_{\{a,\mu\},\{b,\nu\}}(p) = i\delta_{ab}\left[p^2 g_{\mu\nu} - p_\mu p_\nu\right] \tag{2.12}$$

[2]Erroneously. This was originally named after Ludvig Lorenz, but due to the proliferation of Hendrik Lorentz's name throughout theoretical physics it has been commonly misaccredited.

which is not invertible. Adding in the term

$$\mathcal{L}_{gauge-fixing} = -\frac{1}{2\xi}\left(\partial^\mu A_\mu\right)^2 \tag{2.13}$$

fixes this, and leads to the propagator given in the next section. A full treatment of the derivation of $\mathcal{L}_{ghost}$ is not given here, see eg [1]. The result is that we must introduce scalar, anti-commuting Grassman fields $\bar{c}, c$ and add the term

$$\mathcal{L}_{ghost} = \partial_\mu \bar{c} D^\mu c \tag{2.14}$$

to the Lagrangian. The ghost fields cancel the unphysical gluon degrees of freedom. An alternate choice of $F(A)$ in eq. (2.11) is

$$F(A) = n^\mu A_\mu = 0 \tag{2.15}$$

which is referred to as *axial gauge*. This is a useful choice is of gauge, as it can remove the internal ghost particles from the theory. For this reason it is also sometimes referred to as *physical gauge*, as it retains only the physical particle content of the theory. The disadvantage of the axial gauge is that it breaks Lorentz-invariance.

2.5 Feynman Rules

Feynman introduced a diagrammatic way to represent perturbative processes in QFT, which can be derived directly from the Lagrangian of the theory. For perturbative QFT to be a useful tool, the Lagrangian must be of the form

$$\mathcal{L} = \mathcal{L}_{free} + \lambda \mathcal{L}_{int} \tag{2.16}$$

with $\mathcal{L}_{free}$ defining the evolution of particles without interactions, $\mathcal{L}_{int}$ describing the interactions between particles, and λ is a small parameter.

2.5.1 $\lambda\phi^4$ Theory

For a simple example we consider $\lambda\phi^4$ theory.

The matter content is just one real scalar field ϕ with mass m. The Lagrangian is written as

$$\mathcal{L} = \frac{1}{2}\partial^\mu\phi\partial_\mu\phi - \frac{1}{2}m^2\phi^2 - \frac{\lambda}{4!}\phi^4 \tag{2.17}$$

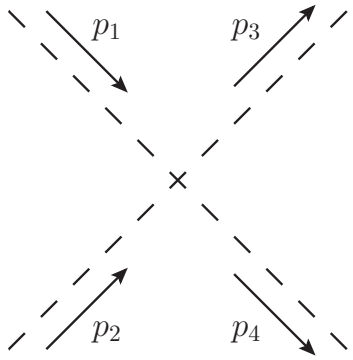
The Feynman propagator is found from the first two terms, by taking the inverse of the operator in momentum space, ie

$$D_F(p) = \frac{i}{p^2 - m^2 + i\delta} \tag{2.18}$$

The interaction vertex comes from the $\frac{\lambda}{4!}\phi^4$ term, and it gives a valence 4 vertex, with a factor of $-i\lambda$.

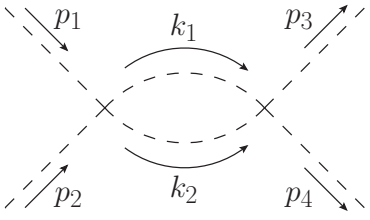
To illustrate how Feynman diagrams work, we will consider the up to $O(\lambda^2)$ contribution to the process $\phi(p_1)\phi(p_2) \to \phi(p_3)\phi(p_4)$. To represent an allowed contribution to this process, the diagram must have 4 external legs, and only valence 4 vertices. It should be noted that d is the dimensionality of space-time, which here we will consider as equal to 4.

The diagrams of figures 2.2 and 2.1 contribute to the scattering amplitude of



$$-i\lambda(2\pi)^d\delta^{(d)}\left(p_1 + p_2 - p_3 - p_4\right)$$

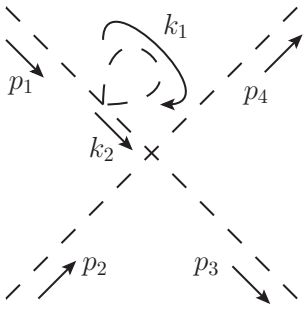
Figure 2.1: $O(\lambda)$ contribution for $\phi\phi \to \phi\phi$



$$\begin{aligned}(-i\lambda)^2 \quad \int \quad & \frac{d^dk_1d^dk_2}{(2\pi)^{2d}}\frac{i^2}{(k_1^2 - m^2 + i\delta)(k_2^2 - m^2 + i\delta)} \\ \times \quad & (2\pi)^d\delta^{(d)}\left(p_1 + p_2 - k_1 - k_2\right) \\ \times \quad & (2\pi)^d\delta^{(d)}\left(k_1 + k_2 - p_3 - p_4\right)\end{aligned}$$

Figure 2.2: $O(\lambda^2)$ contribution for $\phi\phi \to \phi\phi$

$\phi\phi \to \phi\phi$ at leading order (LO), and next-to-leading order (NLO) in λ respectively. The diagram of figure 2.3 does not contribute to the scattering amplitude - notice that the first δ function gives $k_2 = p_1$, and so the propagator becomes $\frac{i}{p_1^2 - m^2 + i\delta}$



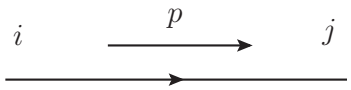
$$(-i\lambda)^2 \int \frac{d^dk_1 d^dk_2}{(2\pi)^{2d}} \frac{i^2}{(k_1^2 - m^2 + i\delta)(k_2^2 - m^2 + i\delta)} \times (2\pi)^d \delta^{(d)} (p_1 + k_1 - k_1 - k_2) \times (2\pi)^d \delta^{(d)} (k_2 + p_2 - p_3 - p_4)$$

Figure 2.3: $O(\lambda^2)$ diagram for $\phi\phi \to \phi\phi$

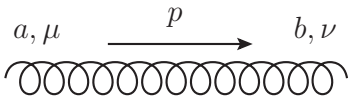
which gives $\frac{1}{0}$. This diagram is considered as a correction term for the external field, and not for the scattering process. For more details on this see section 2.7

2.5.2 QCD

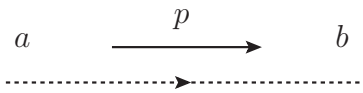
Feynman diagrams for QCD follow the usual convention of quarks represented by a solid line, gluons by a curly line and ghosts by a dotted line. The Feynman rules for QCD (in covariant gauge) are as follows:



$$\delta_{ij} \frac{i(\not{p} + m)}{p^2 - m^2 + i\delta}$$



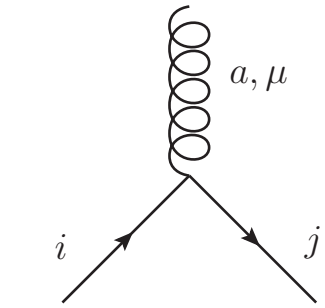
$$-\delta_{ab} \frac{i}{p^2 + i\delta} \left(g^{\mu\nu} - (1 - \xi) \frac{p^\mu p^\nu}{p^2} \right)$$



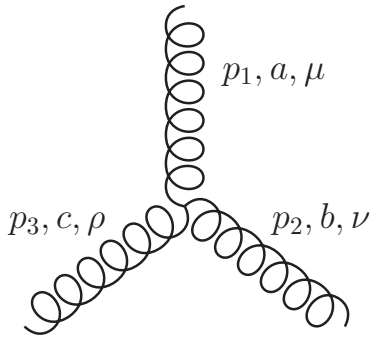
$$\delta_{ab} \frac{i}{p^2 + i\delta}$$

External lines come with spinors/polarisation vectors according to table 2.1

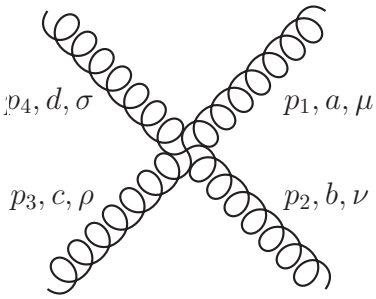
When calculating the square of the amplitude, expressions containing sums over



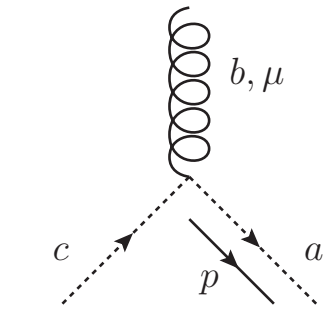
$$ig\gamma^\mu T^a_{ij}$$



$$\begin{aligned} gf^{abc} \quad [\quad & g^{\mu\nu}(p_1 - p_2)^\rho \\ + \quad & g^{\nu\rho}(p_2 - p_3)^\mu \\ + \quad & g^{\rho\mu}(p_3 - p_1)^\nu \ ] \end{aligned}$$



$$\begin{aligned} -ig^2 \quad [\quad & f^{abe}f^{cde}\,(g^{\mu\rho}g^{\nu\sigma} - g^{\mu\sigma}g^{\nu\rho}) \\ + \quad & f^{ace}f^{bde}\,(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\sigma}g^{\nu\rho}) \\ + \quad & f^{ade}f^{bce}\,(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\rho}g^{\nu\sigma}) \ ] \end{aligned}$$



$$-gf^{abc}p^\mu$$

spins/polarisations will occur. Some useful relations for these are

$$\begin{aligned} \sum_s \bar{u}^s(p)u^s(p) &= \not{p} + m \\ \sum_s \bar{v}^s(p)v^s(p) &= \not{p} - m \\ \sum_\lambda \epsilon^{\mu *}_\lambda(p)\epsilon^\nu_\lambda(p) &= -g^{\mu\nu} + \frac{p^\mu n^\nu + p^\nu n^\mu}{p.n} \end{aligned} \tag{2.19}$$

External Particle	Spinor/Polarisation Vector
Incoming quark	$u(p)$
Incoming anti-quark	$\bar{v}(p)$
Incoming gluon	$\epsilon^{\mu}(p)$
Outgoing quark	$\bar{u}(p)$
Outgoing anti-quark	v(p)
Outgoing gluon	$\epsilon^{\mu *}(p)$

Table 2.1: Spinor/Polarisation Vectors for external particles

where we work in axial gauge for external gluons. n^{μ} can be different for each external gluon, and must satisfy $n.p \neq 0$. A good choice of this for incoming gluons with momenta k_1, k_2 is $n_1 = k_2$, $n_2 = k_1$.

The (internal) gluon propagator is dependent on ξ, however the final result must be independent of the choice of gauge-fixing parameter ξ. It is convenient to perform calculations in the *Feynman gauge*, namely ξ=1, which simplifies the propagator as

$$-\delta_{ab}\frac{i}{p^2+i\delta}\left(g^{\mu\nu}-(1-\xi)\frac{p^{\mu}p^{\nu}}{p^2}\right) \rightarrow -\delta_{ab}\frac{ig^{\mu\nu}}{p^2+i\delta} \tag{2.20}$$

Of course any other choice of ξ is equally valid. Two such choices are *Landau gauge* ($\xi = 0$) and *Unitary gauge* ($\xi \rightarrow \infty$).

To calculate the amplitude for a given process, the procedure is to draw every possible diagram which can contribute to the process (at the desired order in the coupling g), and then for each diagram:

1. Include a factor of (-1) for each fermion/ghost loop
2. Impose momentum conservation at each vertex
3. Integrate over unconstrained (loop) momenta with the measure $\frac{d^d k}{(2\pi)^d}$
4. Include a symmetry factor to account for the permutation of identical fields.

2.6 Regularisation

There are two qualitatively different types of divergence found within a pQFT:

- *Ultra-Violet* (UV) divergences arise solely from loop integrals, and relate to the fact that the loop momentum is unbounded.

- *Infra-Red* (IR) divergences come both from loop integrals where one (or more) of the particles in the loop can become soft, or collinear with another particle, and from soft/collinear final state particles as described above.

For a renormalisable theory (see next section), the UV divergences must be removed by absorbing them into definitions of fields and couplings in order to be able to obtain meaningful physical results. In order to be able to do this, these divergences must be *regularised*.
Soft and final state collinear IR divergences cancel when all contributions to a physical observable are combined, by the Kinoshita-Lee-Nauenberg theorem [5,6]. However, intermediate contributions are likely to be divergent. In order to calculate these contributions, the IR divergences must also be regularised.

2.6.1 Pauli-Villars Regularisation

Pauli-Villars regularisation is a procedure to regulate UV divergent loop integrals. The idea is to introduce a fictitious particle with large mass Λ, which cancel the contributions of massless (or mass $m \ll \Lambda$) particles for large loop momenta k, but do not affect the integrand at small k. ie the replacement in a massless propagator of

$$\begin{aligned} \frac{1}{k^2} &\rightarrow \frac{1}{k^2} - \frac{1}{k^2-\Lambda^2} \\ &= \frac{-\Lambda^2}{k^2(k^2-\Lambda^2)} \\ &\sim \begin{cases} \frac{1}{k^2} & \text{for } k^2 \ll \Lambda^2, \\ \frac{-\Lambda^2}{k^4} & \text{for } \Lambda^2 \ll k^2 \end{cases} \end{aligned} \tag{2.21}$$

with the usual $+i\delta$ prescription implied. Pauli-Villars regularisation breaks gauge invariance, and so is not suitable for QCD.

2.6.2 Mass Regularisation

This regularisation method can be used to regulate IR divergences. The method involves giving a small mass to massless particles which induce the IR divergences, and once the full calculation is performed the result will be independent of this mass. The above methods put together can be used to regularise a theory, a good demonstration of these methods for QED can be found in [1]. Giving mass to a gauge boson in this way breaks gauge invariance for non-Abelian gauge groups, and so this is not suitable for QCD.

Here we describe the regularisation scheme that will be used throughout this thesis.

2.6.3 Dimensional Regularisation

Dimensional regularisation (DR) is a regularisation method which regulates both UV and IR singularities. The method takes the dimensionality of space-time, d, is taken to be different from 4, ie

$$d = 4 - 2\epsilon \tag{2.22}$$

DR is a particularly good method, as it respects both Lorentz- and gauge-invariance. All previous definitions have been given for general d, and are still valid. One should, however, take a moment to consider the *mass dimension* (henceforth referred to as just 'dimension') of the fields and parameters within the theory.

Firstly, the action S should be dimensionless, ie $[S] = 0$. Since $S = \int d^d x \mathcal{L}(x)$, we find that $[\mathcal{L}] = d$. By examining terms which appear in the Lagrangian, we find that

$$\begin{aligned} [\partial^\mu A^\nu \partial_\mu A_\nu] &= d \implies [A] = \frac{d-2}{2} \\ [\bar{\psi} \not{\partial} \psi] &= d \implies [\psi] = \frac{d-1}{2} \\ [g \bar{\psi} \not{A} \psi] &= d \implies [g] = \frac{4-d}{2} \end{aligned}$$

We would like to work with a dimensionless parameter g, so we make the substitution

$$g \to \mu_R^\epsilon g \tag{2.23}$$

where $\mu_R = \mu$ is an arbitrary mass scale, which is not fixed a priori. Clearly any physical result should be independent of this scale as it is put in by hand. However,

in practice there will be a dependence on this scale for perturbatively calculated quantities, which arises due to the truncation of the perturbative series.

Within DR, there are different choices of dimension for internal/external particles. The *'t Hooft-Veltman scheme* (HV) takes external states to be 4-dimensional, with internal states being of dimension $4-2\epsilon$. *Conventional DR* (CDR) has all states with dimension $4-2\epsilon$. This is more convenient for dealing with IR singularities induced by external particles than HV, and it is the scheme used throughout this thesis. For a recent discussion of regularisation schemes, see [7].

2.7 Renormalisation

We have discussed where divergences arise in perturbative QFT, and how we regularise these divergences. In order to obtain finite cross-sections for physical quantities, we need to remove the UV divergences from the theory in a consistent way. This process is known as *renormalisation*. First we will describe how this procedure is applied to $\lambda\phi^4$ theory, and then extend the argument to QCD.

2.7.1 $\lambda\phi^4$ Theory

Recall eq. (2.17). We rewrite this with subscripts 0 to indicate that these quantities (mass, coupling constant and field) are unrenormalised, *bare* quantities.

$$\mathcal{L} = \frac{1}{2}\partial^\mu\phi_0\partial_\mu\phi_0 - \frac{1}{2}m_0^2\phi_0^2 - \frac{\lambda_0}{4!}\phi_0^4 \tag{2.24}$$

First we want to renormalise the field ϕ_0. To see how this should be done, let's first consider the two-point function for the full theory. Let $|\Omega\rangle$ be the vacuum of the interacting theory. Let us define a complete set of states

$$\mathbf{1} = |\Omega\rangle\langle\Omega| + \sum_\alpha \int \frac{d^dp}{(2\pi)^d}|\alpha_\mathbf{p}\rangle\langle\alpha_\mathbf{p}| \tag{2.25}$$

Inserting this into the definition of the two-point function gives

$$\langle\Omega|\phi(x)\phi(0)|\Omega\rangle = \sum_\alpha \int \frac{d^dp}{(2\pi)^d}\frac{1}{p^2-m_\alpha^2+i\delta}\, e^{-ip\cdot x}\,|\langle\Omega|\phi(0)|\alpha_0\rangle|^2 \tag{2.26}$$

for $x_0 > 0$, where m_α is the energy of the state $|\alpha_0\rangle$, ie the given state in its centre of mass frame. If we just consider the one-particle state $|\alpha\rangle$, we see that this gives us the usual Feynman propagator, but with the physical mass instead of the bare mass, and a multiplicative factor $Z = |\langle\Omega|\phi(0)|\alpha\rangle|^2$. If we rescale the field we can remove this factor of Z, hence we make the substitution $\phi_0 = Z^{1/2}\phi_r$ in eq. (2.24) to get

$$\mathcal{L} = \frac{1}{2}Z\partial^\mu\phi_r\partial_\mu\phi_r - \frac{1}{2}Z\,m_0^2\phi_r^2 - \frac{\lambda_r}{4!}Z^2\phi_r^4 \tag{2.27}$$

The bare mass and coupling still appear, so we remove them by making the definitions

$$\delta_Z = Z - 1 \quad \delta_m = m_0^2 Z - m^2, \quad \delta_\lambda - \lambda_0 Z^2 - \lambda \tag{2.28}$$

where m, λ are the physically measured values of the mass and coupling constant. Rewriting the Lagrangian gives us

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\partial_\mu\phi_r\partial^\mu\phi_r - \frac{1}{2}m^2\phi_r^2 - \frac{\lambda}{4!}\phi_r^4 \\ &+ \frac{\delta_Z}{2}\partial_\mu\phi_r\partial^\mu\phi_r - \frac{1}{2}\delta_m\phi_r^2 - \frac{\delta_\lambda}{4!}\phi_r^4 \end{aligned} \tag{2.29}$$

This leads to the usual Feynman rules, plus *counterterms* given in figure 2.4. We

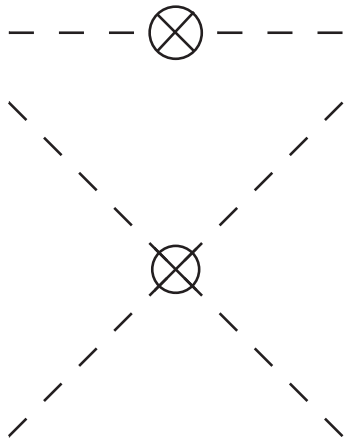
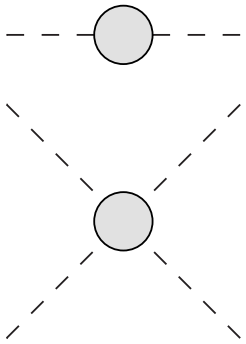


Figure 2.4: Counterterms for ϕ^4 theory

must also give some conditions as to what we mean by physical mass and coupling constant. To do this we fix the *renormalisation conditions* of eq. (2.30), where the second diagram is amputated (ie no leg corrections). These conditions fix $\delta_Z, \delta_m, \delta_\lambda$ up to any given order in perturbation theory, and then calculations of physical



$$= \frac{i}{p^2-m^2} + (\text{terms regular at } p^2 = m^2)$$

$$= -i\lambda \quad \text{at } s = 4m^2,\ t = u = 0 \qquad (2.30)$$

quantities at this order in perturbation theory will be finite.
This method is different in its application to that usually applied to QCD, where it is the bare quantities themselves which are renormalised.

2.7.2 QCD

We will renormalise all the bare fields, masses and coupling constants by replacing them as follows:

$$\begin{aligned} \psi &= Z_\psi^{1/2}\psi_r \\ A &= Z_A^{1/2}A_r \\ c &= Z_c^{1/2}c_r \\ g &= Z_g g_r \\ m &= Z_m m_r \\ \xi &= Z_\xi \xi_r \end{aligned} \qquad (2.31)$$

When calculating a process to a fixed order in perturbation theory, all UV divergences up to this order can be absorbed into the Z factors, and we can make physical predictions which are free of these divergences. How the divergences are absorbed into the Z factors is specified by the singularity structure, but how the finite terms are dealt with is a free choice (providing it is done consistently), and defines the *subtraction scheme*. The simplest scheme is *minimal subtraction* (MS), where only the pole terms of ϵ are absorbed. Another more common scheme is *modified MS* ($\overline{\text{MS}}$), where the pole parts and a particular finite part are absorbed, which is equivalent

to setting subtracting the poles of $\bar{\epsilon}$, where

$$\frac{1}{\bar{\epsilon}} = \frac{1}{\epsilon}\left(4\pi\right)^{\epsilon} e^{-\epsilon\gamma_E} \tag{2.32}$$

2.8 Running Coupling

In section 2.6.3 we saw how DR introduces a new energy-scale, μ, into calculations. As previously stated, the result for the calculation of any physical observable should be independent of this μ. Let us consider this more formally.

For a massless theory with exactly one coupling constant g, consider a dimensionless physical observable, R, dependent on a single energy scale Q. By naive dimensional analysis, one would expect $R(Q) = const.$ This is, however, not the case for a renormalisable QFT. If R is calculated as a perturbative expansion in $\alpha = \frac{g^2}{4\pi}$, then this series requires renormalisation to remove the UV divergences. This introduces another mass scale, μ, into the calculation. Dimensional analysis leads us to deduce that R is now dependent on the ratio Q^2/μ^2, and is thus not constant. We also deduce that α is dependent on μ. As stated, the true value of $R\left(Q^2/\mu^2, \alpha(\mu^2)\right)$ cannot depend on μ, so we derive the equation

$$\mu^2 \frac{d}{d\mu^2} R\left(Q^2/\mu^2, \alpha(\mu^2)\right) \equiv \left[\mu^2 \frac{\partial}{\partial \mu^2} + \mu^2 \frac{\partial \alpha}{\partial \mu^2}\frac{\partial}{\partial \alpha}\right] R = 0 \tag{2.33}$$

Introducing the notation

$$t = \log\left(\frac{Q^2}{\mu^2}\right), \qquad \beta\left(\alpha\right) = \mu^2 \frac{\partial \alpha}{\partial \mu^2} \tag{2.34}$$

gives

$$\left[-\frac{\partial}{\partial t} + \beta\left(\alpha\right)\frac{\partial}{\partial \alpha}\right] R\left(e^t, \alpha\right) = 0 \tag{2.35}$$

This is a first order partial differential equation, and can be solved by defining the *running coupling*, $\alpha(Q^2)$, via

$$t = \int_{\alpha}^{\alpha(Q^2)} \frac{dx}{\beta(x)}, \quad \alpha\left(\mu^2\right) \equiv \alpha \tag{2.36}$$

Differentiating this gives

$$\frac{\partial \alpha\left(Q^2\right)}{\partial t} = \beta\left(\alpha\left(Q^2\right)\right), \quad \frac{\partial \alpha\left(Q^2\right)}{\partial \alpha} = \frac{\beta\left(\alpha\left(Q^2\right)\right)}{\beta(\alpha)} \tag{2.37}$$

thus $R\left(1, \alpha\left(Q^2\right)\right)$ is a solution of eq. (2.35). ie the scale dependence of R on Q only enters through the running of the coupling $\alpha\left(Q^2\right)$.

2.8.1 β Function

Let's consider the β function as defined above in eq. (2.34) for QCD. We identify $g \equiv g_s$, $\alpha \equiv \alpha_s = \frac{g_s^2}{4\pi}$. The β function can be computed perturbatively to all orders in α_s. The four-loop β function has been calculated in the $\overline{\text{MS}}$ scheme in [8,9]. The five-loop QED beta function has been presented in [10]. The perturbative expansion is given by

$$\beta(\alpha_s) = -\alpha_s \sum_{n=0} \beta_n \left(\frac{\alpha_s}{4\pi}\right)^{n+1} \tag{2.38}$$

Coefficients $\beta_n, n \geq 2$ depend on the particular subtraction scheme used.

The explicit β_0 is given by

$$\beta_0 = \frac{11C_A - 2n_f}{3} \tag{2.39}$$

with $C_A = N_c = 3$, and n_f the number of light quarks (ie quarks with a mass $m_q \ll Q^2$). The sign of the β function determines the asymptotic ($Q^2 \to \infty$) behaviour of a theory. Solving the first order calculation (ie throwing away $\beta_n, n \geq 1$) gives

$$\log\left(\frac{Q^2}{\mu^2}\right) = \int_{\alpha_s(\mu^2)}^{\alpha_s(Q^2)} \frac{4\pi\, dx}{-\beta_0 x^2} \tag{2.40}$$

Which is rearranged to give

$$\alpha_s(Q^2) = \frac{\alpha_s\left(\mu^2\right)}{1 + \frac{\beta_0}{4\pi}\alpha_s(\mu^2)\log\left(\frac{Q^2}{\mu^2}\right)} \tag{2.41}$$

So as $Q^2 \to \infty$ we find that $\alpha_s(Q^2) \to 0$ if $\beta_0 > 0$. This is indeed the case for QCD with $n_f \leq 16$. This behaviour is known as *asymptotic freedom*, and it allows the use of perturbation theory for processes with large characteristic energy scales Q. The second implication of eq. (2.41) is that for some scale, named Λ_{QCD}, the coupling constant blows up. Λ_{QCD} has been calculated using lattice QCD in eg [11], with different assumed values of n_f, and the results were $\Lambda_{QCD} \simeq 260\text{MeV}$.

This difference in behaviour for small/large energies suggests that we must treat phenomena in these regimes in a qualitatively different manner. We have already discussed the use of perturbation theory, and how it is a valid description for high energies. For low energies perturbation theory breaks down, and we have to work non-perturbatively. This non-perturbative behaviour is responsible for the structure of hadrons, and as the LHC is colliding hadrons, and producing multiple hadrons

in final state jets, we need to have a good description of the physics behind this. To this end, we introduce another energy scale, $\mu = \mu_F$, below which QCD physics is non-perturbative, and above which we work perturbatively. This procedure is known as *factorisation*

2.9 Factorisation

Factorisation is the act of factorising, or decoupling, long- and short- distance effects of a QFT. Within the standard model, factorisation is applied to QCD only, since the low energy/long distance effects of the Electro-Weak sector are well described perturbatively. For a detailed review on factorisation, including theories relating to the validity of the method in different situations, see [12]. We begin with a discussion of the *parton model*.

2.9.1 The Parton Model

The parton model was presented by Feynman in [13]. It states that hadrons are extended objects, consisting of constituents (partons) held together by their interactions. We assume that hadrons are described in terms of virtual partonic states, but that we cannot calculate the structure of these states. We do know how to calculate the scattering processes of a free (ie neglecting parton-parton interactions within the hadron) parton with, for example, an electron.

Consider electron-proton scattering via a virtual photon at high energy and momentum transfer. If we consider this in the center of mass frame, the proton is length contracted in the direction of the collision, and the internal interactions are time dilated. Increasing the center of mass energy of the collision leads to an increased lifetime of the virtual partonic state,τ_{vps}, and a decreased time for the electron to traverse the proton, t_{ep}. If $\tau_{vps} > t_{ep}$ then the proton can be described as being in a particular virtual partonic state, with a definite number of partons throughout the interaction. Each parton in this state is modelled as carrying a definite non-negative fraction, x, of the momentum of the proton. Also, if the momentum transfer is large, the virtual photon can only travel a short distance, so if the density of the partons

in the proton is low enough, the virtual photon will interact with only one parton. Interactions occurring in the final state happen over a timescale too long to interfere with the initial interaction.

So with the p the proton momentum, q the momentum carried by the virtual photon, $Q^2 = -q^2$, $x = Q^2/2p \cdot q$, the parton model states that

$$\sigma_{ep}(x, Q^2) = \sum_a \int_x^1 d\xi \, f_a(\xi) \hat{\sigma}_{ea}(x/\xi, Q^2) \tag{2.42}$$

where a labels the types of parton in the proton, and the $f_a(\xi)$ are the *parton distribution functions* (PDFs), and give the probability of finding a parton of type a with momentum fraction ξ within the proton. $\hat{\sigma}_{ea}(x/\xi, Q^2)$ is the cross-section for parton a with momentum fraction ξ scattering with the electron. It is clear that f describes soft physics, and $\hat{\sigma}$ describes hard physics, but what exactly do we mean by this? Consider the diagram of figure 2.5. If $(k_2)^2_\perp$ is large, ie of the order of some

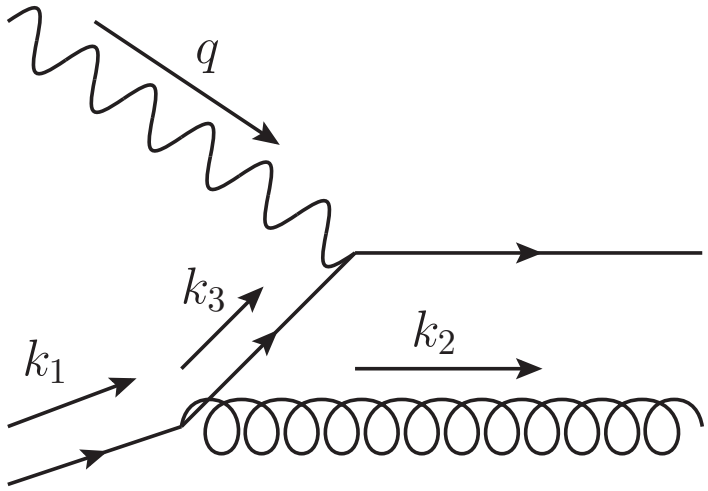


Figure 2.5: A contribution to Deep Inelastic Scattering

hard scale Q^2 in the process, then this is treated as an NLO correction to the hard cross-section, and we consider the incoming parton momentum $\xi p = k_1$. However if $(k_2)^2_\perp$ is small, say of the order Λ^2_{QCD}, then this is treated as a correction to the PDF, and the incoming momentum of the parton for the hard process is $\xi p = k_3$. The scale at which these corrections move from the PDFs to the hard cross-section is referred to as the *factorisation scale*, μ_F. Clearly both the PDFs and the hard

cross-section depend on this scale, although the full physical result cannot depend on this scale, as it is put in by hand. Eq. (2.42) is thus modified as

$$\sigma_{ep}(x, Q^2) = \sum_a \int_x^1 d\xi \, f_a(\xi, \mu_F^2) \sigma_{ea}(x/\xi, Q^2, \mu_F^2) \tag{2.43}$$

This logic can be extended to hadron-hadron initial states, where we assume that exactly one parton from each hadron interacts. Thus the cross-section for proton-proton collision $p(p_1) + p(p_2) \to X$ can be written as

$$\begin{aligned} \sigma_{pp}\left(p_1, p_2\right) &= \sum_{a,b} \int_0^1 dx_1 \int_0^1 dx_2 f_a\left(x_1, \mu_F^2\right) f_b\left(x_2, \mu_F^2\right) \\ &\times \hat{\sigma}_{ab}\left(x_1 p_1, x_2 p_2, \alpha_S\left(\mu_R^2\right), \frac{Q^2}{\mu_F^2}, \frac{Q^2}{\mu_R^2}\right) \end{aligned} \tag{2.44}$$

PDFs are determined from a wide range of experimental results, including fixed target DIS, HERA data, fixed target Drell-Yan, Tevatron W/Z production and Tevatron jet production. There are a number of different PDF sets available, up to NNLO, eg [14–16]

As well as factorising the initial state long-distance physics from the hard cross-section, we also need to consider final state long-distance physics, namely *hadronisation* - the process of going from final state partons to final state hadrons. This is implemented by *fragmentation functions* (FFs), analogous to PDFs. The FFs are written as $D_{H/p}(z, \mu_F^2)$, which gives the probability that a parton of type p produces a hadron of type H with momentum fraction z. Putting these results together for the cross-section of proton-proton collision forming hadron H + anything, we get

$$\begin{aligned} \sigma_{pp\to H}(p_1, p_2) &= \sum_{i,j,k} \int_0^1 dx_1 dx_2 dz \, f_i(x_1, \mu^2) f_j(x_2, \mu^2) \\ &\times \hat{\sigma}_{ij\to k}\left(x_1 p_1, x_2 p_2, \alpha_s(\mu^2), \frac{Q^2}{\mu^2}\right) D_{H/k}(z, \mu^2) \end{aligned} \tag{2.45}$$

where, for ease of notation, we have set $\mu_F = \mu_R \equiv \mu$. This is a common choice for calculations.

Chapter 3

Higher Order Corrections in Perturbative Quantum Field Theories

In this chapter we will describe a number of mechanisms involved in performing higher order calculations in pQFTs. We start with specifying what types of process need to be calculated, then go on to give examples of specific methods used in the calculation of both virtual and real radiative corrections.

3.1 Components of a Higher Order Calculation

Here we will discuss the components of the calculation of higher order corrections to the hard scattering process. We will consider only the case of $2 \to n$ processes, but the same methods apply for any number of incoming particles.

Consider a $2 \to n$ process. The NLO corrections to this process can be split into two types: One-loop corrections to the $2 \to n$ process, and $2 \to n+1$ processes, where one final state particle can become unresolved (ie soft or collinear). For $\mathrm{N^kLO}$ corrections, each individual component is the i-loop correction to the $2 \to n+k-i$ process, with $k-i$ particles which can become unresolved in the final state (with $0 \leq i \leq k$). When the sum of all $\mathrm{N^kLO}$ contributions is taken, the soft and final state collinear singular parts of the hard scattering process sum to zero by the KLN

theorem. Soft and collinear singularities produced by initial state radiation (ISR) are absorbed into N^{k}LO PDFs (see section 2.9.1) to give a finite physical result. For example, the soft singularity from the virtual/real gluon energy going to zero in figures 3.1 and 3.2 will exactly cancel each other, but the soft/collinear singularities from the initial state radiated gluon in figure 3.3 must be absorbed into an NLO PDF to give a finite physical result. Note also that the processes $qg \to t\bar{t} + q$ and $\bar{q}g \to t\bar{t} + \bar{q}$ will also contribute to $pp \to t\bar{t}$ at NLO via ISR.

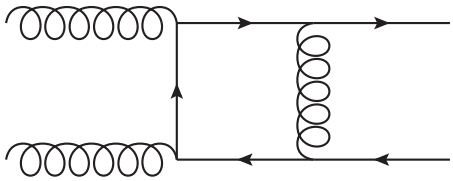


Figure 3.1: gg$\to t\bar{t}$

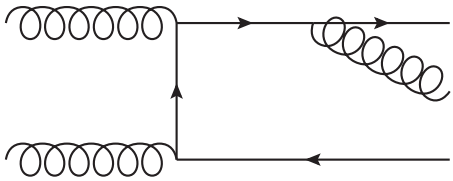


Figure 3.2: gg$\to t\bar{t}$+g

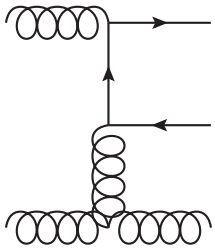


Figure 3.3: gg$\to t\bar{t}$+g

3.2 Methods for Multi-loop Calculations

The general steps for calculating a multi-loop amplitude are as follows:

1. Write down the amplitude using Feynman rules, in terms of an integral over each of the undetermined loop momenta.

2. Employ reduction techniques to express the amplitude as a sum of master integrals with certain coefficients[1].

[1]SecDec can perform integrals with arbitrary tensor structures of loop momenta in the numerator, contracted with external momenta, in a universal and systematic way. Thus this reduction is not strictly required, but it greatly speeds up the calculation

3. Compute the master integrals.

Numerous methods for steps 2 and 3 exist. We shall describe a few of these in this section.

3.2.1 Integration By Parts (IBP)

IBP [17] can be used to find relations between Feynman diagrams, which can in turn be used to reduce a diagram down to a linear combination of certain 'master integrals'. The method is based on the fact that:

$$\int d^d k \frac{\partial}{\partial k^\mu} \left(v^\mu f(k, ...) \right) = 0 \tag{3.1}$$

where $f(k, ...)$ is an integrand arising from a feynman diagram, containing products of denominators, and irreducible numerators in the case where the number of possible scalar products is greater than the number of possible denominators, and v is either a loop momentum or external momentum.

As a simple example, consider the one-loop massless self-energy diagram given in figure 3.4, with propagators raised to powers n_1, n_2,

$$D_1 = k^2,\ D_2 = (k+p)^2$$

$$I(n_1, n_2) = \int d^d k \frac{1}{D_1^{n_1} D_2^{n_2}} \tag{3.2}$$

Consider

$$\begin{aligned} 0 &= \int d^d k \frac{\partial}{\partial k} \cdot k \frac{1}{D_1^{n_1} D_2^{n_2}} \\ &= \int d^d k \left(\frac{d}{D_1^{n_1} D_2^{n_2}} - \frac{2 n_1 k^2}{D_1^{n_1+1} D_2^{n_2}} - \frac{2 n_2 k \cdot (k+p)}{D_1^{n_1} D_2^{n_2+1}} \right) \\ &= \int d^d k \left(\frac{d - 2n_1 - n_2}{D_1^{n_1} D_2^{n_2}} + \frac{n_2 p^2}{D_1^{n_1} D_2^{n_2+1}} - \frac{n_2}{D_1^{n_1-1} D_2^{n_2+1}} \right) \end{aligned} \tag{3.3}$$

Rearranging eq. (3.3), and relabelling $n_2 \to n_2 - 1$ gives

$$p^2 I(n_1, n_2) = \frac{(n_2 + 2n_1 - d - 1)}{n_2 - 1} I(n_1, n_2 - 1) + I(n_1 - 1, n_2) \tag{3.4}$$

and so we can reduce any $I(n_1, n_2)$ down to a constant times $I(1, 1)$ by applying this relationship, and the symmetry $I(n_1, n_2) = I(n_2, n_1)$.

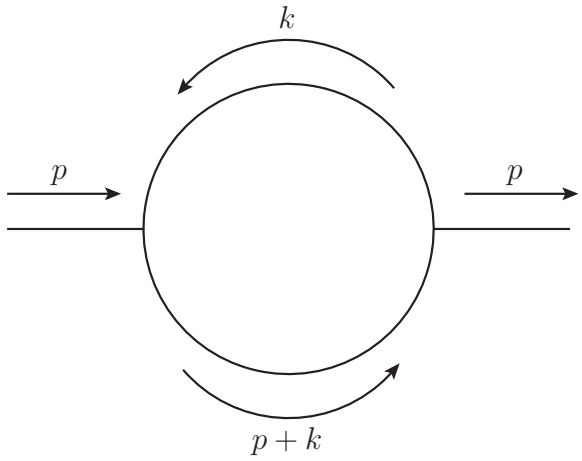


Figure 3.4: A one-loop massless self-energy diagram

For diagrams with many loops and legs there will be a vast number of these relationships, and only a small number will form a linearly independent set. To reduce the original integral into a sum of 'master integrals' one must employ certain rules to automate which relations are used, and which integrals are considered masters. One way to do this is the Laporta Algorithm.

3.2.2 Laporta Algorithm

The Laporta algorithm was proposed in [18]. It has since been implemented in a number of programs, eg [19–21]. The algorithm sets out a systematic way to reduce the huge number of possible IBP relations to a small, manageable set which can be used to express the amplitude as a linear sum of master integrals with various coefficients.

For a generic Feynman diagram, the amplitude can be written as a sum of integrals over loop momenta of terms with different numerators, and different powers of denominators. These can be expressed in a standard form by choosing a basis of denominators and irreducible numerators, and then each term is categorised by the powers of the denominators/numerators.

The algorithm proceeds in a systematic way to generate IBP relations such that each term can be reduced to a sum of master integrals with certain coefficients, and the problem of calculating the amplitude is then to calculate the remaining master integrals.

A system of IBP relations is generated by considering integrands of the type:

$$\frac{\prod_j (D_j)^{\beta_j}}{\prod_i (D_i)^{\alpha_i}}, \tag{3.5}$$

where the D_k are the denominators/irreducible scalar products, and the α_i, β_j are the exponents of these. IBP relations generated from these integrands consist of terms with either the same denominator structure, or terms with one denominator power reduced by one, due to cancellations from the numerator.

First, the simplest case where there are no numerators, and the number of denominators is the same as the number of internal legs in the feynman diagram, all of which have exponent 1, is considered. All IBP relations are generated from this integrand, and those which are linearly independent are added to the system of identities, and substituted into the already existing identities. Then more 'complicated' integrands are considered in a systematic way, until a predetermined stopping point is reached. If this stopping point is chosen suitably, the algorithm will have generated a system of identities which allows you to reduce any integral down to a sum of master integrals. For full details of the algorithm, see [18].

3.2.3 Parametrising the Integrand

A number of different methods for calculating master integrals start with parametrising the integrand using either α-parameters or Feynman parameters.

3.2.3.1 α-Parametrisation

α-parameters are introduced by using the identity

$$\frac{1}{(A+i\delta)^a} = \frac{(-i)^a}{\Gamma(a)} \int_0^\infty d\alpha\, \alpha^{a-1} e^{i(A+i\delta)\alpha} \tag{3.6}$$

where in the case of Feynman diagram, the A are the propagators of the diagram.

3.2.3.2 Feynman Parametrisation

Feynman parametrisation is based on the identity

$$\frac{1}{\prod_{j=1}^{N} A_j^{\nu_j}} = \frac{\Gamma(N_\nu)}{\prod_{j=1}^{N}\Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_j - 1} \, \delta\left(1 - \sum_{i=1}^{N} x_i\right) \frac{1}{\left[\sum_{j=1}^{N} x_j A_j\right]^{N_\nu}} ,$$

$$\text{where } N_\nu = \sum_{j=1}^{N} \nu_j \tag{3.7}$$

where in the case of Feynman diagrams the A_i are the propagators of the diagram.

3.2.4 Mellin-Barnes (MB) Representation

Mellin-Barnes representation has been successfully applied to a number of multi-loop calculations, eg [22–24]. MB representation is used to replace a sum of two terms raised to a certain power by the product of these terms raised to certain powers. This method uses the following identity:

$$\frac{1}{(X+Y)^\nu} = \frac{1}{\Gamma(\nu)} \frac{1}{2\pi i} \int_{-i\infty}^{+i\infty} dw \frac{Y^w}{X^{\nu+w}} \Gamma(\nu+w)\Gamma(-w) \; , \tag{3.8}$$

with the contour chosen such that the poles of $\Gamma(\nu + w)$ all fall to the left of the contour (referred to as *left* poles), and poles of $\Gamma(-w)$ fall to the right (*right* poles). Poles in ϵ occur when left and right poles coincide at $\epsilon = 0$. These poles are resolved by shifting the contour away from these poles, which picks up the residue of all the right poles crossed. For example, for the scalar massless one-loop box, using MB representation and integrating out Feynman parameters leaves us with an integral of the form

$$I(s,t) \sim \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} dw \left(\frac{t}{s}\right)^w \Gamma(2+\epsilon+w)\Gamma(1+w)^2\Gamma(-1-\epsilon-w)^2\Gamma(-w) \tag{3.9}$$

The poles in the complex w plane are arranged as in figure 3.5, where $\epsilon \propto -1-i$ for illustrative purposes. The contour we need to integrate over is $\mathcal{C}$, with the contour closed to the right, but as $\epsilon \to 0$, the right pole at $w = -1-\epsilon$ and the left pole at $w = -1$ coincide, so we need to shift to contour $\mathcal{C}'$, picking up the residue at $w = -1-\epsilon$. Specifically

$$I(s,t) = \frac{1}{2\pi i} \int_{\mathcal{C}} f\,(x,w)\, dw \tag{3.10}$$

$$= \left(-Res\,(f(x,-1-\epsilon)) + \frac{1}{2\pi i} \int_{\mathcal{C}'} f\,(x,w)\, dw\right) \tag{3.11}$$

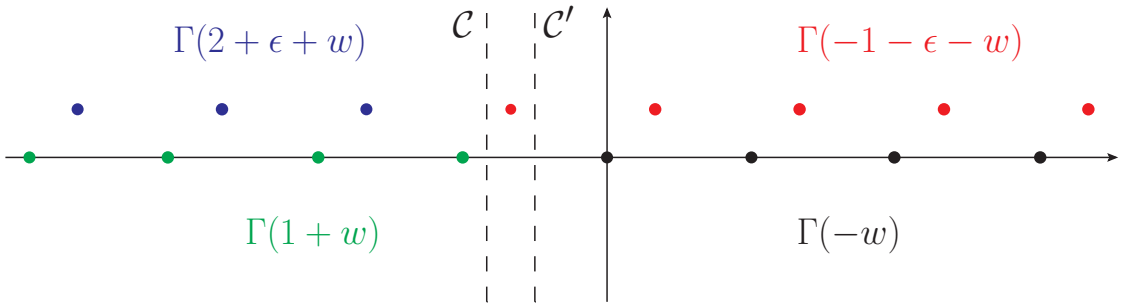


Figure 3.5: Poles of the integrand in the w plane

where $x = \frac{t}{s}$. The integrand can then be expanded as a Taylor series in ϵ, and at each order the residues are summed, giving the result as an infinite sum over residues of right poles. These sums are simple enough for lower orders in ϵ, but for higher orders various results for series of polylogarithms[2] are required (see eg [28]). The public `FORM` [29] package `XSUMMER` [30] implements these results.

3.2.5 Differential Equations

The differential properties of Feynman amplitudes were considered by de Alfaro, Jakšić and Regge [31]. Kotikov presented the method of using differential equations with respect to invariants of the amplitude [32], and the method was further developed by Remiddi and Gehrmann [33–38] and used, together with Lorentz invariance identities, to calculate the massless two-loop, four-point function master integrals with one leg off-shell.

The goal of the method is to find a linear system of first order differential equations for the master integral, which can be solved numerically (or in some cases analytically), or used to examine the behaviour of the amplitude around certain values of the external parameters of the calculation.

As the master integrands will involve the external momenta, but the integrated

[2]The use of symbols [25] to simplify long expressions containing polylogarithms is yielding very impressive results including the evaluation of the one-loop hexagon in 6 dimensions with three massive corners [26], and a massive non-planar two-loop box appearing in $t\bar{t}$ production [27]

result will depend only on external invariants, differentiation with respect to these invariants is rewritten in terms of differentiation with respect to the external momenta. For example,

$$p^2 \frac{\partial}{\partial p^2} \equiv \frac{1}{2}\, p \cdot \frac{\partial}{\partial p} \tag{3.12}$$

or, for a four-point function,

$$s_{12} \frac{\partial}{\partial s_{12}} \equiv \frac{1}{2} \left(p_1 \cdot \frac{\partial}{\partial p_1} + p_2 \cdot \frac{\partial}{\partial p_2} - p_3 \cdot \frac{\partial}{\partial p_3} \right) \tag{3.13}$$

These differentiations can then be applied to the integrands, and algebraic manipulations can be used to cast the numerators of the resulting expression into irreducible numerators, and then we can use IBP identities and LI identities to rewrite these expressions in terms of the original master integrals, and master integrals of subtopologies which are already known. Applying these differentiations leaves us with a system of linear, first order differential equations for the master integrals, which can be solved numerically once certain boundary conditions are fixed.

3.2.6 Difference Equations

The method of difference equations takes the space-time dimension d as a complex variable, and calculates a relationship between the amplitudes $I(d)$ and $I(d-2)$. Analytic properties of $I(d)$ are then used to calculate the integral in $d = 4 - 2\epsilon$ dimensions. The method was proposed by Lee [39], drawing from work of Tarasov [40]. It has since been used successfully in a number of applications by Lee, Smirnov and Smirnov [41, 42], including the evaluation of all four-loop massless self-energy diagrams up to at least $O(\epsilon^5)$.

To illustrate this we will follow the method of Tarasov.

Consider a scalar Feynman integral in d-dimensions as

$$I^{(d)}(\{s_i\}, \{m_i^2\}, \{n_i\}) = \int d^d k_1 \cdots d^d k_L \prod_j^N D_j^{-n_j} \tag{3.14}$$

Where N is the number of propagators, L is the number of loops, D_j are the denominators appearing in the graph,$\{s_i\}$ are invariants made up of external momenta, and $\{m_i^2\}$ are the masses squared of each propagator. For this method, each mass

must originally be considered as independent - desired values for the masses can be reinstated later. Introducing $\alpha-$parametrisation, and applying

$$\mathcal{U}(\partial) \equiv \mathcal{U}\left(\frac{\partial}{\partial m_1^2}, \cdots, \frac{\partial}{\partial m_N^2}\right) \tag{3.15}$$

where $\mathcal{U}$ is the Feynman graph polynomial, discussed at length in section 4.3.1.1. This leads to the result

$$I^{(d-2)}(\{s_i\}, \{m_i^2\}, \{n_i\}) = \left(-\frac{1}{\pi}\right)^L \mathcal{U}(\partial) I^{(d)}(\{s_i\}, \{m_i^2\}, \{n_i\}) \tag{3.16}$$

The action of $\mathcal{U}(\partial)$ on $I^{(d)}(\{s_i\}, \{m_i^2\}, \{n_i\})$ will have the effect of raising some of the n_i by one, and we can then relate these back to the original I using IBP relations. We are then left with a difference equation of the form

$$I^{(d-2)} = A(d) I^{(d)} + B(d) \tag{3.17}$$

where $A(d), B(d)$ are known.

The DRA Method

The method of Lee follows a different route. Firstly, let us assume we have only one master integral for a given topology. If this is not the case the method still holds, but we are left with a system on linear, first order inhomogeneous difference equations in d.

For our master integral, using the method given above we end up with an equation of the form of eq. (3.17), where $B(d)$ is made up of master integrals for subtopologies, and $A(d)$ is a rational function in d. To solve this equation, knowledge of the analytic properties of the integral in a given strip $Re\, d \in [d_0, d_0 + 2)$ is required. It may also be necessary to fix remaining constants by evaluating the integral at some particular values of d. The result for the master integral is then given by an exponentially decaying infinite sum. Thus these sums have a fast convergence, and a very accurate numerical result can be obtained quickly. PSLQ (see section 3.2.7) is then applied to obtain an analytic expression for the master integral, as a series in $\epsilon = (4 - d)/2$. This fast convergence is a significant advantage to this method when compared with, for example, the harmonic (multiple) sums produced by MB representation.

3.2.7 PSLQ Algorithm

While not a method for calculating loop integrals itself, the PSLQ algorithm is a very useful tool for experimentally finding the analytic form of an amplitude which can be computed numerically to a high degree of accuracy.

The algorithm is based on a PSOS (Partial Sum of Squares) scheme, in conjunction with LQ (lower trapezoidal-orthogonal) matrix factorisation, hence 'PSLQ'. The algorithm was first devised by Bailey and Ferguson [43]. A simplified version, also supporting complex and quaternion numbers, was developed by Arno, Bailey and Ferguson [44].

Given a set of numbers $\{x_i\}$ (for applications to loop calculations, we will only discuss the method over $\mathbb{R}$), the aim is to find integers $\{a_i\}$ such that

$$\sum_i a_i x_i = 0, \text{with some } a_i \neq 0 \tag{3.18}$$

The way this is used in particle physics is to take one coefficient in ϵ of the numerical result for an amplitude, A, and a basis of constants, $\{c_i\}$, expected to appear in the result, and run the algorithm with

$$\{x_i\} = \{c_1, \cdots, c_n, A\} \tag{3.19}$$

If PSLQ finds an integer relation (ie a relation of the type given in eq. (3.18)), then this can be rearranged to give[3]

$$A = -\sum_i^n \frac{a_i}{a_{n+1}} c_i \tag{3.20}$$

Notice that $a_{n+1} \neq 0$, otherwise the set of constants $\{c_i\}$ chosen are linearly dependent over $\mathbb{Z}$ and thus not a basis. The choice of constants $\{c_i\}$ is not straightforward. However there are certain insights that can be used to pick the basis. To discuss this, the concept of 'transcendentality weights' should be introduced.

Transcendentality weights follow the rules given in table 3.1 [4]. The maximum tran-

[3] this should perhaps be written $A\text{“} = \text{”} - \sum_i^n \frac{a_i}{a_{n+1}} c_i$, since there is no formal proof that the relationship holds exactly. Sørensen [45] gives an interesting philosophical review of the use of experimental mathematics, with particular reference to PSLQ.

Number, x	Transcendentality Weight, $TW(x)$
1	0
π	1
$log(2)$	2
$Li_n(2)$	n
ζ_n	n
$\zeta_{n_1,\cdots,n_k}$	$\sum_i^k n_i$
$a \times b$	$TW(a) + TW(b)$

Table 3.1: Table of transcendentality weights for various numbers which may occur in the result of a loop calculation

scendentality weight which appears at a given order in ϵ for a given graph can be predicted. Moreover, with an intelligent choice of ϵ-dependent prefactor for the amplitude, all numbers appearing at a given order in ϵ have the same transcendentality weight. This observation serves to keep the set of expected constants $\{c_i\}$ small.

3.3 Methods for Real Radiative Corrections

The methods for dealing with soft/collinear real radiative corrections are very well known and applied at NLO. Extension to NNLO and beyond is a highly non-trivial step, as the singularity structure of the final state becomes much more complicated, involving many overlapping soft and collinear limits. In this section we will discuss the methods of *phase space slicing* and *subtraction* at NLO, and in some extensions to NNLO. The common aspects of phase space slicing and subtraction are that they rely on analytic calculation in only a minimal part of the calculation, namely only

[4]this is by no means an exhaustive list - it has been theorised by Brown [46] that four-loop non-planar massless self-energy diagrams can contain not only multiple zeta values (MZVs), but also Goncharov's polylogarithms [47] of 6th roots of unity. However, recent work by Lee, Smirnov and Smirnov [42] has shown that only MZVs appear up to transcendentality weight twelve in these diagrams

the contributions which can give rise to singularities. Also, for a given process, these contributions are computed without explicit dependence on the particular observable considered. The remainder of the calculation is done using numerical integration.

3.3.1 Subtraction

The idea of subtraction has been widely used for NLO calculations. It was used in the calculation of e^+e^- annihilation by eg [48], and for hadron-hadron collisions in eg [49]. There are a number of different methods within the concept of subtraction, we will mention a few of these below. Firstly a general introduction. Let us consider the components of the NLO calculation of some $2 \to n$ cross-section,

$$\sigma_{NLO} = \int_n (d\sigma_{Born} + d\sigma_{virtual}) + \int_{n+1} d\sigma_{real} \tag{3.21}$$

where the first term is integrated over n-particle phase space, and the last term is integrated over $(n+1)$-particle phase space. There will be poles in ϵ in both the real and virtual parts of this calculation, which cancel when summed (UV divergences have already been taken care of via renormalisation). This fact stops us from being able to directly calculate the cross-section numerically in 4 dimensions.

The idea of subtraction is to add and subtract an extra term, so that the various terms in the calculation can be grouped together into terms which are finite over n- or $(n+1)$-particle phase space, and can thus be computed numerically in 4 dimensions. The cross section is written as

$$\sigma_{NLO} = \int_n \left(d\sigma_{Born} + d\sigma_{virtual} + \int_1 d\sigma_s \right) + \int_{n+1} (d\sigma_{real} - d\sigma_s) \tag{3.22}$$

The subtraction term, $d\sigma_s$, must have 2 properties:

1. $d\sigma_{real} - d\sigma_s$ must be numerically integrable over $n+1$ particle phase space in 4 dimensions.

2. The analytic integration $\int_1 d\sigma_s$ over the unresolved particle phase space must be possible, so that the poles can be resolved analytically, and cancelled with those of $d\sigma_{virtual}$ so that the integration over n particle phase space can be performed in 4 dimensions.

The construction of the subtraction term differs from method to method. Each method involves calculating the matrix element squared in every singular limit (soft, final state collinear and, for hadronic initial states, initial state collinear), and, in these limits, factorising the $n+1$ particle phase space into $n \otimes 1$ phase space in such a way that the integration over the phase space of the unresolved particle is simple, and can be performed analytically. Infact, in these limits, the singular behaviour can be factorised in a process independent way, such that these integrals can be done once and for all, and not repeated for each different process.

Below, I list three popular subtraction methods. For more details, see the references given.

Dipole Subtraction

The method of *dipole subtraction* was proposed by Catani and Seymour [50, 51]. It has been automated in a number of packages, for example MadDipole [52].

FKS Subtraction

The method of Frixione-Kunszt-Signer subtraction (also referred to as *residue subtraction*) was presented for 3 jet NLO cross-sections [53]. It has since been automated in MadFKS [54]

Antenna Subtraction

Antenna subtraction was proposed in [55]. It has been widely used for NLO calculations, eg $e^+e^- \to 4j$ in [56]. It was later applied to colourless initial states at NNLO in [57–59], with the example of $e^+e^- \to 3j$. It has since been applied to hadronic initial states [60–67], and to massive coloured particles in the final state at NLO in [68, 69]. The method can be combined with parton shower Monte Carlo, see [70].

3.3.2 Phase Space Slicing

The method of *phase space slicing* has been widely used for NLO calculations, for example $e^+e^- \to 3j$ in [71,72], and $e^+e^- \to \gamma + j$ in [73]. It has been further applied for hadronic initial states in [74]. The method involves partitioning, or 'slicing', the phase space into different regions, and integrating analytically those regions containing soft or collinear singularities. Let us consider the real radiative NLO corrections to a $2 \to n$ process. The phase space is first split into two regions: hard and soft. The hard region is defined as the region where the energy of each final state particle is greater than a certain value, which introduces a new parameter, δ_s, into the calculation.

If collinear singularities are present, the hard region must also be split into two regions: collinear and non-collinear. The non-collinear region is defined by the fact that the magnitude of all 2-particle invartiants is above a certain value. This introduces a second parameter, δ_c, into the calculation.

The unresolved degrees of freedom in the soft and hard-collinear regions are integrated over analytically, and these terms are then combined with the virtual NLO corrections and the singularities cancel, so they can be integrated numerically over the n-particle phase space in 4 dimensions. The hard-non-collinear region contains no singularities in the $n+1$-particle phase space by construction, and so this can also be integrated numerically in 4 dimensions.

To calcuate the contribution from the soft region, the phase space corresponding to the soft particle is parametrised in terms of the energy and angles of the soft particle, and terms of order δ_s are neglected. For example, in the region where particle $n+3$ is soft we have

$$\begin{aligned} d\Phi_{n+1} &= \left(\prod_{i=3}^{n+3} \frac{d^{d-1}p_i}{2p_i^0(2\pi)^{d-1}}\right)(2\pi)^d\delta^{(d)}\left(p_1+p_2-\sum_{j=3}^{n+3}p_j\right) \\ &= \left[\left(\prod_{i=3}^{n+2} \frac{d^{d-1}p_i}{2p_i^0(2\pi)^{d-1}}\right)(2\pi)^d\delta^{(d)}\left(p_1+p_2-\sum_{j=3}^{n+2}p_j\right)\right]\frac{d^{d-1}p_{n+3}}{2p_{n+3}^0(2\pi)^{d-1}} + \mathcal{O}(\delta_s) \\ &= d\Phi_n \frac{d^{d-1}p_{n+3}}{2p_{n+3}^0(2\pi)^{d-1}} + \mathcal{O}(\delta_s) \end{aligned} \tag{3.23}$$

For particle k soft, we parametrise p_k (up to $\mathcal{O}(\delta_s)$) as

$$p_k = E_k \left(1, \vec{0}^{d-4}, \sin\theta_1 \sin\theta_2, \sin\theta_1 \cos\theta_2, \cos\theta_1\right) \tag{3.24}$$

and use

$$d^{d-1}p_k = dE_k E_k^{d-2} \sin^{d-3}\theta_1 d\theta_1 \sin^{d-4}\theta_2 d\theta_2 d\Omega_{d-4} \tag{3.25}$$

The eikonal approximation of the matrix element squared is calculated, and is then integrated over the degrees of freedom of the soft particle. As with subtraction, the dependence on the soft momentum of the soft approximation to the matrix element squared is well known and process independent, so these integrations need only be done once and for all.

A similar process is carried out for the collinear region. For the region where particles i and j become collinear, the phase space is factorised into an $n \otimes 1$ phase space, where the the two partons i and j are combined with $p_{ij} = p_i + p_j$ in the n particle phase space, and the one particle phase space corresponds to integration over the unresolved degrees of freedom of p_i. These degrees of freedom are then reparametrised in terms of s_{ij} and z, the double invariant and momentum fraction of parton i in parton $i' = ij$ respectively, with terms of $\mathcal{O}(\delta_c)$ neglected. The $d-3$ remaining angles are integrated out. The collinear approximation to the matrix element is then calculated, making use of the Altarelli-Parisi splitting kernels in $4-2\epsilon$ dimensions, $P_{ii'}(z,\epsilon)$. The integration over s_{ij} is then performed analytically, resulting in a single pole in ϵ. Again, the dependence of the collinear approximation to the matrix element squared on s_{ij} is well known and process independent, and so this analytic integration need only be done once.

How do we choose values for δ_s and δ_c? As the two parameters have been introduced by hand, the true result must be independent of their values. Thus as we have neglected terms of $\mathcal{O}(\delta)$ we should choose them small enough such that these terms are small. However we should not choose values too small, as this will affect the numerical stability of the integration in regions labelled as finite.

3.3.3 Sector Improved Phase Space for Real Radiation

The method of *SecToR Improved Phase sPacE for real Radiation* (`STRIPPER`) was proposed by Czakon in [75]. It is based on sector decomposition, together with knowledge of the singularity structure inspired by FKS subtraction. It has been applied to double real radiation in hadronic $t\bar{t}$ production in [76]. It is applicable to NNLO calculations with at least two massive particles in the final state for the LO cross-section. The method requires clever parametrisation of the phase space, which is then decomposed into carefully chosen sectors whereby the singular limits of the amplitudes are factorised.

For a description of the sector decomposition method, see chapter 4.

3.3.4 More Methods

There are a number of full NNLO results available which have been computed via a method not mentioned above. Many of these rely on sector decomposition, for example the NNLO QCD corrections to $pp \to H \to WW \to l\nu l\nu$ of [77], and recently for $H \to b\bar{b}$, where non-linear transformations were used alongside sector decomposition [78]. Also Electro-Weak boson production at NNLO has been calculated using sector decomposition [79]. For further examples of results produced via sector decomposition, see section 4.1.

NNLO QCD results for colourless final states produced in hadronic collisions have been calculated using the q_T subtraction method [80, 81] most recently for diphoton production [82].

Chapter 4

Sector Decomposition

4.1 Current Status of Sector Decomposition

Sector decomposition is an algorithmic method to isolate divergences from parameter integrals as they occur for instance in perturbative quantum field theory. Originally it was devised by Hepp [83] in the context of the the proof of the BPHZ theorem in order to disentangle overlapping ultraviolet singularities. Similar ideas, applied to the subtraction of infrared divergences, can be found e.g. in [84]. It was employed later to extract logarithmic mass singularities from massive multi-scale integrals in the high energy limit at two loops [85, 86].

In [87], the concept of sector decomposition was elaborated to a general algorithm in the context of dimensional regularisation, allowing the isolation of ultraviolet as well as infrared singularities from Feynman parameter integrals in an automated way. First applications of this algorithm were the numerical evaluation of two-loop box diagrams at certain Euclidean points, see e.g. [87–89]. More recently, the method has been used to numerically check a number of analytic three-loop and four-loop results [24, 41, 90–101], most of them produced by either the public program FIESTA [102, 103] or SecDec [104]. Further references about recent applications of sector decomposition to multi-loop calculations can be found in [103, 105].

Sector decomposition also has been combined with other methods for a numerical calculation of loop amplitudes, first on a diagrammatic level in Refs. [106, 107], later for whole amplitudes in Refs. [108–111]. The latter approaches contain a com-

bination of sector decomposition and contour deformation [112–116], which allows one to integrate the Feynman parameter representation of an amplitude numerically in the physical region.

As phase space integrals in D dimensions can be written as dimensionally regularised parameter integrals, sector decomposition can also serve to factorise entangled singularity structures in the case of soft and collinear real radiation. This idea was first presented in [117] and was subsequently applied to calculate all master four-particle phase space integrals where up to two particles in the final state can become soft and/or collinear [118]. Shortly after, this approach has been extended to be applicable to exclusive final states as well by expressing the functions produced by sector decomposition in terms of distributions [119]. Further elaboration on this approach [120, 121] has lead to differential NNLO results for a number of processes [77, 79, 122–128]. The combination of the Frixione-Kunszt-Signer subtraction scheme [53] for soft and collinear real radiation with the decomposition into sectors to treat real radiation at NNLO, as proposed recently in [75], and applied to hadronic $t\bar{t}$ production in [76], is also promising with regards to the reduction of the number of functions produced by the decomposition. A combination of sector decomposition with non-linear variable transformations as proposed in [129], and applied to Higgs decay to bottom quarks in [78], can also serve to reduce considerably the number of functions to integrate, but is less straightforward to automate completely.

To date, the method of sector decomposition has been applied successfully to a considerable number of higher order calculations, for a review we refer to [105, 130]. Here we will concentrate on the method of sector decomposition from a programming point of view.

Despite its success in practical applications, for quite some time there was no formal proof for the existence of a strategy for the iterated sector decomposition such that the iteration is always guaranteed to terminate. This gap has been filled in Ref. [131], by mapping the problem to Hironaka's Polyhedra game [132] and offering three strategies which are proven to terminate. Bogner and Weinzierl also implemented the algorithm in a public computer program for iterated sector de-

composition written in C++ [131]. A Mathematica interface to this program, which also allows the calculation of contracted tensor integrals, has recently been published in [133].

A different strategy guaranteed to terminate, leading to less subsectors than the strategies of Ref. [131], was given by A.V. Smirnov and M. Tentyukov, who implemented the algorithm in the public program FIESTA [102]. Based on a detailed analysis of Hepp and Speer sectors in Ref. [134], an alternative strategy, which is based on Speer sectors, has been implemented in `FIESTA2` [103]. As the latter strategy also uses information on the topology of the graph, it can perform the decomposition more efficiently in certain cases.

Another group has implemented [135] the sector decomposition algorithm in `FORM` [29]. Mapping sector decomposition to convex geometry and using algorithms in computational geometry lead to a guaranteed terminating strategy which seems to be optimal with regards to the number of produced subsectors [136, 137].

4.2 Basic Concept

Consider the two parameter integral

$$I = \int_0^1 dx \int_0^1 dy \, (x + y)^{-2+\epsilon} \; . \tag{4.1}$$

The integrand is singular when x and y tend to zero simultaneously. This type of singularity is referred to as *overlapping*. The aim of sector decomposition is to factorise these overlapping singualrities. To this end, we split the hypercube in two, where one half has $x > y$ and the other half has $y > x$.

$$I = I_1 + I_2 = \int_0^1 dx \int_0^x dy \, (x + y)^{-2+\epsilon} + \int_0^1 dy \int_0^y dx \, (x + y)^{-2+\epsilon} \tag{4.2}$$

We then substitute $y = xt$ in the first sector, and $x = yt$ in the second sector, as shown in figure (4.1). This gives us

$$\begin{aligned} I_1 &= \int_0^1 dx \int_0^1 dt \, x^{-1+\epsilon} (1 + t)^{-2+\epsilon} \\ I_2 &= \int_0^1 dy \int_0^1 dt \, y^{-1+\epsilon} (1 + t)^{-2+\epsilon} \end{aligned} \tag{4.3}$$

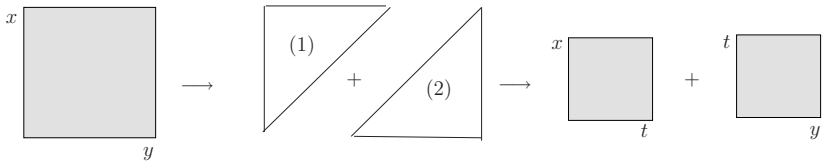


Figure 4.1: Sector decomposition schematically.

Since the term $(1+t)^{-2+\epsilon}$ is finite throughout the integration region, the singularities are now factorised, with the singularity structue determined by the powers of simple monomials of the integration variables. This concept can be applied to multi-dimensional polynomial parameter integrals. In general, not all overlapping singularities are factorised after one step of this process, so the method is iterated until every singularity is factorised.

4.3 The Method

4.3.1 Multi-Loop Integrals

4.3.1.1 Feynman Parametrisation

A general Feynman graph $G^{\mu_1\dots\mu_R}_{l_1\dots l_R}$ in D dimensions at L loops with N propagators and R loop momenta in the numerator, where the propagators can have arbitrary, not necessarily integer powers ν_j, has the following representation in momentum space:

$$\begin{aligned} G^{\mu_1\dots\mu_R}_{l_1\dots l_R} &= \int \prod_{l=1}^{L} \mathrm{d}^D\kappa_l \, \frac{k^{\mu_1}_{l_1}\dots k^{\mu_R}_{l_R}}{\prod_{j=1}^{N} P_j^{\nu_j}(\{k\},\{p\},m_j^2)} \\ \mathrm{d}^D\kappa_l &= \frac{\mu^{4-D}}{i\pi^{\frac{D}{2}}}\mathrm{d}^Dk_l \ , \ P_j(\{k\},\{p\},m_j^2) = q_j^2 - m_j^2 + i\delta \ , \end{aligned} \tag{4.4}$$

where the q_j are linear combinations of external momenta p_i and loop momenta k_l. Introducing Feynman parameters via

$$\frac{1}{\prod_{j=1}^{N} P_j^{\nu_j}} = \frac{\Gamma(N_\nu)}{\prod_{j=1}^{N}\Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_j-1}\, \delta\big(1-\sum_{i=1}^{N} x_i\big) \frac{1}{\Big[\sum_{j=1}^{N} x_j P_j\Big]^{N_\nu}} \,, \quad (4.5)$$

$$\text{where } N_\nu = \sum_{j=1}^{N} \nu_j \text{ , leads to}$$

$$G^{\mu_1\ldots\mu_R}_{l_1\ldots l_R} = \frac{\Gamma(N_\nu)}{\prod_{j=1}^{N}\Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{\nu_j-1}\, \delta\big(1-\sum_{i=1}^{N} x_i\big) \int \mathrm{d}^D\kappa_1 \ldots \mathrm{d}^D\kappa_L$$
$$k_{l_1}^{\mu_1}\ldots k_{l_R}^{\mu_R} \left[\sum_{i,j=1}^{L} k_i^{\mathrm{T}}\, M_{ij}\, k_j - 2\sum_{j=1}^{L} k_j^{\mathrm{T}}\cdot Q_j + J + i\,\delta\right]^{-N\nu} , \quad (4.6)$$

where M is a $L\times L$ matrix containing Feynman parameters, Q is an L-dimensional vector composed of external momenta and Feynman parameters, and J contains kinematic invariants and Feynman parameters.

To perform the integration over the loop momenta k_l, we perform the following shift in order to obtain a quadratic form for the term in square brackets in eq. (4.6):

$$k_l' = k_l - v_l \,, \quad v_l = \sum_{i=1}^{L} M^{-1}_{li} Q_i \,. \quad (4.7)$$

After momentum integration one obtains

$$G^{\mu_1\ldots\mu_R}_{l_1\ldots l_R} = (-1)^{N_\nu} \frac{1}{\prod_{j=1}^{N}\Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} dx_j \; x_j^{\nu_j-1}\, \delta(1-\sum_{l=1}^{N} x_l)$$
$$\sum_{m=0}^{\lfloor R/2\rfloor} (-\frac{1}{2})^m \Gamma(N_\nu - m - LD/2) \underbrace{\Big[(\tilde{M}^{-1}\otimes g)^{(m)}\, \tilde{l}^{(R-2m)}\Big]^{\Gamma_1,\ldots,\Gamma_R}}_{\mathcal{N}(\vec{x})}$$
$$\times \frac{\mathcal{U}^{N_\nu-(L+1)D/2-R}}{\mathcal{F}^{N_\nu-LD/2-m}} \quad (4.8)$$

where

$$\mathcal{F}(\vec{x}) = \det(M)\left[\sum_{j,l=1}^{L} Q_j\, M^{-1}_{jl}\, Q_l - J - i\,\delta\right] \quad (4.9)$$
$$\mathcal{U}(\vec{x}) = \det(M)$$
$$\tilde{M}^{-1} = \mathcal{U} M^{-1} \,, \quad \tilde{l} = \mathcal{U}\, v$$

and $\lfloor R/2 \rfloor$ denotes the nearest integer less or equal to $R/2$. The expression $[(\tilde{M}^{-1} \otimes g)^{(m)}\, \tilde{l}^{\,(R-2m)}]^{\Gamma_1,\ldots,\Gamma_R}$ stands for the sum over all different combinations of R double-indices distributed to m metric tensors and $(R-2m)$ vectors $\tilde{l}$, and is included in the decomposition as the numerator function $\mathcal{N}(\vec{x})$. The double indices $\Gamma_i = (l, \mu_i(l))\,,\ l \in \{1,\ldots,L\}, i \in \{1,\ldots,R\}$ denote the i^{th} Lorentz index, belonging to the l^{th} loop momentum.

As can be seen from Eq. (4.8), the difference between scalar ($R = 0$) and tensor ($R > 0$) integrals, once the Lorentz structure is extracted, is given by the fact that there are additional polynomials of Feynman parameters in the numerator. These polynomials can simply be included into the sector decomposition procedure, thus treating contracted tensor integrals directly without requiring reduction to scalar integrals.

The functions $\mathcal{U}$ and $\mathcal{F}$ also can be constructed from the topology of the corresponding Feynman graph. Cutting L lines of a connected L-loop graph to produce a connected tree graph T defines a *chord* $\mathcal{C}(T)$, with $\mathcal{C}(T)$ being the set of lines not in T. The Feynman parameters associated with the chord $\mathcal{C}(T)$ define a monomial of degree L. The set of all possible trees (referred to as 1-trees to denote that there is 1 connected component) is denoted by $\mathcal{T}_1$. Cutting one more line from $T \in \mathcal{T}_1$ gives 2 disconnected trees, ie a 2-tree. The set of all 2-trees is denoted by $\mathcal{T}_2$, and the corresponding chord to a 2-tree defines a monomial in Feynman parameters of degree L+1. For a 2-tree $\hat{T}$, define the Lorentz invariants $s_{\hat{T}} = \left(\sum_{j \in Cut(\hat{T})} p_j\right)^2$, ie $s_{\hat{T}}$ is the square of the momentum flowing from one component of $\hat{T}$ to the other. Then $\mathcal{U}$ and $\mathcal{F}$ are constructed as follows:

$$\begin{aligned}
\mathcal{U}(\vec{x}) &= \sum_{T \in \mathcal{T}_1} \Big[\prod_{j \in \mathcal{C}(T)} x_j \Big]\,, \\
\mathcal{F}_0(\vec{x}) &= \sum_{\hat{T} \in \mathcal{T}_2} \Big[\prod_{j \in \mathcal{C}(\hat{T})} x_j \Big] (-s_{\hat{T}})\,, \\
\mathcal{F}(\vec{x}) &= \mathcal{F}_0(\vec{x}) + \mathcal{U}(\vec{x}) \sum_{j=1}^{N} x_j m_j^2\,. \qquad (4.10)
\end{aligned}$$

For further methods of calculating $\mathcal{U}, \mathcal{F}$, see [138].

$\mathcal{U}$ is positive semi-definite. Its vanishing is related to the UV subdivergences of the graph. Overall UV divergences, if they occur, will always be contained in the prefactor $\Gamma(N_\nu - m - LD/2)$. In the region where all invariants formed from external momenta are negative (henceforth referred to as the *Euclidean region*) $\mathcal{F}$ is also a positive semi-definite function of the Feynman parameters x_j. Its vanishing does not necessarily lead to an IR singularity. Only if some of the invariants are zero, for example if some of the external momenta are light-like, the vanishing of $\mathcal{F}$ may induce an IR divergence. Thus it depends on the *kinematics* and not only on the topology (like in the UV case) whether a zero of $\mathcal{F}$ leads to a divergence or not. The necessary (but not sufficient) conditions for an IR divergence are given by the Landau equations [139–141], which, in parameter space, simply mean that the necessary condition $\mathcal{F} = 0$ for an IR divergence can only be fulfilled if some of the parameters x_i go to zero, provided that all kinematic invariants formed by external momenta are negative. Now $\mathcal{U}, \mathcal{F}$ and where applicable $\mathcal{N}$ are the starting point for the decomposition.

4.3.1.2 Primary Sector Decomposition

The following is valid for all multi-loop integrals. For ease of notation, consider the scalar case ($R = 0$). Eq. (4.8) then becomes

$$G = (-1)^{N_\nu} \frac{\Gamma(N_\nu - LD/2)}{\prod_{j=1}^{N} \Gamma(\nu_j)} \int_0^\infty \prod_{j=1}^{N} dx_j \; x_j^{\nu_j - 1} \, \delta(1 - \sum_{l=1}^{N} x_l) \; \frac{\mathcal{U}^{N_\nu - (L+1)D/2}}{\mathcal{F}^{N_\nu - LD/2}} \; . \tag{4.11}$$

We would like to transform this into integrations over the unit hypercube, while retaining the feature that only regions with some $x_i \to 0$ can lead to singularities. Furthermore, we would like to preserve any possible symmetries between certain Feynman parameters. To this end we split the integration domain into N sectors, where sector j has $x_i < x_j$ for all $i \neq j$. Using the identity

$$\int_0^\infty d^N x = \sum_{l=1}^{N} \int_0^\infty d^N x \prod_{\substack{j=1 \\ j \neq l}}^{N} \theta(x_l \geq x_j) \; . \tag{4.12}$$

With the θ-function defined as

$$\theta(x \geq y) = \begin{cases} 1 & \text{if } x \geq y \\ 0 & \text{otherwise.} \end{cases}$$

The integral is now split into N domains corresponding to N integrals G_l. We extract a common factor and write: $G = (-1)^{N_\nu}\Gamma(N_\nu - LD/2)\sum_{l=1}^{N} G_l$. For each integral G_l we substitute

$$x_j = \begin{cases} x_l t_j & \text{for} \quad j < l \\ x_l & \text{for} \quad j = l \\ x_l t_{j-1} & \text{for} \quad j > l \end{cases} \tag{4.13}$$

and eliminate x_l using the δ–distribution. Since $\mathcal{U}, \mathcal{F}$ are homogeneous of degrees $L, L{+}1$ respectively, the above substitution gives $\mathcal{U}(\vec{x}) \to \mathcal{U}(\vec{t})x_l^L$, $\mathcal{F}(\vec{x}) \to \mathcal{F}(\vec{t})x_l^{L+1}$, and so using $\int \frac{dx_l}{x_l}\delta\left(1 - x_l(1 + \sum_{k=1}^{N-1} t_k)\right) = 1$ gives

$$G_l \;=\; \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}t_j \, t_j^{\nu_j - 1} \, \frac{\mathcal{U}_l^{N_\nu - (L+1)D/2}(\vec{t})}{\mathcal{F}_l^{N_\nu - LD/2}(\vec{t})} \quad , \quad l = 1, \ldots, N \, . \tag{4.14}$$

Now eq. (4.14) has each G_l as a polynomial parameter integral over the $(N-1)$-dimensional unit hypercube, and these can be treated via iterated sector decomposition as described in the next section.

For a diagram with massless propagators, none of the Feynman parameters occurs quadratically in the function $\mathcal{F} = \mathcal{F}_0$. If massive internal lines are present, $\mathcal{F}$ gets an additional term $\mathcal{F}(\vec{x}) = \mathcal{F}_0(\vec{x}) + \mathcal{U}(\vec{x}) \sum_{j=1}^{N} x_j m_j^2$. If the power of the Feynman parameters in the polynomial forming $\mathcal{F}$ is larger than one for at least two different parameters, initially or at a later stage in the iterated decomposition, an infinite recursion can occur. This happens in the example given in section 5.4.1.2 if the default decomposition strategy is employed. A heuristic procedure is implemented in SecDec to change to a different decomposition strategy only in cases where at least two Feynman parameters occur quadratically, which lead to a terminating algorithm without producing a large number of subsectors in all examples up to 3 loops considered so far. We do not claim that this procedure is guaranteed to terminate, and examples where it does not terminate exist, however it has proved useful for practical purposes. For more details, see Appendix K.2.

4.3.2 General parameter integrals

Sector decomposition can also be employed for more general multi-dimensional parameter integrals. The general form of the integrals is

$$I = \int_0^1 dx_1 \ldots \int_0^1 dx_N \prod_{i=1}^{m} P_i(\vec{x}, \{\alpha\})^{\nu_i} \,, \tag{4.15}$$

where $P_i(\vec{x}, \{\alpha\})$ are polynomial functions of the parameters x_j, which can also contain some constants $\{\alpha\}$. The ν_i are powers of the form $\nu_i = a_i + b_i\epsilon$ (with a_i such that the integral is convergent; note that non-integer powers are also possible). It should be pointed out that most phase space integrals in D dimensions over real radiation matrix elements can also be remapped to functions of the type (4.15). Examples are given in section 5.4.

4.3.2.1 Iterated sector decomposition

Our starting point is a function of the form of Eq. (4.15). First we have to determine which of the integration variables generate singularities at $x_j = 1$, and which ones can lead to singularities at zero *and* one. The parameters x_j for which a denominator vanishes at $x_j = 1$ but not at $x_j = 0$ should be remapped by the transformation $x_j \to 1 - x_j$. If the integrand can become singular at both endpoints of the integration range for a parameter x_j, we split the integration range at 1/2: After the split

$$\int_0^1 \mathrm{d}x_j = \underbrace{\int_0^{\frac{1}{2}} \mathrm{d}x_j}_{(a)} + \underbrace{\int_{\frac{1}{2}}^1 \mathrm{d}x_j}_{(b)} \tag{4.16}$$

and the substitution $x_j = z_j/2$ in (a) and $x_j = 1 - z_j/2$ in (b), all endpoint singularities occur at $z_j \to 0$ only. This splitting is done automatically in SecDec; the user only has to define which integration variables should be split.

So our starting point is a parametric integral where the integrand is singular if some of the integration parameters go to zero. Our aim is to factorise the singularities, i.e. extract them in terms of overall factors of type $x_j^{a_j + b_j\epsilon}$, $a_j \leq -1$. We proceed as follows.

1. Determine a minimal set of parameters, say $\mathcal{S} = \{x_{\beta_1}, \ldots, x_{\beta_r}\}$, such that at least one of the functions $P_i(\vec{x}, \{\alpha\})$ vanishes if the parameters of $\mathcal{S}$ are set to zero.

Notice that $\mathcal{S}$ is in general not unique, and certain heuristic selection criteria can be applied to pick the set likely to result in the minimum number of sectors produced (see Appendix K.1). Notice also that if the exponent $\nu_i = a_i + b_i\epsilon$ has $a_i > 0$, then the function P_i does not cause a singularity in the integrand even if $P_i \to 0$, so decomposing in this case is unnecessary.

2. The corresponding integration range is an r-cube which is decomposed into r *subsectors* by decomposing unity according to

$$\prod_{j=1}^{r} \theta(1 - x_{\beta_j} \geq 0)\, \theta(x_{\beta_j}) = \sum_{k=1}^{r} \prod_{\substack{j=1 \\ j\neq k}}^{r} \theta(x_{\beta_k} - x_{\beta_j} \geq 0)\, \theta(x_{\beta_j}) \,. \tag{4.17}$$

3. Remap the variables to the unit hypercube in each new subsector by the substitution

$$x_{\beta_j} \to \begin{cases} x_{\beta_k} x_{\beta_j} & \text{for} \quad j \neq k \\ x_{\beta_k} & \text{for} \quad j = k \,. \end{cases} \tag{4.18}$$

This gives a Jacobian factor of $x_{\beta_k}^{r-1}$. By construction x_{β_k} factorises from at least one of the functions $P_i(\vec{x}, \{\alpha\})$.

For each subsector the above steps have to be repeated as long as a set $\mathcal{S}$ can be found such that one of the rescaled functions $\tilde{P}_i(\vec{x}, \{\alpha\})$ vanishes if the elements of $\mathcal{S}$ are set to zero. This way new subsectors are created in each subsector of the previous iteration, resulting in a tree-like structure after a certain number of iterations. The iteration stops if the functions $\tilde{P}_i(\vec{x}, \{\alpha\})$ contain a constant term, i.e. if they are of the form

$$\tilde{P}_i(\vec{x}, \{\alpha\}) \;\; = \;\; \alpha_0 + \tilde{Q}_i(\vec{x}\,, \{\alpha\}) \,, \tag{4.19}$$

where $\tilde{Q}_i(\vec{x}, \{\alpha\})$ are polynomials in the variables x_j, and α_0 is a constant, i.e. $\lim_{\vec{x}\to 0} \tilde{Q}_i(\vec{x}, \{\alpha\})$ is nonzero.

The resulting subsector integrals have the general form

$$I \;\; = \;\; \int_0^1 \left(\prod_{j=1}^{N} \mathrm{d}x_j \; x_j^{a_j + b_j\epsilon} \right) \prod_{i=1}^{m} \tilde{P}_i(\vec{x}, \{\alpha\})^{\nu_i} \,. \tag{4.20}$$

The singular behaviour of the integrand now can be read off directly from the exponents a_j, b_j for a given subsector integral.

III. Subtraction of the poles

For a particular x_j the integrand, after the factorisation described above, is of the form

$$I_j = \int_0^1 dx_j \, x_j^{a_j + b_j \epsilon} \, \mathcal{I}(x_j, \{x_{i \neq j}\}, \epsilon) \; . \tag{4.21}$$

If $a_j > -1$, no subtraction is needed and one can go to the next variable x_{j+1}. If $a_j \leq -1$, one expands $\mathcal{I}(x_j, \{x_{i \neq j}\}, \epsilon)$ into a Taylor series around $x_j = 0$. Subtracting the Taylor series (to order[1] p for $|a_j| = p+1$) and adding it back in integrated form, we obtain a part where the poles are subtracted and a part exhibiting $1/\epsilon$ poles times a function depending only on the remaining integration parameters.

$$\begin{aligned} I_j = & \sum_{p=0}^{\lfloor |a_j| \rfloor - 1} \frac{1}{a_j + p + 1 + b_j \epsilon} \frac{\mathcal{I}_j^{(p)}(0, \{x_{i \neq j}\}, \epsilon)}{p!} + \int_0^1 dx_j \, x_j^{a_j + b_j \epsilon} \, R(\vec{x}, \epsilon) \\ R(\vec{x}, \epsilon) = & \, \mathcal{I}(\vec{x}, \epsilon) - \sum_{p=0}^{\lfloor |a_j| \rfloor - 1} \mathcal{I}_j^{(p)}(0, \{x_{i \neq j}\}, \epsilon) \, \frac{x_j^p}{p!} \; . \end{aligned} \tag{4.22}$$

For $a_j = -1$, expanding the above expression in ϵ is equivalent to an expansion in "plus distributions" [119, 142]

$$\begin{aligned} & x^{-1+b\,\epsilon} = \frac{1}{b\,\epsilon} \, \delta(x) + \sum_{n=0}^{\infty} \frac{(b\,\epsilon)^n}{n!} \left[\frac{\ln^n(x)}{x} \right]_+ \; , \\ & \text{where} \\ & \int_0^1 dx \, f(x) \, [g(x)/x]_+ = \int_0^1 dx \, \frac{f(x) - f(0)}{x} \, g(x) \; , \end{aligned} \tag{4.23}$$

with the integrations over the terms containing $\delta(x)$ already carried out.

After having done the subtractions for each x_j, all poles are extracted, such that the resulting expression can be expanded in ϵ. This defines a Laurent series in ϵ

$$I = \sum_{n=-LP}^{r} C_n \, \epsilon^n + \mathcal{O}(\epsilon^{r+1}) \; , \tag{4.24}$$

where the coefficients are finite parameter integrals of dimension $(N - 1 - |n|)$ for $n < 0$ and of dimension $(N-1)$ for $n \geq 0$. LP denotes the leading pole, which

[1] To account for half-integer exponents, e.g. $a_j = -3/2$, we use $\lfloor |a_j| \rfloor$, denoting the nearest integer less or equal to $|a_j|$.

can be at most $2L$ for an L-loop integral. The finite coefficient functions can be directly integrated by Monte Carlo integration if the Mandelstam invariants in $\mathcal{F}$ respectively the numerical constants in a general integrand have been chosen such that the none of the $\tilde{P}_i(\vec{x}, \{\alpha\})$ with exponent $\nu_i < 0$ vanish in the integration domain. If this condition is not met, then the integrands are still integrable, but the integration contour runs through a singularity (since the $-i\delta$ prescription is omitted for practical purposes). Deforming this contour away from the singularity in theory gives the correct result provided no singularities are crossed during the deformation, but automating this process is not a solved problem. A routine implementing this contour deformation is currently in the testing stage in SecDec. See [109, 110], and Appendix F

4.3.2.2 Improving the numerical stability

For $a_j = -1$ in eq. (4.22), the singularity is of logarithmic nature, i.e. $\sim \log(\Lambda)$ if a lower cutoff Λ for the parameter integral was used. In renormalisable gauge theories, linear ($a_j = -2$) or even higher ($a_j < -2$) poles should not occur. However, they can occur at intermediate stages of a calculation, and as they are formally regulated by dimensional regularisation, a method has been worked out for SecDec to be able to deal with higher than logarithmic singularities efficiently. Details are found in appendix A

4.3.2.3 Error treatment

The end result after all algebraic calculation has finished is a number of functions to be integrated at each order in ϵ. The problem of how to integrate these functions to obtain the numerical result then arises. The two basic choices are to integrate the sum of the functions, or to sum the integrated functions, at each required order in ϵ. If the former is chosen, for more complicated integrals the numerical integrator finds it difficult to efficiently deal with the structure of the integrand, and if the latter is chosen the question of how to sum the errors arises. For a discussion of error treatment in SecDec, see Appendix B

Chapter 5

SecDec

5.1 Structure of the program

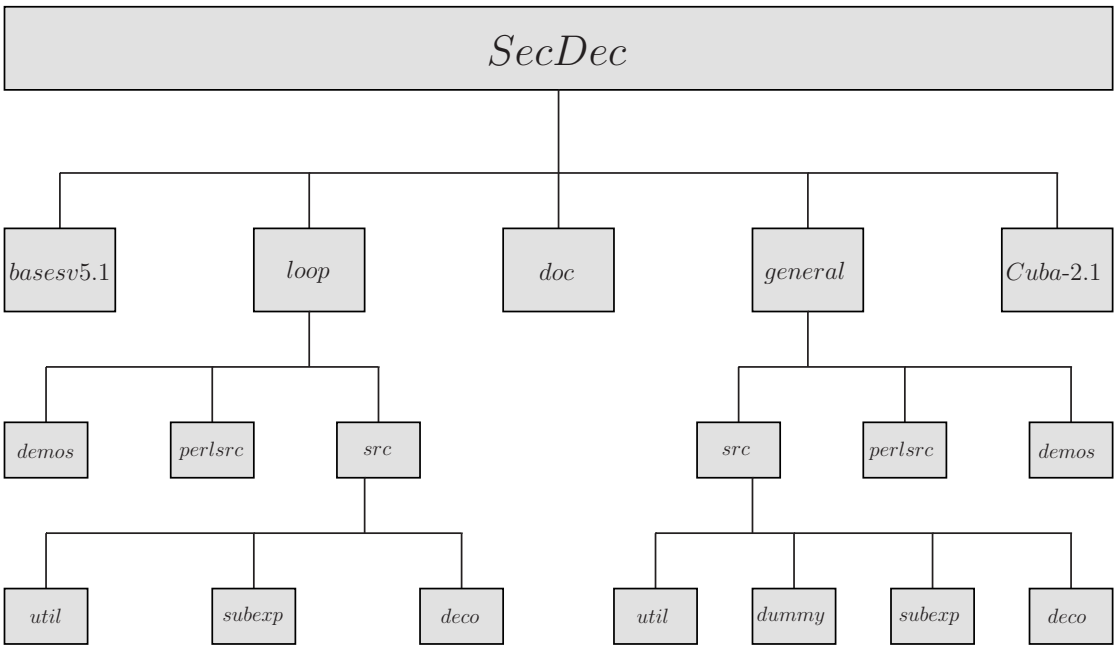


Figure 5.1: Directory structure of the SecDec program.

The program consists of two parts, an algebraic part and a numerical part. The algebraic part uses code written in Mathematica [143] and does the decomposition into sectors, the subtraction of the singularities, the expansion in ϵ and the generation of the files necessary for the numerical integration. In the numerical part, Fortran or C++ functions forming the coefficient of each term in the Laurent se-

ries in ϵ are integrated using the Monte Carlo integration program BASES, version 5.1 [144], or one of the routines from the CUBA library, version 2.1 [145]. The different subtasks are handled by perl scripts. The directory structure of the program is shown in figure 5.1, while the flowchart in figure 5.2 shows the basic flow of input/output streams.

The directories `loop` and `general` have the same global structure, only some of the individual files are specific to loops or to more general parametric functions. The directories contain a number of perl scripts steering the decomposition and the numerical integration. The scripts use perl modules contained in the subdirectory `perlsrc`.

The Mathematica source files are located in the subdirectories `src/deco`: files used for the decomposition, `src/subexp`: files used for the pole subtraction and expansion in ϵ, `src/util`: miscellaneous useful functions. `src/dummy`: files used to create optimized fortran files for functions left implicit during decomposition (`general` only). For the translation of the Mathematica expressions to Fortran77 or C++ functions we use the package `Format.m` [146]. The subdirectories `basesv5.1` and `Cuba-2.1` contain the libraries for the numerical integration, taken from [144] and [145], respectively. The documentation, created by *robodoc* [147] is contained in the subdirectory `doc`. It contains an index to look up documentation of the source code in html format by loading `masterindex.html` into a browser.

The intermediate files and the results will be stored in a subdirectory of the working directory whose name *mysubdir* can be specified by the user (*subdir=mysubdir* in `param.input`, leaving this blank is a valid option). A subdirectory of *mysubdir* with the name of the graph, respectively integral to calculate will be created by default. If the user would like to store the files in a directory which is not the subdirectory of the working directory, for example in */scratch*, he can do this by specifying the full path *outputdir=/scratch* in `param.input`. An example of a directory structure created by running the examples *NPbox, QED, ggtt1, A61*, a user-defined three-loop example, and a four-loop example to be written to the scratch disk is given in figure 5.3.

The directory created for each graph will contain subdirectories according to

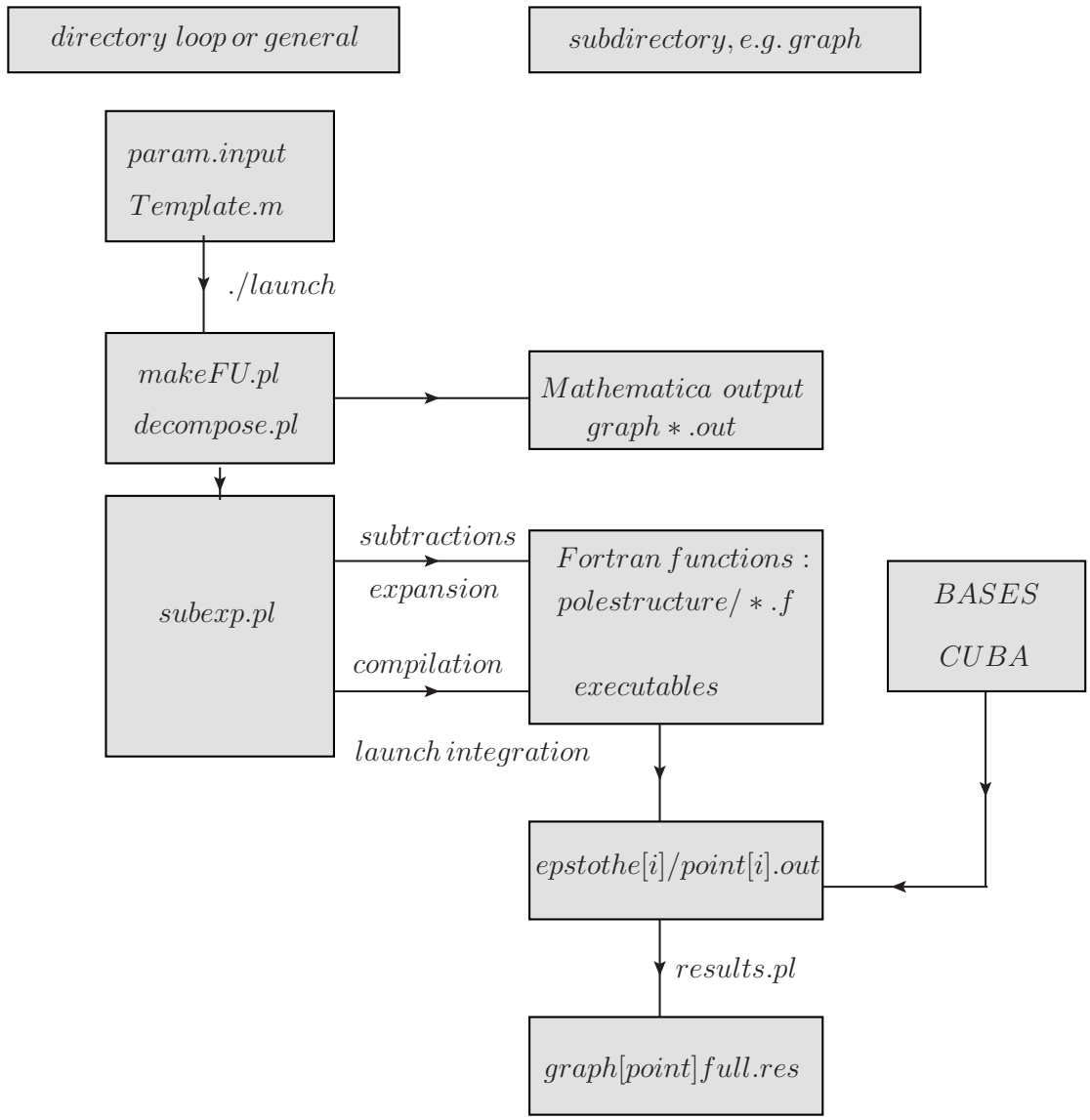


Figure 5.2: Flowchart showing the main steps the program performs to produce the result files. In each of the subdirectories `loop` or `general`, the file `Template.m` can be used to define the integrand. The produced files are written to a subdirectory created according to the settings given in `param.input`. By default, a subdirectory with the name of the graph or integrand is created to store the produced functions. This directory will contain subdirectories according to the pole structure of the integrand. The perl scripts (extension .pl) are steering the various steps to be performed by the program.

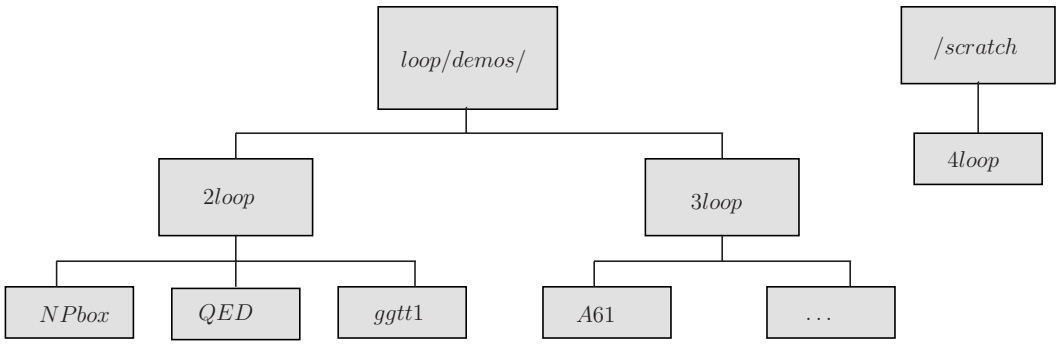


Figure 5.3: Example for a directory structure created by running the loop demo programs NPbox, QED, ggtt1, A61. A four-loop example defined by the user to be written to the scratch disk is also shown.

the pole structure of the graph. The labelling for the pole structure is of the form e.g. `2l0h0`, denoting 2 logarithmic poles, no linear and no higher than linear poles. It should be pointed out that this labelling does not necessarily correspond to the final pole structure of the integral. It is merely for book-keeping purposes, and is based on the counting of the powers of the factorised integration variables. In more detail, if i_1 variables have power -1, i_2 variables have a power $-2 \leq i_2 < -1$ and i_3 variables have a power < -2, the labelling will be $i_1\, l\, i_2\, h\, i_3$, even though the non-logarithmic poles will disappear upon ϵ-expansion. In particular, for half-integer powers, the labelling does not correspond to "true" poles, but rather to terms which can be cast into functions like $\Gamma(-3/2 - \epsilon)$, which are well-defined in the context of dimensional regularisation, where ϵ can be regarded as an arbitrary (complex) parameter. Note also that in the case of a prefactor containing $1/\epsilon$ poles multiplying the parameter integral, the poles which are flagged up at this stage of the program will only correspond to the poles read off from the integration parameters. In any case, the final result will be given to the order specified by the user, eg. *epsord=0* in `param.input`.

Each of these "polestructure" directories contains further subdirectories where the files for a particular power in epsilon are stored. An example is given in figure 5.4.

The user only has to edit the following two files:

- `param.input`: (text file)

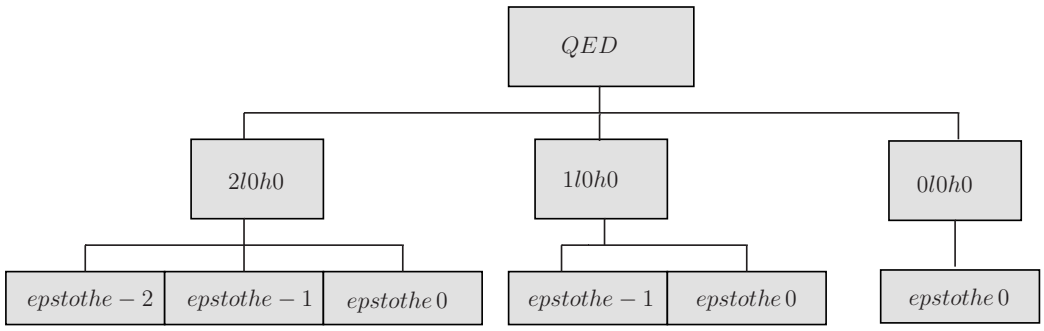


Figure 5.4: Example for a directory tree corresponding to the pole structure of the graph `QED` contained in the demo programs.

specification of paths, type of integrand, order in ϵ, output format, parameters for numerical integration, further options

- `Template.m`: (Mathematica syntax)
  - for loop integrals: specification of loop momenta, propagators; optionally numerator, non-standard propagator powers
  - for general functions: specification of integration variables, integrand, variables to be split at 1/2

For a detailed explanation of requirements/options in parameter and template files, see appendices C and D.
To give a specific example rather than empty templates, the files `param.input` and `Template.m` in the `loop` subdirectory contain the setup for example 5.4.1.1, the non-planar massless on-shell two-loop box diagram, while those in the `general` directory contain the setup for example 5.4.2.1, a hypergeometric function of type ${}_5F_4$. Apart from these default parameter/template files, the program comes with example input and template files in the subdirectories `loop/demos` respectively `general/demos`, described in detail in section 5.4.

The user can choose the numerical integration routine and the settings for the different integrators contained in the Cuba library, or for BASES, in the file `param.input`. The compilation of the chosen integration routine with the corresponding settings will be done automatically by the program.

5.2 Features

SecDec has a number of useful features available, and a number of exciting new features planned for the future. Examples illustrating these features can be found in the section 5.4.

Original Features

- Parameters (eg invariant masses) can be left implicit up until the numerical integration. This means that the algebraic part needs to only be run once, and then various numerical points can be calculated
- Contracted tensors in the numerator and non-integer powers of propagators in the denominator allowed for loop integrals
- Choice of integrators - BASES and the Cuba library are available, with full control over parameters used
- Simple to use with Portable Batch System for parallel processing, and readily adjusted to work for different batch syntax
- Subtraction with non-integer powers of variables

New Features

A number of new features have been added since SecDec-1.0:

- Functions can be left implicit for the algebraic stage. This is particularly useful for dealing with complicated but finite functions, where quantitative knowledge of these functions is not required to guide the decomposition, or for including measurement functions (currently `general` only). See Appendix E.2 for full instructions
- Choice of Fortran or C++ for numerics (currently `loop` only)
- Integration of complex functions implemented in C++

- Automation of calculating a set of different numerical points (currently `general` only), useful for investigating behaviour of an integral across a range of parameter values.

Future Features

There are various features currently being planned/developed:

- Applying contour deformation to calculate loop integrals in the physical region. This has been implemented, and is in the testing stage. It has proved successful in a number of cases.
- Interfacing with a matrix element generator, eg FeynArts+FormCalc [148], to automate calculation of real radiation contributions
- Interfacing with a reduction package, eg Reduze [21], to automate numerical evaluation of required master integrals for a given process.

5.3 Installation and usage

Installation

The program can be downloaded from `http://projects.hepforge.org/secdec/`.

Installation is done by unpacking the tar archive, using the command *tar xzvf SecDec.tar.gz*. This will create a directory called `SecDec` with the subdirectories as described above. Change to the `SecDec` directory and run *./install*.

Prerequisites are Mathematica, version 6 or above, perl (installed by default on most Unix/Linux systems), a Fortran compiler (e.g. gfortran, ifort), and a C++ compiler (eg gcc). The install script only checks if Mathematica and perl are installed on the system and inserts the corresponding path into the perl scripts. The install script does not test the existence of a Fortran or C++ compiler because the compiler should be specified by the user in `param.input`. If no compiler is specified, it defaults to *gfortran*.

Usage

1. Change to the subdirectory `loop` or `general`, depending on whether you would like to calculate a loop integral or a more general parameter integral.

2. Copy the files `param.input` and `Template.m` to create your own parameter and template files `myparamfile`, `mytemplatefile`.

3. Set the desired parameters in `myparamfile` and define the propagators etc. (`loop`) or integrand etc. (`general`) in `mytemplatefile`.

4. Execute the command *./launch -p myparamfile -t mytemplatefile* in the shell. If you omit the option *-p myparamfile*, the file `param.input` will be taken as default. Likewise, if you omit the option *-t mytemplatefile*, the file `Template.m` will be taken as default. If your files *myparamfile, mytemplatefile* are in a different directory, say, *myworkingdir*, use the option
   -d myworkingdir, i.e. the full command then looks like *./launch -d myworkingdir -p myparamfile -t mytemplatefile*, executed from the directory `SecDec/loop` or `SecDec/general`.
   Alternatively, you can call the launch script from any directory if you prepend the path to the launch script, i.e. the command
   path_to_launch/launch -p myparamfile -t mytemplatefile executed from **myworkingdir** would run the program in the same way. *path_to_launch* can be either the full or relative path for `SecDec/loop` or `SecDec/general`.

   The program tries to detect the path to Mathematica automatically. In case you get the message "path for Mathematica not automatically found", please insert the path to Mathematica on your system manually for the variable `$mathpath` in the file `perlsrc/mathlaunch.pl`.

   The *./launch* command will launch the following perl scripts:

   - `makeFU.pl`: (only for loop integrals) constructs the integrand functions $\mathcal{F}$,$\mathcal{U}$ and the numerator function from the propagators and indices given in `Template.m`.

- `decompose.pl`: launches the iterated sector decomposition
- `subexp.pl`: launches the subtractions and epsilon-expansions and writes the Fortran or C++ functions. Depending on the "exe-flag" specified in the parameter file (see below for a detailed explanation of the flag), this script also launches the compilation and the numerical integrations.

5. Collect the results. Depending on whether you have used a single machine or submitted the jobs to a cluster, the following actions will be performed:

   - If the calculations are done sequentially on a single machine, the results will be collected automatically (via `results.pl` called by `launch`). The output file will be displayed with your specified text editor. The results are also saved to the files `[graph]_[point]epstothe*.res` and `[graph]_[point]full.res` in the subdirectory `subdir/graph` (loops) respectively `subdir/integrand` (general integrands) (name specified in `param.input`, where you can also specify different names for different numerical points).
   - If the jobs have been submitted to a cluster: when all jobs have finished, execute the command *./results.pl [-d myworkingdir -p myparamfile]* in a shell from the directory `SecDec/loop` or `SecDec/general` to create the file containing the final results.

     If the user needs to change the batch system settings: manually edit `perlsrc/makejob.pm` and `perlsrc/launchjob.pm`. This writes the user-specified syntax to the scripts `job[polestructure]` in the corresponding `subdir/graph` or `subdir/integrand` subdirectory.

6. After the calculation and the collection of the results is completed, you can use the shell command *./launchclean[graph]* to remove obsolete files.
   If called with no arguments, the script only removes object files, launch scripts, makefiles and executables, but leaves the Fortran or C++ files created by Mathematica, so that different numerical points can be calculated without re-

running the Mathematica code. If called with the argument 'all' (i.e. *./launchclean[graph] all*), it removes everything except the result files displaying the final result and the timings.

The **'exe' flag** contained in `param.input` offers the possibility to run the program only up to certain intermediate stages. The flag can take values from 0 to 4. The different levels are:

exe=0: does the iterated sector decomposition and writes files containing lists of subsector functions (`graphsec*.out`) for each pole structure to the output subdirectory. Also writes the Mathematica files `subandexpand*.m` for each pole structure, which serve to do the symbolic subtraction, epsilon expansion and creation of the Fortran or C++ files. Also writes the scripts `batch[polestructure]` which serve to launch these jobs at a later stage.

exe=1: launches the scripts `batch[polestructure]`. This will produce the Fortran or C++ functions and write them to individual subdirectories for each pole structure.

exe=2: creates all the additional files needed for the numerical integration.

exe=3: compilation is launched to make the executables.

exe=4: the executables are run.

If the first steps of the calculation, e.g. the decomposition or the creation of the Fortran or C++ functions, are already done, the following commands are available to continue the calculation without having to restart from scratch:

- **finishnumerics.pl [-d myworkingdir -p myparameterfile]:**
  if the 'exe' flag in `param.input` resp. `myworkingdir/myparameterfile` is set smaller than four, this will complete the calculation without redoing previous steps.

- **justnumerics.pl [-d myworkingdir -p myparameterfile]:**
  if you would like to redo just the numerical integration, for example to produce results for a different numerical point or to try out a different number

of sampling points, iterations etc. for the Monte Carlo integration: change the values for the numerical point resp. the settings for the Monte Carlo integration and the *pointname* in the parameter file, and then use the command *./justnumerics.pl [-d myworkingdir -p myparameterfile]* to redo only the numerical integrations (if the Fortran files f*.f or C++ files f*.c have been produced already). Using this option skips the Mathematica subtraction and epsilon expansion step which can be done once and for all, as the variables at this stage are still symbolic. After completion of the numerical integrations, use the command *./results.pl [-d myworkingdir -p myparameterfile]* to collect and display the results as above.

For details on how to automate the calculation for a set of different numerical points, see appendix E.1.

For a description of how to leave finite functions implicit throughout the algebraic stage, see appendix E.2

It should be mentioned that the code starts working first on the most complicated pole structure, which takes longest. This is because in case the jobs are sent to a cluster, it is advantageous to first send the jobs which are expected to take the most time.

5.4 Description of Examples

5.4.1 Loop integrals

The examples described below can be found in the subdirectory `loop/demos`.

5.4.1.1 Non-planar massless two-loop box

The non-planar massless two-loop box is a non-trivial example, as the sector decomposition applied to the standard representation, produced by combining all propagators simultaneously with Feynman parameters, exhibits "non-logarithmic poles" (i.e. exponents of Feynman parameters ≤ -1) in the course of the decomposition. It should be pointed out that, even though the program can deal with linear or

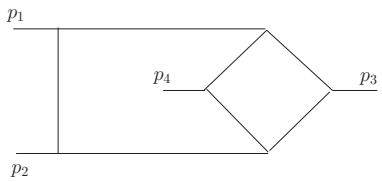


Figure 5.5: The non-planar two-loop box, called *NPbox* in example 5.4.1.1.

higher poles in a completely automated way, it is often a good idea to investigate if the integrand can be re-parametrised such that poles of this type do not occur, because these poles require complicated subtraction terms which slow down the calculation. Non-linear transformations as e.g. described in [129] can be useful in this context. Further, integrating out first one loop momentum, and then combine the remaining propagators with the obtained intermediate result using another set of Feynman parameters often leads to a representation where at least one of the parameters can be factorised without sector decomposition, thus speeding up the calculation considerably. This is demonstrated for the non-planar massless two-loop box in appendix H, and the template to calculate the graph in this way can be found in `SecDec/general/demos`.

To obtain results for the non-planar massless two-loop box shown in figure 5.5 without doing any analytical steps, copy the file `loop/param.input` to a new parameter file, say `paramNPbox.input`, and specify the desired order in ϵ, the numerical point and possible further options. Likewise, copy the file `loop/Template.m` to a new template file, say `templateNPbox.m` (this already has been done for the examples described here, see the subdirectory `loop/demos`). Then, in the `loop/demos` directory, use the command *../launch -p paramNPbox.input -t templateNPbox.m*. The file `templateNPbox.m` already has the propagators of the non-planar double box predefined. In `paramNPbox.input`, we defined the prefactor such that a factor of $-\Gamma(3+2\epsilon)$ is *not* included in the numerical result: if we define

$$G_{NP}(s,t,u) = -\Gamma(3+2\epsilon)\sum_{n=0}^{4}\frac{P_n}{\epsilon^n} + \mathcal{O}(\epsilon)\ , \tag{5.1}$$

the program should yield the results for P_n, given in Table 5.1. Note that according to eq. (4.4), we always divide L-loop integrals by $(i\pi^{\frac{D}{2}})^L$, so this factor is never

included in the numerical result. The decomposition produces 384 subsectors.

(s,t,u)	(-1,-1,-1)	(-1,-2,-3)
P_{-4}	$1.75006 \pm 1.3 \times 10^{-4}$	$0.41670 \pm 1.1 \times 10^{-4}$
P_{-3}	-2.99969± 0.00055	-0.9313 ±0.00067
P_{-2}	-22.821 ± 0.003	-5.8599 ± 0.0035
P_{-1}	113.629 ±0.013	42.79 ±0.02
P_0	-395.27 ± 0.05	-162.73±0.09

Table 5.1: Numerical results for the points $(s,t,u) = (-1,-1,-1)$ and $(-1,-2,-3)$ of the massless non-planar double box.

The result for the graph called *NPbox* at the numerical point called *point* in the input file will be written to the file `NPbox_[point]full.res` in the subdirectory `2loop/NPbox`, where `2loop` is a subdirectory which has been created by the program, using the directory name the user has specified in the first entry of `paramNPbox.input`. By default, a subdirectory with the name of the graph is created, but the user can also specify a completely different directory (e.g. `scratch`) where the results will be written to (second entry in `paramNPbox.input`).

More information about the decomposition is given in the file `NPboxOUT.info`. Information about the numerical integration is contained in the files `[point]intfile.log` in the subdirectories `graph/polestructure/epstothe[i]`, where "polestructure" is of the form e.g. `2l0h0`, denoting 2 logarithmic poles and 0 linear, 0 higher poles.

It should be emphasized that in `param.input`, the numbers for the Mandelstam invariants should be defined as the *Euclidean* values, so the values for s, t, u, p_i^2 should always be negative in `param.input`. Note also that the condition $s + t + u = 0$ cannot be fulfilled numerically in the Euclidean region, so it should not be used in `onshell=`$\{\ldots\}$ in the template file to eliminate u from the function $\mathcal{F}$ in the case of non-planar box graphs.

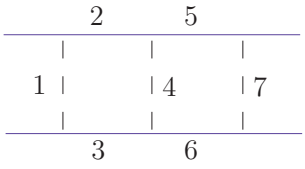


Figure 5.6: Blue (solid) lines denote massive particles.

5.4.1.2 Planar two-loop ladder diagram with massive on-shell legs

The purpose of this example is to show how to deal with diagrams where the decomposition could run into an infinite recursion if the default strategy is applied. The rungs of the ladder are massless particles (e.g. photons), while the remaining lines are massive on-shell particles, depicted by solid (blue) lines in figure 5.6. To run this example, execute the command *../launch -p paramQED.input -t templateQED.m* from the `loop/demos` directory. Only the primary sectors number one and seven are at risk of running into infinite recursion, therefore they are listed in the third-last item of `paramQED.input` as the ones to be decomposed by a different strategy. The results for the numerical point called *point* will be written to the file `QED_[point]full.res` in the subdirectory `2loop/QED`. Numerical results for some sample points are given in Table 5.2. The kinematic points are defined by the mass m and the Mandelstam variables $s = (p_1 + p_2)^2, t = (p_2 + p_3)^2$. We extracted a prefactor of $\Gamma(1+\epsilon)^2$.

(s,t, m)	(-0.2,-0.3,1)	(-3/2,-4/3,1/5)
P_{-2}	-1.56161± 1.33×10^{-4}	-2.1817 ± 0.0003
P_{-1}	-5.3373 ±0.0018	-1.4701 ±0.0026
P_0	1.419 ±0.025	30.191± 0.014
P_1	62.46 ± 0.18	140.73±0.057
P_2	284.76 ± 0.87	450.67±0.19

Table 5.2: Numerical results up to order ϵ^2 for the points $(s,t,m) = (-0.2,-0.3,1)$ and $(-3/2,-4/3,1/5)$ of the two-loop ladder diagram shown in figure 5.6. An overall factor of $\Gamma(1+\epsilon)^2$ is not included in the numerical result.

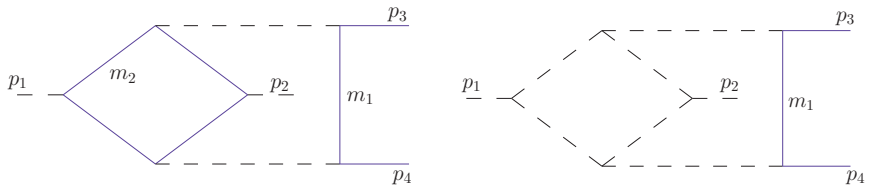


Figure 5.7: Non-planar graphs occurring in the calculation of $gg \to t\bar{t}$ at NNLO. Blue (solid) lines denote massive particles.

5.4.1.3 Non-planar two-loop diagrams with two massive on-shell legs

This example gives results for two non-planar graphs occurring in the calculation of $gg \to t\bar{t}$ at NNLO, shown in figure 5.7. The analytic results for *ggtt*1 is not yet available, and for *ggtt*2, the result is in preparation [27]. Numerical results at Euclidean points can be produced by choosing numerical values for the invariants s, t, u, m^2 in `paramggtt1.input` respectively `paramggtt2.input` and then executing the command *../launch -p paramggtt1.input -t templateggtt1.m* in the `loop/demos` directory, analogously for *ggtt2*. Results for two sample points are shown in Table 5.3.

5.4.1.4 A rank one tensor two-loop box

In order to demonstrate how to run the program for integrals with non-trivial numerators, we give the example of a rank one planar massless on-shell two-loop box, where we contract one loop momentum in the numerator by $2\,p_3^\mu$.

$$G = \int \frac{d^D k\, d^D l}{(i\pi^{\frac{D}{2}})^2} \frac{2\,p_3 \cdot k}{k^2(k-p_1)^2(k+p_2)^2(k-l)^2(l-p_1)^2(l+p_2)^2(l+p_2+p_3)^2}\,, \tag{5.2}$$

where we omitted the $i\delta$ terms in the propagators. The result for the kinematic sample point $(s,t,u) = (-3,-2,5)$ is shown in Table 5.4. Note that in this example, we used a positive value for the Mandelstam invariant u, which seems to contradict

ggtt1		
(s,t,u,m_1^2,m_2^2)	(-0.5,-0.4,-0.1,0.17,0.17)	(-1.5,-0.3,-0.2,3,1)
P_0	-38.0797±0.0027	-0.19904±1.5×10^{-5}
P_1	-263.22± 0.015	-0.71466±6×10^{-5}
P_2	-936.86± 0.06	-1.45505± 0.0002
ggtt2		
(s,t,u,m_1^2,m_2^2)	(-0.5,-0.4,-0.1,0.17,0)	(-1.5,-0.3,-0.2,3,0)
P_{-4}	-10.9159 ± 0.0006	-0.13678±1.46×10^{-5}
P_{-3}	-43.5213 ± 0.0075	-0.2087 ±0.00024
P_{-2}	165.384 ± 0.048	3.3417 ±0.0014
P_{-1}	20.842±0.268	-6.593±0.007
P_0	2117.5 ± 1. 57	20.42±0.04

Table 5.3: Numerical results for the diagrams shown in figure 5.7. The finite diagram ggtt1 has been calculated up to order ϵ^2. An overall factor of $\Gamma(1+\epsilon)^2$ is extracted.

(s,t,u)	(-3,-2,5)
P_{-4}	-0.319449 ±1.7×10^{-5}
P_{-3}	0.46536 ±8×10^{-5}
P_{-2}	0.5848 ±0.0004
P_{-1}	-3.3437 ± 0.0013
P_0	-1.6991± 0.0035

Table 5.4: Numerical results for the point $(s,t,u)=(-3,-2,5)$ of the rank one two-loop ladder diagram given by eq. (5.2). An overall factor of $\Gamma(1+\epsilon)^2$ has been extracted.

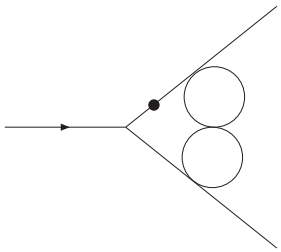


Figure 5.8: The three-loop vertex diagram $A_{6,1}$ with the dotted propagator raised to the power $1+\epsilon$.

the requirement to have only Euclidean values for the invariants. However, in this case we can do this because the function $\mathcal{F}$ does not depend on u at all. The numerator does depend on u, but as a numerator which is not positive definite does not spoil the numerical convergence, we can as well choose a numerical value for u such that the relation $s+t+u=0$ is fulfilled. This has the advantage that it allows us to use the latter relation to simplify the numerator.

5.4.1.5 A three-loop vertex diagram with ϵ-dependent propagator powers

This example shows how to calculate diagrams with propagator powers different from one. The results for the graph $A_{6,1}$ (notation of Ref. [90]), given in Table 5.5, can be produced by running *../launch -p paramA61.input -t templateA61.m* from the `loop/demos` directory.

P_{-3}	P_{-2}	P_{-1}	P_0	P_1	P_2
0.16666	1.8334	18.123	125.32	889.96	5325.3

Table 5.5: Numerical results for the diagram shown in figure 5.8 with the dotted propagator raised to the power $1+\epsilon$. The errors are below one percent.

The analytical result for this diagram with general propagator powers is given in Ref. [90] and is also given in the file `3loop/A61/A61analytic.m` to allow comparisons between analytical and numerical results for arbitrary propagator powers.

5.4.1.6 Two-Loop Massive Triangle in the Physical Region

This example demonstrates the use of contour deformation to calculate diagrams in the physical region. This feature is not currently publicly available, and as such template and parameter files for this example are not provided in the `loop/demos` directory.

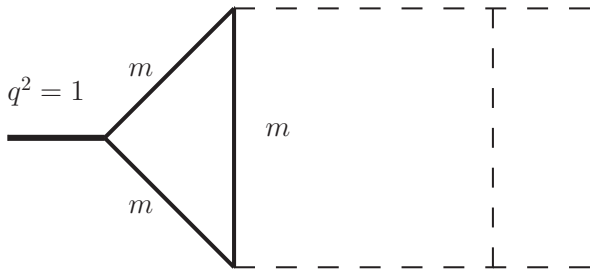


Figure 5.9: Two-Loop Massive Triangle

Results for $m^2 = 0.2$ are given in table 5.6. A factor of $-m^{-2\epsilon}Exp[-2\epsilon\gamma_E]$ has been extracted from the result. Analytic result is that of P_{126}, taken from [149]

	SecDec	Analytic
ϵ^{-2}	4.47+3.02*I	4.47167+3.02354*I
ϵ^{-1}	-16.6+2.98*I	-16.54938+2.98071*I
ϵ^{0}	10.98-63.57*I	10.9682- 63.63681*I

Table 5.6: Results for Two-Loop Massive Triangle, $m^2 = 0.2$

5.4.2 More general polynomial functions

The examples described below can be found in the subdirectory `general/demos`.

5.4.2.1 Hypergeometric functions

As an example for "general" polynomial functions, we consider the hypergeometric functions ${}_pF_{p-1}(a_1, \ldots, a_p; b_1, \ldots, b_{p-1}; \beta)$, using the integral representation recur-

sively:

$$ {}_pF_{p-1}(a_1,\ldots,a_p;b_1,\ldots,b_{p-1};\beta) = \frac{\Gamma(b_{p-1})}{\Gamma(a_p)\Gamma(b_{p-1}-a_p)} \times \tag{5.3} $$
$$ \int_0^1 dz\,(1-z)^{-1-a_p+b_{p-1}}\,z^{-1+a_p}\,{}_{p-1}F_{p-2}(a_1,\ldots,a_{p-1};b_1,\ldots,b_{p-2};\beta)\;, $$

$$ {}_2F_1(a_1,a_2;b_1;\beta) = \frac{\Gamma(b_1)}{\Gamma(a_2)\Gamma(b_1-a_2)} \int_0^1 dz\,(1-z)^{-1-a_2+b_1}\,z^{-1+a_2}\,(1-\beta\,z)^{-a_1}\;. $$

Considering ${}_5F_4(a_1,\ldots,a_5;b_1,\ldots,b_4;\beta)$ with the values $a_1 = \epsilon, a_2 = -\epsilon, a_3 = -3\epsilon, a_4 = -5\epsilon, a_5 = -7\epsilon,\ b_1 = 2\epsilon, b_2 = 4\epsilon, b_3 = 6\epsilon, b_4 = 8\epsilon, \beta = 0.5$ we obtain the results shown in Table 5.7. The "analytic result" has been obtained using HypExp [150, 151].

ϵ order	analytic result	numerical result	time taken (secs)
ϵ^0	1	$1.0000002 \pm 4 \times 10^{-7}$	2
ϵ^1	0.189532	0.189596±0.00036	21
ϵ^2	-2.299043	-2.306±0.011	124
ϵ^3	55.46902	55.61 ±0.39	248
ϵ^4	-1014.39	-1018.4±5.9	429

Table 5.7: Results for ${}_5F_4(\epsilon,-\epsilon,-3\epsilon,-5\epsilon,-7\epsilon;2\epsilon,4\epsilon,6\epsilon,8\epsilon;\beta)$ at $\beta = 0.5$. The timings in the last column are the ones for the numerical integration. The time taken for decomposition, subtraction and ϵ-expansion was 11 seconds.

This result can be produced by typing *./launch -d demos -p param5F4.input -t template5F4.m* in the subdirectory `general`, or by typing *../launch -p param5F4.input -t template5F4.m* in the subdirectory `general/demos`.

The program can also deal with functions containing half integer exponents. Table 5.8 shows results for ${}_4F_3$ with arguments $a_1 = -4\epsilon, a_2 = -1/2-\epsilon, a_3 = -3/2-2\epsilon, a_4 = 1/2-3\epsilon, b_1 = -1/2+2\epsilon, b_2 = -1/2+4\epsilon, b_3 = 1/2+6\epsilon$. These results can be produced by the command *../launch -p param4F3.input -t template4F3.m* in the subdirectory `general/demos`.

ϵ order	analytic result	numerical result	time taken (seconds)
ϵ^0	1	$0.999997 \pm 1.7 \times 10^{-5}$	1.6
ϵ^1	-4.27969	-4.2810 ± 0.0055	54
ϵ^2	-26.6976	-26.625 ± 0.121	90

Table 5.8: Results for the hypergeometric function ${}_4F_3(-4\epsilon, -\frac{1}{2}-\epsilon, -\frac{3}{2}-2\epsilon, \frac{1}{2}-3\epsilon; -\frac{1}{2}+2\epsilon, -\frac{1}{2}+4\epsilon, \frac{1}{2}+6\epsilon; \beta)$ at $\beta = 0.5$

5.4.2.2 Phase space integrals

Sector decomposition can be useful for the calculation of phase space integrals where infrared divergences are regulated dimensionally. This is particularly the case for double real radiation occurring in NNLO calculations involving massive particles, where analytic methods show their limitations.

Here we give examples of $2 \to 3$ phase space integrals, which should be considered as part of a $2 \to n$ phase space written in factorised form. We choose particles 3 and 4 to be massless, while p_5 is the momentum of a massive state, either a single particle or a pseudo-state formed by n additional momenta $\tilde{p}_i$ in the final state, i.e. $p_5 = \sum_{i=5}^{n} \tilde{p}_i$. After all integrations have been mapped to the unit interval, we have integrals of the form

$$\begin{aligned} \int d\Phi_3 &= C_\epsilon \int \prod_{i=1}^{4} dx_i \, [x_1(1-x_1)x_2(1-x_2)]^{\frac{D-4}{2}} \, [x_3\,(1-x_3)]^{D-3} \\ &\quad [x_4\,(1-x_4)]^{\frac{D-5}{2}} \, [1-\beta\, x_3\,(1-x_2)]^{2-D} \ , \qquad (5.4) \\ \beta &= 1 - \frac{m^2}{s} \ , \ C_\epsilon = \frac{1}{(2\pi)^{2D-3}} d\Omega_{D-3} d\Omega_{D-4}\, s^{D-3} 2^{D-8} \beta^{2D-5} \ . \qquad (5.5) \end{aligned}$$

The derivation is given in the appendix, section I.1.

The invariants in this parametrisation are given by

$$\begin{aligned}
s_{13} &= -s\beta\, x_3\,(1-x_1) \\
s_{23} &= -s\beta\, x_3\, x_1 \\
s_{34} &= \beta\, K\, x_3\,(1-x_2)\ ,\ K = \frac{s\beta\,(1-x_3)}{1-\beta\, x_3\,(1-x_2)} \\
s_{35} &= s\,\frac{1-\beta(1-x_2x_3)}{1-\beta\, x_3(1-x_2)} \\
s_{14} &= -K\,\left\{t^- + x_4\,(t^+ - t^-)\right\} = -K\ \tilde{s}_{14} \\
s_{24} &= -K\,\left\{u^+ - x_4\,(u^+ - u^-)\right\} = -K\ \tilde{s}_{24}\ ,
\end{aligned}$$

where

$$\begin{aligned}
t^\pm &= \left(\sqrt{x_1(1-x_2)} \pm \sqrt{x_2(1-x_1)}\right)^2 \\
u^\pm &= \left(\sqrt{(1-x_1)(1-x_2)} \pm \sqrt{x_1x_2}\right)^2\ .
\end{aligned} \tag{5.6}$$

We would like to point out that for the examples below, more convenient parametrisations, i.e. parametrisations where the variables in the denominator factorise, and/or reflect symmetries of the squared matrix element, certainly do exist. However, the purpose of the examples is to illustrate that the code can deal with denominators which are amongst the most complicated ones which do occur in NNLO real radiation involving two (unresolved) massless particles in the final state, where they cannot always be "rotated away" by suitable transformations. A hybrid approach combining sector decomposition with convenient parametrisations/transformations is certainly the method of choice for real radiation at NNLO. The program can be used to evaluate the integrals occurring in such an approach.

Three massless particles in the final state

We first consider a case where p_5 is a massless particle, i.e. the limit $\beta \to 1$ in eq. (5.4). If we combine the phase space with the toy matrix element $1/(s_{35}s_{23})$, we have singularities at $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$. Such denominators come e.g. from the interference of diagrams as shown in figure 5.10.

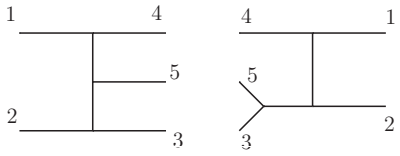


Figure 5.10: Interference of diagrams leading to factors of $s_{35}s_{23}$ in the denominator.

$$\int d\Phi_3 \frac{s^2}{s_{35}s_{23}} = C_\epsilon \int_0^1 \prod_{i=1}^{4} dx_i \left[(1-x_1)(1-x_2)\right]^{\frac{D-4}{2}} \left[x_1\, x_2\right]^{\frac{D-6}{2}} x_3^{D-5} \left(1-x_3\right)^{D-3} \left[x_4\,(1-x_4)\right]^{\frac{D-5}{2}} \left[1-\,x_3\,(1-x_2)\right]^{3-D} \,. \tag{5.7}$$

The term $[1-\,x_3\,(1-x_2)]^{3-D}$ goes to zero for $x_3 \to 1, x_2 \to 0$. Although $x_3 = 1$ does not lead to a singularity in the above example, for numerical stability reasons, and having in mind the presence of more complicated matrix elements than our toy example, it is preferable to transform this factor to an expression which is finite in the above limits. Splitting the x_3 integration at 1/2 and then doing sector decomposition achieves this goal. The program will do this automatically if the template file contains `splitlist={3}`, to tell the program that the integration over x_3 should be split at 1/2. Of course the singularities at $x_i = 0$ will also be extracted automatically.

Using the command *../launch -p params23s35.input -t templates23s35.m* in the subdirectory `general/demos`, sector decomposition leads to the result given in Table 5.9.

P_{-3}	P_{-2}	P_{-1}	P_0
-1.5705± 0.0005	-4.3530 ± 0.0025	1.712± 0.005	31.040± 0.014

Table 5.9: Results for the integral given by eq. (5.7). The factor C_ϵ is not included in the numerical result.

Two massless and one massive particles in the final state

The example in this subsection illustrates the program option to exclude certain parts of the integrand from the decomposition, even though they can become zero at certain values of the integration parameters. This can be useful if a particular term is known not to lead to a singularity. Note that terms with powers ≥ 0 are excluded from the decomposition by default.

As an example we pick an integral over s_{14}, where the line singularity has been remapped already (see appendix, section I.1).

$$\begin{aligned}\int d\Phi_3 \frac{s\,\beta}{s_{14}} &= 2\,C_\epsilon \int_0^1 dx_1 dx_2 dx_3 dx_4 [x_4\,(1-x_4)]^{\frac{D-5}{2}}\, x_3^{D-3}\,(1-x_3)^{D-4} \\ &\times [1-\beta\,x_3\,(1-x_1+x_1x_2)]^{3-D}\, x_1^{D-4}\, x_2^{D-5} \\ &\times [(1-x_1)(1-x_2)(1-x_1+x_1x_2)]^{\frac{D-4}{2}} \\ &\times \Big[(\sqrt{(1-x_1)(1-x_2)}-\sqrt{1-x_1+x_1x_2})^2 \\ &+ 4\,x_4\,\sqrt{(1-x_1)(1-x_2)(1-x_1+x_1x_2)}\Big]^{4-D}\,. \end{aligned} \tag{5.8}$$

Choosing the option "n" for "no decomposition" in the definition of the integrand for the term in square brackets $[\ldots]^{4-D}$ (see `templates14.m`), there will be no decomposition in the variables x_2, x_4, although this term vanishes in the limit $x_2, x_4 \to 0$, but this limit does not lead to a singularity. The result, which can be produced by *../launch -p params14.input -t templates14.m* in the subdirectory `general/demos`, is given in Table 5.10.

P_{-1}	P_0
-1.12635 ± 0.0003	-8.771 ± 0.003

Table 5.10: Numerical result for the integral given by eq. (5.8) for β=0.75. The factor $2\,C_\epsilon$ is not included in the numerical result.

5.4.2.3 Implicit Functions

This example demonstrates the ability to leave certain functions implicit until numerical integration. We want to integrate

$$f(x_1,x_2,x_3,x_4) = (x_1{+}x_2)^{-2-2\epsilon} x_3^{-1-4\epsilon} dum_1(x_1,x_2,x_3,x_4)^{1+\epsilon} dum_2(x_2,x_4)^{2-6\epsilon} cut(x_3) \tag{5.9}$$

with

$$dum_1(x_1,x_2,x_3,x_4) = 2 + x_1^2 + x_2^3 + x_3^4 + x_4^5 + 4x_1x_2x_3x_4 - x_1^2x_2^3x_3^4x_4^5 \tag{5.10}$$

$$dum_2(x_2,x_4) = x_2^2 + x_4^2 + \beta^2 + 4x_2x_4 - \sqrt{x_2x_4\beta} + 3x_2^2x_4^2 \tag{5.11}$$

Where β is a symbol defined in the parameter file. dum_1 and dum_2 are both > 0 for $\beta > 0$ within the integration region (and even if they were not, their exponents are such that they could only reduce the singular behaviour of the integrand), and so quantitative knowledge of their exact form is not required to guide the decomposition. Thus they can be left implicit, and only introduced at the numerical integration stage. The function $cut(x_3) \equiv \Theta(x_3 - cut3)$, where the value of $cut3$ is given in the parameter file. The command *../launch -p paramdummy.input -t templatedummy.m* from the folder `general/demos` runs this example. The fortran files containing the explicit form of the functions dum_1, dum_2, *cut* are found in `demos/testdummy`. Details on how to automatically produce these functions are found in appendix E.2 Results for $\beta = 0.8$ for $cut3 = 0, 0.1$ can be found in table 5.11.

cut3:	0	0.1
ϵ^{-2}	0.12890	0
ϵ^{-1}	-3.622	-1.241
ϵ^{0}	32.48	31.89
ϵ^{1}	-120.3	-139.5
ϵ^{2}	429	264

Table 5.11: Numerical result for the integral given by eqs. (5.9) - (5.10) for β=0.8

Phase Space Revisited

Let us reconsider eq. (5.7). One may wish to calculate this as a leading order contribution to $3j$ production, or an NLO contribution to $2j$ production. To this

end, we include a jet measurement function in the integrand, such that

$$min\left(s_{34}, s_{35}, s_{45}\right) \begin{cases} < & sy_{cut} \implies 2j \text{ event} \\ > & sy_{cut} \implies 3j \text{ event} \end{cases} \tag{5.12}$$

where y_{cut} is a dimensionless parameter. This is an implementation of the JADE algorithm, as described in eg [152]. This is not a realistic example, as the JADE algorithm has problems when applied to hadronic collisions, due in part to the fact that the overall partonic energy varies, and so it is difficult to define a dimensionless distance measure. However, it demonstrates how the use of a jet measurement function can be implemented in SecDec. This example can be run with the command *../launch -p params23s35JADE.input -t templates23s35JADE.m* from the `general/demos` directory. The results for various values of y_{cut} are given in table 5.12.

For the $3j$ result, a cut of $p_\perp \geq 0.05s$ for each jet (parton) has been applied, which removes singularities coming from collinear initial state radiation.

y_{cut}	order	$2j$	$3j$
0.01	ϵ^{-3}	-1.570	—
	ϵ^{-2}	-4.353	—
	ϵ^{-1}	32.31	—
	ϵ^{0}	284.7	136.12
0.03	ϵ^{-3}	-1.570	—
	ϵ^{-2}	-4.353	—
	ϵ^{-1}	18.21	—
	ϵ^{0}	153.1	87.49
0.1	ϵ^{-3}	-1.570	—
	ϵ^{-2}	-4.353	—
	ϵ^{-1}	7.068	—
	ϵ^{0}	66.12	32.66
0.15	ϵ^{-3}	-1.570	—
	ϵ^{-2}	-4.353	—
	ϵ^{-1}	4.437	—
	ϵ^{0}	48.24	17.20

Table 5.12: Contributions to NLO $2j$ and LO $3j$ toy process

Chapter 6

Conclusions and Outlook

We presented a number of important methods for calculating real and virtual corrections for cross-sections in QCD. The increase in difficulty from NLO to NNLO is vast, and extending methods that work well at NLO has proved very difficult. For multi-loop integrals, a number of analytical methods (Mellin-Barnes, differential equations, the DRA method,...) have proved successful for the evaluation of master integrals, but the difficulty of calculating these master integrals analytically greatly increases as the number of mass scales in the problem increase. For real radiative corrections, the method of antenna subtraction seems the most promising amongst the analytical methods presented for calculations at NNLO.

We then introduced the method of sector decomposition, and described how it is applicable to both virtual and real radiative corrections. Sector decomposition is particularly useful for higher order calculations, as algorithmically the procedure is unchanged as we increase from NLO to NNLO and beyond. Another fundamental difference between sector decomposition and the methods mentioned above is that analytic integration is not required.

In chapter 5 we discussed the publicly available computer code `SecDec`, which can be downloaded from `http://projects.hepforge.org/secdec/`. We gave a review of both existing and planned features, and many examples of how the program performs very well in a variety of calculations. We saw how `SecDec` displayed a significant improvement with respect to the time taken for calculation when compared with `FIESTA2`, with the example of a four-loop self-energy diagram. We

demonstrated how `SecDec` deals with implicit functions, and how this feature can be used to implement, for example, jet measurement functions. The example of ${}_4F_3$ showed how `SecDec` can perform subtractions with non-integer exponents. Results for four-loop vertex corrections and five-loop self-energy corrections have been calculated with `SecDec` [153, 154]. We gave preliminary results for calculations in the physical region via contour deformation. This contour deformation strategy provides a framework to calculate multi-loop amplitudes in the physical region, which will prove invaluable both as a test of analytic calculations, and to provide results where analytic methods have so far not been successful - particularly calculations containing many scales.

Appendix A

Numerical Stability

After decomposition, complicated diagrams/integrands can have multiple linear (or higher) poles. When these are subtracted, the resulting integrands, while formally well behaved for all integration variables tending to zero, can be numerically unstable. For example, consider the 1−dimensional integrand:

$$\int_0^1 dx\, x^{-2+\epsilon} f(x) \tag{A.1}$$

Where $f(x)$ is $O(1)$ as $x \to 0$. After subtraction, one is left with the integral:

$$\int_0^1 dx\, x^{-2+\epsilon} g(x), \quad g(x) = f(x) - f(0) - f'(0)x \tag{A.2}$$

$x^{-2}g(x)$ is formally $O(1)$ as $x \to 0$, but the denominator and numerator both $\to 0$, which can cause numerical instability. The way this is addressed in SecDec is via integration by parts in the following way: Let

$$I(a, b, g) = \int_0^1 dx\, x^{a+b\epsilon} g(x) \tag{A.3}$$

Where $g(x)$ is the integrand after subtraction, so $g(x) \sim O(x^{-a})$ as $x \to 0$ Applying integration by parts gives us

$$I(a, b, g) = \left[\frac{x^{a+1+b\epsilon}}{a+1+b\epsilon} g(x)\right]_0^1 - \frac{1}{a+1+b\epsilon} \int_0^1 dx\, x^{a+1+b\epsilon} g'(x) \tag{A.4}$$

The first term is $O(x)$ as $x \to 0$, and $g'(x) \sim O(x^{-a-1})$, so one can rewrite this as

$$I(a, b, g) = \frac{1}{a+1+b\epsilon} \left(g(1) - I(a+1, b, g')\right) \tag{A.5}$$

For the special case where $a = -1$, we have

$$I(-1, b, g) = \frac{1}{b\epsilon}\left(g(1) - \int_0^1 dx\, x^{b\epsilon} g'(x)\right) \tag{A.6}$$

Notice $g(1) \equiv \int_0^1 dx\, g'(x)$, as $g(x) \sim O(x)$ as $x \to 0$ and hence

$$I(-1, b, g) = \int_0^1 dx\, \frac{1 - x^{b\epsilon}}{b\epsilon} g'(x) = \int_0^1 dx(-\log(x) - ...)g'(x) \tag{A.7}$$

Thus numerical instabilities of the integrand are removed, with a trade-off of one power of $\log(x)$, which is easily integrated numerically. This method is trivial to extend to functions of more than one variable. Of course in practice this $1-$dimensional example would be coped with by the numerical integrator, but when there is more than one linear or higher pole, or even when there are many logarithmic poles, these numerical instabilities arise, and this method serves to remove them.

One drawback to this method is that expressions can become large when differentiated, but in fact there is a very neat solution to this. Notice that after subtraction, the function $g(x)$ is a sum of a number of terms. Let us write

$$g(x) = \sum_k h_k(x) \tag{A.8}$$

where, by construction, each $h_k \sim O(1)$ or higher as $x \to 0$. Let us also define the map

$$\tilde{I}(a, b, g) = \begin{cases} \frac{1}{a+1+b\epsilon}\left(g(1) - \tilde{I}(a+1, b, g')\right) & \text{for } a < -1, \\ \int_0^1 dx\, \frac{1-x^{b\epsilon}}{b\epsilon} g'(x) & \text{for } a = -1, \\ \tilde{I}(a, b, g) = \int_0^1 dx\, x^{a+b\epsilon} g(x) & \text{for } a > -1. \end{cases} \tag{A.9}$$

Notice that this map is well-defined for $g(x) \sim O(1)$ as $x \to 0$, and is linear in g. Notice also that for the stronger condition $g(x) \sim O(x^{-a})$ as $x \sim 0$,
$\tilde{I}(a, b, g) \equiv I(a, b, g)$. Hence one can write

$$I(a, b, g) = \tilde{I}(a, b, g) = \tilde{I}(a, b, \sum_k h_k(x)) = \sum_k \tilde{I}(a, b, h_k) \tag{A.10}$$

Thus the problematic large expressions can be broken down into many smaller ones, each of which can be evaluated separately.

There are of course other methods of dealing with these instabilities. One such method is using the Taylor expansion of the integrand in the region(s) where these instabilities occur. One advantage of this method is that no extra powers of $\log(x)$ are generated, and so the integrand behaves better for small x. However there are a number of drawbacks to this method:

- Extension to more than one linear or higher pole is less trivial. Eg, if x_1 and x_2 cause instabilities at 0, then expansions are needed for $x_1 \to 0$, $x_2 \to 0$, and $x_1, x_2 \to 0$
- The definition of where to use the expansion instead of the full function is ad hoc - in some more complicated examples, even a taking the expansion for $x < 0.5$ is still not enough to tame the instabilities.
- introducing an approximation to the function complicates error estimation - using IBP is exact.

Appendix B

Numerical Evaluation

Details of all parameters referred to in italics in this section can be found in Appendix C.

Once the algebraic part of the program is completed for a given integral, the structure of the result is

$$I\left(\frac{\mathcal{P}_u}{\mathcal{P}_d}\right) = \sum_n^r C_n \epsilon^n + \mathcal{O}(\epsilon^{r+1}) \tag{B.1}$$

$$C_n = \sum_i^{k_n} \int_0^1 d\vec{x} f_{n,i}(\vec{x}, \{\alpha\}) \tag{B.2}$$

with $\mathcal{P}_u$, $\mathcal{P}_d$ the user-defined and default prefactors respectively (`loop` only - see *prefactorflag*. For `general`, $\mathcal{P}_u = \mathcal{P}_d = 1$). k_n is the number of integrations for order ϵ^n, $\{\alpha\}$ is a set of parameters/invariants, and $f_{n,i}$ are integrable functions independent of ϵ. The order of the sum and integral sign in eq. (B.2) can be changed, and equally subsets of the functions $f_{n,i}$ can be summed and then integrated, such that the sum of these integrals is the required result. SecDec offers these options - to sum all functions and then integrate, select *togetherflag* = 1. This should only be used for relatively simple integrals, as the numerical integrator will struggle to map all the features of the integrand for a large sum of functions. This option leads to the error for C_n to be the error as given by the chosen numerical integrator. If *togetherflag* = 0, then the functions f_{n_i} arising from each different pole structure will be considered together, and the sum of the contributions from each of the present pole structures will be summed to produce C_n. Within each pole structure, there

may be a large number of functions to integrate, in which case these are summed in small groups (specifically, the functions are summed until the sum of the size of the functions exceeds a given value, *grouping* = 2000000 KB is default), and then integrated. In this case, The error for C_n is given by summing all the individual stated errors in quadrature.

The user should be aware that for complicated functions containing many subtractions, the Monte Carlo error estimate is not quite appropriate: it is calculated on a purely statistical basis, scaling like $1/\sqrt{N}$ if N is the number of sampling points. However, this is only a reliable error estimate under the assumption that the sampling has mapped all the important features of the function (i.e. all peaks) sufficiently precisely, and strictly is only valid for square integrable functions. If the function is not square integrable (but integrable), the Monte Carlo estimate for the integral will still converge to the true value, but the error estimate will become unreliable. For more involved integrals, we are faced with functions which have gone through numerous decompositions and subtractions, such that their shape in the unit hypercube is quite complicated, and therefore the naive Monte Carlo error estimate tends to underestimate the "true" error.

Often the main source of underestimated errors in the final result is the fact that there are a large number of integrations to sum, and so adding the errors in quadrature would only give a truly appropriate error estimate if there were no systematic errors in the numerical integration.

Notice that the accuracy requested in *acc1, acc2* (*epsrel, epsabs*) for BASES (Cuba), are the accuracies requested for each integration. So if there are large cancellations between functions in different integrations, the resulting error will be higher than that requested by the user, eg $(-112 \pm 1.12) + (113 \pm 1.13) = 1 \pm 1.5$. In these cases, it can be useful to use *togetherflag* = 1.

The above discussion deals with errors for C_n. To compute the error at each order in the final answer, we also need to include the multiply by the necessary prefactor, which mixes error contributions from various orders in ϵ. The errors are again summed in quadrature. Again, cancellations between different contributions can occur, leading to the requested accuracy not being reached.

Appendix C

Full Explanation of Parameters

There are many useful options available to the user of SecDec. Here we give details of these options.

C.1 Parameters Common to `loop` and `general`

- *subdir*: subdirectory where the directory containing all files relating to this calculation will be placed (if it does not exist, it will be created). If left blank, the directory containing the parameter file will be used.

- *outputdir*: The absolute path to the directory for all files relating to this calculation. This is only required if *subdir* is not defined. Useful for example to specify the `/scratch` directory for results files.

- *epsord*: The order in ϵ at which the Laurent series will be truncated. The default is 0. All orders $<$*epsord* will be calculated.

- *IBPflag*: This relates to the treatment of numerical instabilities. There are 3 options: '0' means that no additional treatment is applied, '1' means that integration by parts is always used, and '2' means that integration by parts is used only for pole structures deemed complicated enough. Option '2' is recommended (and is default), unless implicit functions are used (`general` only), in which case '0' should be used.

- *compiler*: The fortran compiler to be used. SecDec has been tested with gfortran (default), g77 and ifort.

- *exeflag*: flag to decide at which stage the program terminates:
  0 - the iterated sector decomposition is done and the scripts to do the subtraction and epsilon expansion and to create the fortran or C++ files and to launch the numerical integration are created (scripts batch* in *outputdir*) but not run (useful if a cluster is available to run each pole structure on a different node)
  1 - the subtraction and epsilon expansion are performed, and the resulting functions are written to fortran or C++ files
  2 - all the files needed for the numerical integration are created
  3 - compilation is launched to make the executables
  4 - the executables are run, either by batch submission or locally

- *clusterflag*: flag for job submission: '0' for single machine or '1'for a cluster.

- *batchsystem*: if a cluster is used, this specifies the syntax used.
  '0' indicates that PBS (Portable Batch System) is to be used, '1' indicates that a user-defined syntax is to be used.

- *maxjobs*: if using a cluster, the maximum number of jobs allowed in the queue simultaneously.

- *maxcput*: estimated time required for longest job. If using a cluster, this is used to specify which queue the jobs are placed in. If not using a cluster, this option can be omitted.

- *pointname*: The name of the numerical point to be calculated. This is useful when you want to calculate points with different parameters (invariants, user-defined symbols, numerical integration parameters etc). Can be left blank.

- *integrator*: Which numerical integration routine is to be used.
  '0' - BASES (NB BASES is not a valid option when using SecDec with C++) or from the Cuba library: '1' - Vegas, '2' - Suave, '3' - Divonne, '4' - Cuhre

- *basespath*: Absolute path for BASES library. Leave blank for `SecDec/basesv5.1`
- *cubapath*: Absolute path for Cuba library. Leave blank for `SecDec/Cuba`−2.1

The following are all BASES specific parameters. Each has suitable defaults and can safely be left blank if desired. They should be entered as a comma separated list, specifying the value to be used for the leading pole, subleading pole,...

- *ncall*: number of Monte Carlo points
- *acc1*: relative accuracy (as %) for MC grid construction.
- *acc2*: relative accuracy (as %) for MC integration step.
- *iter1*: maximum number of iterations for MC grid construction. Bases will only perform as many iterations as it needs to reach the desired accuracy at this stage, up to the maximum *iter1*.
- *iter2*: maximum number of iterations for MC integration step (*iter2* iterations are not always needed - see above)

The following are all Cuba specific parameters. Each has suitable defaults and can safely be left blank if desired. They should be entered as a comma separated list (except for *cubaflags*), specifying the value to be used for the leading pole, subleading pole,... Further detail can be found in `SecDec/Cuba`−2.1`/cuba.pdf`

- *maxeval*: maximum number of function evaluations.
- *mineval*: minimum number of function evaluations.
- *epsrel*: relative accuracy required (as decimal).
- *epsabs*: absolute accuracy required (as decimal).
- *cubaflags*: encodes verbosity, how samples are used, and how random numbers are generated.
- *nstart*: Vegas parameter - number of function evaluations on the first iteration.

- *nincrease*: Vegas parameter - number of extra function evaluations per iteration.
- *nnew*: Suave parameter - number of new function evaluations in each subdivision.
- *flatness*: Suave parameter - measure of how 'flat' the function to evaluate is (default is 1).
- *key1*: Divonne parameter - determines sampling in partition phase.
- *key2*: Divonne parameter - determines sampling in final integration phase.
- *key3*: Divonne parameter - sets strategy for refinement phase.
- *maxpass*: Divonne parameter - controls thoroughness of partitioning phase (essentially the number of 'safety' iterations performed before a partition is accepted).
- *border*: Divonne parameter - the width of the border of the integration region. Functions which cannot be evaluated on the border (most functions in SecDec have some logarithmic behaviour as some of the integration variables $\to 0$ and so cannot be explicitly evaluated at the border) need a non-zero border. Function values in this region are extrapolated from inside the integration region.
- *maxchisq*: Divonne parameter - maximum chi-squared value a single subregion is allowed to have in the final integration phase.
- *mindeviation*: minimum deviation (as decimal) of the requested error of the entire integral which determines whether it is worthwhile further examining a region which failed chi-squared test.
- *key*: Cuhre parameter - chooses basic integration rule

Further common parameters:

- *togetherflag*: Flag to integrate subsets of functions for each pole order separately and then sum them ('0'), or to sum all functions for a given order and then integrate ('1'). '1' will allow cancellations between functions and give a more accurate error estimate, but should not be used for complicated calculations.

- *editor*: Specifies which text editor the results file will be displayed in. If left blank, the results will not be automatically displayed

- *grouping*: The maximum size in bytes of the fortran/C++ functions to be grouped together for integration. Default is 2000000

- *seed*: Gives control over the seed used for random number generation by the numerical integration routines. For BASES, a value of 0 specifies that the same seed is used for each integration, and 1 specifies that the seed is different each time. For Cuba, *seed* specifies the value of 'seed' as documented in `SecDec/Cuba-2.1/cuba.pdf`

C.2 Parameters Specific to `loop`

- *graph*: Name of the graph to calculate. Can contain underscores and numbers, but not commas.

- *propagators*: Number of internal propagators in the diagram

- *legs*: Number of external legs

- *loops*: Number of loops

- *prefactorflag*: Flag for prefactor:
  '0' - default prefactor $(-1)^{Nn}\Gamma\left[Nn - Nloops * Dim/2\right]$ is factored out of the numerical result
  '1' - default prefactor is included in numerical result
  '2' - user-defined *prefactor* is factored out of the numerical result

- *prefactor*: if *prefactorflag*= 2 then define your desired prefactor here (in Mathematica syntax). NB use Nn, Nloops, Dim to denote number of propagators, loops and dimension ($4-2\epsilon$ by default)

- *sij*: Values for Mandelstam invariants $s_{ij} = (p_i + p_j)^2$. For the Euclidean region *sij* should all be <= 0. Should be a comma separated list, and the order should be s12,s23,s13,s14,s24,s34,s15,s25,s35,s45,s16,s26,s36,s45,s56. If the diagram to be calculated has < 6 legs then you can leave out any invariants you do not require.

- *pi2*: Off-shell legs $p_1^2, p_2^2, ...$ For the Euclidean region, $p_i^2 <= 0$. Should be a comma separated list.

- *ms2*: Masses $m_1^2, m_2^2, ...$ identified with $ms\,[1]\,, ms\,[2]\,, ...$ as entered in the template file. Should be a comma separated list, each entry >= 0.

- *primarysectors*: A comma separated list of primary sectors to be calculated. This option is particularly useful when a diagram has symmetries such that some primary sectors are equivalent. If left blank then primary sectors will be merged.

- *multiplicities*: the multiplicities of the primary sectors listed above (a comma separated list in the same order as *primarysectors*).

- *infinitesectors*: A comma separated list of sectors to be treated with a 'predecomposition' to sidestep infinite recursion. Default is left blank.

- *language*: Currently fortran and C/C++ are supported. Default is fortran, *language=Cpp* for C/C++

- *contourdef*: Whether or not contour deformation is to be performed to give numerical results in the physical region. Should take values of 'True' or 'False'. This option is not yet publicly available, and only supported in conjunction with *language=Cpp*

C.3 Parameters Specific to general

- *integrand*: Name of the integrand. Can contain underscores and numbers, but not commas.
- *symbols*: A comma separated list of symbols found in the integrand. This option lets you leave the values of certain parameters in your calculation symbolic at the algebraic stage, and only when the numerical integration is performed are these defined.
- *values*: A comma separated list giving the numerical values of the *symbols* (must be in the same order)
- *dummys*: A comma separated list of any functions in the integrand which are left implicit at the algebraic stage.

Appendix D

Template Files

D.1 `loop` Template Files

For the `loop` program, there are a number of required inputs for the template file. The following are a list of these inputs, together with the values that they take for the massless scalar non-planar two-loop box:

- *momlist*: a list of the loop momenta in the diagram
  momlist=$\{k1, k2\}$;

- *proplist*: a list of the propagators in the diagram
  proplist=$\{k1^2, (k1+p2)^2, (k1-p1)^2, (k1-k2)^2, (k2+p2)^2, (k1-k2-p1-p2-p3)^2, (k2+p2+p3)^2\}$;

- *numerator*: a list representing the numerator to include in the calculation. This is optional, and the default is $\{1\}$. For example, to represent $(k_1.k_2)(k_1.p1)$ you would have *numerator*=$\{k1\ k2, k1\ p1\}$. For the scalar integral example it is left blank.

- *powerlist*: a list of the powers of the propagators in the diagram. This can be left out, and defaults to $\{1, 1, ..., 1\}$ (ie each propagator is raised to the first power).

- *onshell*: this should be a list of replacement rules for the diagram. By default the program will relabel $p_i^2 \to ssp[i]$, and $p_i.p_j \to (sp[i,j] - ssp[i] - ssp[j])/2$,

so if you chose a different naming convention for your external momenta you should specify similar replacements here. For the massless non-planar box, we need the replacements
onshell=$\{ssp[\_] \to 0, ms[\_] \to 0\}$, ie all external legs and propagators are massless.
It should be noted that if any of your invariants are zero, this should be stated in the *onshell* replacements, and not only in the parameter file, as the decomposition assumes all symbols are non-zero. You may also specify the non-zero numerical values for your invariants here, however these values will supersede any values subsequently entered in the parameter file, and as such you would have to run the algebraic part of the calculation again if you wished to calculated with a different set of invariants.

- *Dim*: dimension of spacetime. If left out, default is
  Dim=$4 - 2 * eps$;

D.2 general Template Files

For the general program, there are a number of inputs required for the template file. The following are a list of these inputs for the integrand
$(x_1(1-x_1))^{-1-\epsilon}(x_2+x_3+x_4)^{2+\epsilon}(x_1x_3-\beta x_2x_4)^{-1-3\epsilon}$

- *nmax*: the number of integration variables
  nmax= 4;
- *intvars*: a list of the integration variables *intvars*=Table$[x[i], \{i, nmax\}]$;
- *factorlist*: a representation of the integrand as a list of functions and exponents
  factorlist=$\{\{x[1], -1-eps\}, \{1-x[1], -1-eps\}, \{x[2]+x[3]+x[4], 2+eps\}, \{x[1]x[3]-x[2]x[4]beta, -1-3eps\}\}$;
- *splitlist*: a list of any variables which can cause singularities at both zero and one.
  splitlist=$\{1\}$;

The template file is processed by Mathematica, so you are free to use any Mathematica syntax you like as long as these required inputs are defined in the way as stated above. In the current example notice that β must be given a value of < 1 in the parameter file, otherwise a singularity will occur inside the integration region.

Appendix E

Advanced Usage

The following features were not included in the original release of SecDec. They are in the most recent version which can now be downloaded from `http://projects.hepforge.org/secdec/`, but are not as yet in the CPC version.

E.1 Automating the Calculation of Multiple Numerical Points

(`general` only) If an integrand contains one or more user-defined parameters then the algebraic stage need only be done once for the calculation of many different numerical points. It is desirable to automate the calculation of many numerical points to minimize the effort for the user. This is done using the perl script `multinumerics.pl`. The user must create a text file **multiparamfile** in **myworkingdir**, and specify a number of options:

- *paramfile=myparamfile*: Specify the name of the parameter file.
- *pointname=myprefix*: Points calculated will have the names *myprefix1, myprefix2,...*
- *lines*: The number of points you wish to calculate - if omitted all points listed will be calculated.

Following these options, specify the values of the parameters for each point you wish to calculate. For example, if you had two parameters in your integrand, a and b (which appear in that order as *symbols=a,b* in `myparamfile`), and you wished to calculate the points $(a, b) = (0.1, 0.1), (0.2, 0.4), (0.3, 0.9)$ then the inputs in `multiparamfile` for this would be:

0.1,0.1

0.2,0.4

0.3,0.9

Furthermore, one may wish to calculate the integrand for values of parameters at incremental steps. This is allowed, and the syntax is as follows: Suppose you wish to calculate each combination of $a = 0.1, 0.2, 0.3$ and $b = 0.1, 0.3, 0.5, 0.7$. The input for this is

minvals=0.1,0.1

maxvals=0.3,0.7

stepvals=0.1,0.2

Of course, one might want non-constant step values. Eg to calculate every combination of $a = 0.1, 0.2, 0.4$, $b = 0.1, 0.3, 0.6$ the syntax would be:

values1=0.1,0.2,0.4

values2=0.1,0.3,0.6

Please note that *values1* must appear before *values2* in `multiparamfile`.

Each of these different syntaxes can be used together - please see

`general/demos/multiparam.input` for an example of this.

In order to execute the script, the Mathematica-generated functions must already be in place. The simplest way to do this is to run the `launch` script, with *exeflag=1* in your parameter file. Then from a terminal, from the `general` directory, issue the command *'./multinumerics.pl [-d myworkingdir -p multiparamfile]'*. If you are using single-machine mode (*clusterflag=0*) then all integrations will be performed, and the results collated and output as files in the directory specified in `myparamfile`. If not, then you will need to run the script again, with the argument '1', to collect the results. ie *'./multinumerics.pl 1 [-d myworkingdir -p multiparamfile]'*. The script generates a parameter file for each numerical point calculated. To remove these in-

termediate parameter files (your original *myparamfile* will not be removed), issue the command *'./multinumerics.pl 2 [-d myworkingdir -p multiparamfile]'*. This should only be done after the results have been collated.

E.2 Leaving Functions Implicit During Algebraic Calculation

(`general` only) There are a number of reasons why one might want to leave functions implicit during the algebraic stage. If you have a large but finite function the algebraic part of the calculation will be quicker and produce much smaller intermediate files if this function is left implicit. Also one might like to use a number of different eg. measurement functions, which this option allows. To implement this, write your template file as usual, but where desired use a function which is left undefined, and list this function under the option *dummys* in your parameter file. Note that you may use more than one implicit function at a time, and that these functions can have any number of arguments. If you also have symbolic parameters, these do not need to be arguments of your implicit function.

Once you have set up your template and parameter files, you will need to define these functions explicitly so that they can be used in the calculation. The simplest way to do this is to prepare a Mathematica syntax file for each implicit function specified, and place them in the *outputdir* specified in your parameter file. Suppose you have a function named *dum1*, a function of 2 variables, defined as $dum1(x_1, x_2) = 1+x_1+x_2$. Create a file `dum1.m`, and insert the lines:

$intvars = \{z1, z2\}$;

$dum1 = 1 + z1 + z2$;

where $z1, z2$ can be replaced by any variable names you wish, as long as they are used consistently in `dum1.m`. Notice that for every function specified in *dummys* in your parameter file, there must be a Mathematica file `dummyname1.m` with the correct name and syntax in the results directory. Once these Mathematica files are in place, issue the command

'*createdummyfortran.pl [-d myworkingdir -p myparamfile]*' from the `general` direc-

tory. This generates the fortran files for the functions you defined, which are found in the same subdirectory as the originals.

Of course you might prefer to write these fortran files yourself instead of having them generated for you. This is certainly possible, however we recommend that you use this perl script to generate functions with the necessary declarations and then edit these.

An example of this can be found in `general/demos`, with the files `paramdummy.input`, `templatedummy.m`, and the directory `/testdummy`

Appendix F

Contour Deformation

Sector decomposition extracts regularised (UV/IR) singularities from loop integrals. There are, however, other singularities which can appear in these calculations; they are unregularised, integrable singularities which depend on the kinematic invariants in the problem. Contour deformation is a method to evaluate these integrals numerically by realising the $i\delta$ prescription.

This method relies on Cauchy's theorem, which states that the integral of an analytic function, ie one which satisfies the Cauchy-Riemann equations

$$f(x+iy) = u(x+iy) + iv(x,iy), \quad \frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{F.1}$$

over a closed contour is zero, provided that contour does not enclose a singularity. That is

$$\oint_{\mathcal{C}} f(z)dz = 0$$

In figure F.1 we see that by Cauchy's theorem

$$\oint_{\mathcal{C}_0-\mathcal{C}_1} f(z)dz = 0 \implies \oint_{\mathcal{C}_0} f(z)dz = \oint_{\mathcal{C}_1} f(z)dz$$

However

$$\oint_{\mathcal{C}_0-\mathcal{C}_2} f(z)dz \neq 0 \implies \oint_{\mathcal{C}_0} f(z)dz \neq \oint_{\mathcal{C}_2} f(z)dz$$

Since the closed contour $\mathcal{C}_0 - \mathcal{C}_2$ encloses a singularity of the integrand $f(z)$.

Let us consider a sector after decomposition, such that all end point singularities have been removed. The integral for this sector is of the form

$$I_s = C(\epsilon) \lim_{\delta\to 0} \int_0^1 \frac{dx_1 \cdots dx_n x_1^{-a_1+b_1\epsilon} \cdots x_n^{-a_n+b_n\epsilon} \mathcal{U}_s(\vec{x})^{N_\nu-(L+1)D/2-R} \mathcal{N}_s(\vec{x})}{[\mathcal{F}_s(\vec{x}, m_i^2, s_{jk}) - i\delta]^{N_\nu-LD/2-m}} \tag{F.2}$$

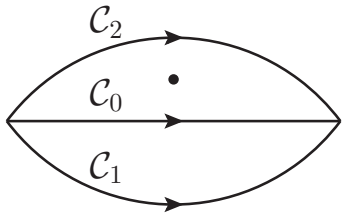


Figure F.1: 3 different contours of integration. The dot represents a singularity in the integrand $f(z)$

With L, D, R, m as defined in eq. (4.3.1.1), n is the number of propagators minus one, m_i^2, s_{jk} are masses/external invariants, and $\mathcal{N}_s, \mathcal{U}_s, \mathcal{F}_s$ are the decomposed functions relating to the original $\mathcal{N}, \mathcal{U}, \mathcal{F}$ for sector s. For ease of notation, let us introduce $\mathcal{U}_s(\vec{x})^{N_\nu-(L+1)D/2-R}\mathcal{N}_s(\vec{x}) \equiv \mathcal{H}_s$, $N_\nu - LD/2 - m \equiv a + b\epsilon$, and $dx_1 \cdots dx_n x_1^{-a_1+b_1\epsilon} \cdots x_n^{-a_n+b_n\epsilon} \equiv \mathcal{D}(\vec{x}, \epsilon)$ ie

$$I_s = C(\epsilon) \lim_{\delta \to 0} \int_0^1 \frac{\mathcal{D}(\vec{x}, \epsilon)\, \mathcal{H}_s(\vec{x}, \epsilon)}{[\mathcal{F}_s(\vec{x}, m_i^2, s_{jk}) - i\delta]^{a+b\epsilon}} \tag{F.3}$$

We wish to construct a contour of integration such that the imaginary part of $\mathcal{F}_s$ is negative. To this end, let us consider the contour $\mathcal{C}$ parametrised by

$$z_i \;=\; x_i - i\lambda x_i^\alpha (1-x_i)^\beta \frac{\partial \mathcal{F}_s}{\partial x_i} \tag{F.4}$$

As long as no poles are crossed in deforming from the unit hypercube to contour $\mathcal{C}$, we have

$$\lim_{\delta \to 0} \int_0^1 \frac{\mathcal{D}(\vec{x}, \epsilon)\, \mathcal{H}_s(\vec{x}, \epsilon)}{[\mathcal{F}_s(\vec{x}, m_i^2, s_{jk}) - i\delta]^{a+b\epsilon}} = \int_{\mathcal{C}} \frac{\mathcal{D}(\vec{z}, \epsilon)\, \mathcal{H}_s(\vec{z}, \epsilon)}{[\mathcal{F}_s(\vec{z}, m_i^2, s_{jk})]^{a+b\epsilon}} \tag{F.5}$$

This choice of contour guarantees that, for small λ, $\mathcal{F}_s$ has a negative $O(\lambda)$ imaginary part

$$\mathcal{F}_s(\vec{z}) = \mathcal{F}_s(\vec{x}) - i\lambda \sum_{i=1}^{n} x_i^\alpha (1-x_i)^\beta \left(\frac{\partial \mathcal{F}_s}{\partial x_i}\right)^2 + O(\lambda^2) \tag{F.6}$$

Where the $O(\lambda^2)$ term is purely real, and further imaginary terms appear at $O(\lambda^3)$. In principle one could add higher order terms in λ to eq. (F.4) to cancel the $O(\lambda^3), O(\lambda^5), ...$ terms, but in practice it is sufficient to choose a small value for λ such that the higher order terms are suppressed. Performing the transformation

of variables as defined in eq. (F.4), and using notation

$$\mathcal{I}_s(\vec{x},\epsilon) \equiv \mathcal{J}(\vec{x}\to\vec{z}(\vec{x}))\frac{\mathcal{H}_s(\vec{z}(\vec{x}),\epsilon)}{[\mathcal{F}_s(\vec{z}(\vec{x}),m_i^2,s_{jk})]^{a+b\epsilon}} \tag{F.7}$$

with $\mathcal{J}$ the Jacobean of the transformation gives us

$$I_s = C(\epsilon)\int_0^1 \prod_{i=1}^{n} dx_i x_i^{-a_i+b_i\epsilon}\left(\frac{z_i}{x_i}\right)^{-a_i+b_i\epsilon}\mathcal{I}_s(\vec{x},\epsilon) \tag{F.8}$$

Note that $\left(\frac{z_i}{x_i}\right) = 1 - i\lambda x_i^{\alpha-1}(1-x_i)^{\beta}\frac{\partial\mathcal{F}_s}{\partial x_i}$, so we take $\alpha \geq 1$, $\beta > 0$ to both fix the end points of the contour, and to ensure that there are no end point singularities in $\prod_{i=1}^{n}\left(\frac{z_i}{x_i}\right)^{-a_i+b_i\epsilon}\mathcal{I}_s(\vec{x},\epsilon)$. This is then the starting point for subtraction and expansion as a Laurent series in ϵ.

It should be noted that α,β,λ can be different for each variable, ie $\alpha_i,\beta_i,\lambda_i$. The choice of λ will have an effect on the convergence of the integral, and in general it should be small wrt any invariants in the calculation. To this end, one invariant should be factored out of the calculation. If the ratio of some invariants is large, then choosing a suitable value of the λ_i is not straightforward. One way to do this is to follow the method given in [155], whereby a small number of sample points for each of the

$$\tau_i = x_i^{\alpha}(1-x_i)^{\beta}\left(\frac{\partial\mathcal{F}_s}{\partial x_i}\right)$$

is taken, and then

$$\lambda_i = \frac{1}{max(\tau_i)}$$

is used for the integration.

Appendix G

Timings for a four-loop two-point diagram

In order to give an idea about the timings for a complicated example which we ran on several processors, we give the timings for the four-loop graph shown in figure G.1. The coefficient of each pole order is composed of a number of functions which can be

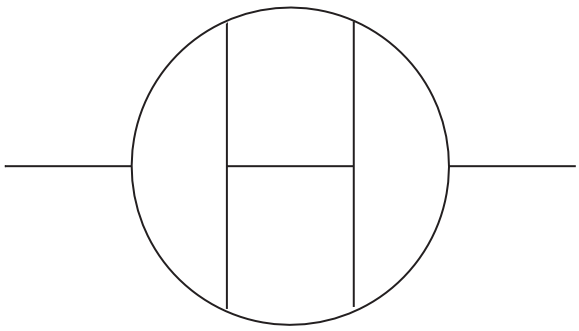


Figure G.1: A four-loop two-point master integral

integrated individually, such that the time taken for the longest job equals the total time for a given pole order, provided that the contributing functions are integrated in parallel. The files to run the integration of these functions in parallel are created by the program automatically. The number of integrations to run individually depends on the size and number of the regular subsector functions contributing at each pole order: these functions are summed until their sum reaches about one Megabyte, and then integrated individually, preferably in parallel.

Stage	Time for longest job	# integrations	Total time
Sector Decomposition	615		2160
Subtraction & ϵ-expansion	809		3767
Numerical integration ϵ^{-1}	156	5	508
Numerical integration ϵ^{0}	422	28	4720
Numerical integration ϵ^{1}	492	28	5946
Numerical integration ϵ^{2}	2172	29	8123

Table G.1: Timings (in seconds) for the diagram shown in figure G.1. The time taken for the longest job equals the total time for a given pole order if the contributing functions are integrated in parallel. The number of sampling points was 500000 for each pole order. The last column shows the timings which would result from a calculation in series.

The timings are listed in Table G.1. For information we also give the timings which would result from a serial calculation in the third column of Table G.1. The results are shown in Table G.2. Results for FIESTA 2 are also included. The total time taken to produce the FIESTA 2 result was 8.9 days, in comparison with 7 hours on a single core with SecDec. The analytical result can be found in [97], [42].

Order	Analytical result	SecDec	FIESTA 2
ϵ^{-1}	-10.3692776	-10.371±0.002	-10.36941±0.00011
ϵ^{0}	-70.99081719	-71.002±0.013	-70.989±0.002
ϵ^{1}	-21.663005	-21.65±0.12	-21.633±0.023
ϵ^{2}	2832.67	2833.79±0.92	2832.86±0.17

Table G.2: "Analytical" and numerical results for the diagram shown in figure G.1.

In this calculation, the symmetry of the problem was used, so only four primary sectors were evaluated. The corresponding multiplicities of the primary sectors are taken into account automatically, provided they are specified in `param.input`.

Appendix H

Another representation of the non-planar two-loop box

Here we derive a representation of the non-planar two-loop box where one integration parameter factorises naturally, such that it can be integrated out analytically, leaving a representation which can be evaluated in a completely automated way by the routines in `SecDec/general`, the evaluation being considerably faster than the one of example 5.4.1.1. This procedure is not limited to our particular example, but requires an analytical step of introducing a convenient parametrisation.

The expression for the non-planar two-loop box shown in figure 5.5 is given by

$$G_{NP} = \int \frac{d^D k\, d^D l}{(i\pi^{\frac{D}{2}})^2} \frac{1}{k^2 (k+p_2)^2 (k-p_1)^2 (k-l)^2 (l+p_2)^2 (k-l+p_4)^2 (l+p_2+p_3)^2} \,. \tag{H.1}$$

Considering first the integration over the loop momentum l only, we have a one-loop box as shown in figure H.1 with $P_1 = p_1 - k, P_2 = p_2 + k$. Feynman parametrisation for this one-loop subgraph leads to

$$\begin{aligned} I_1 &= \int \frac{d^D l}{i\pi^{\frac{D}{2}}} \frac{1}{(k-l)^2 (l+p_2)^2 (k-l+p_4)^2 (l+p_2+p_3)^2} \\ &= \Gamma(2+\epsilon) \int \prod_{i=1}^{4} dx_i\, \delta(1 - \sum_{j=1}^{4} x_j)\, \mathcal{F}(\vec{x},k)^{-2-\epsilon} \qquad \text{(H.2)} \\ \mathcal{F}(\vec{x},k) &= -(p_1+p_4-k)^2\, x_1 x_3 - (p_1+p_3-k)^2\, x_2 x_4 \\ &\quad -(k+p_2)^2\, x_1 x_2 - (p_1-k)^2\, x_3 x_4 \,. \end{aligned}$$

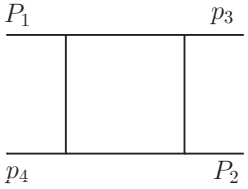


Figure H.1: The "inner" box as part of the non-planar two-loop box shown in figure 5.5.

Now we substitute

$$x_1 = t_2\,(1-t_3)\,,\; x_2 = t_1 t_3\,,\; x_3 = (1-t_1)\,t_3 \tag{H.3}$$

and integrate out the δ-constraint to obtain

$$\begin{aligned} I_1 &= \Gamma(2+\epsilon)\int_0^1 dt_1 dt_2 dt_3\,[t_3\,(1-t_3)]^{-1-\epsilon} \\ &\times \big[-(p_1+p_4-k)^2\,t_2\bar{t}_1 - (p_1+p_3-k)^2\,t_1\bar{t}_2 \\ &\quad -(k+p_2)^2\,t_1 t_2 - (p_1-k)^2\,\bar{t}_1\bar{t}_2\big]^{-2-\epsilon} \\ &= \frac{2\,\Gamma(2+\epsilon)\Gamma(-\epsilon)\Gamma(1-\epsilon)}{\Gamma(1-2\epsilon)}\int_0^1 dt_1 dt_2 \big[-(p_1+p_4-k)^2\,t_2\bar{t}_1 \\ &\quad -(p_1+p_3-k)^2\,t_1\bar{t}_2 - (k+p_2)^2\,t_1 t_2 - (p_1-k)^2\,\bar{t}_1\bar{t}_2\big]^{-2-\epsilon}\,, \end{aligned} \tag{H.4}$$

where we used the shorthand notation $\bar{t}_i = 1 - t_i$. Now we combine the above expression with the remaining propagators, treating the expression in square brackets in eq. (H.4) as a fourth propagator with power $2+\epsilon$. One can use the `SecDec` code to calculate the resulting function integrand function $\mathcal{F}_2$, although this is an easy calculation to do by hand. One obtains

$$\begin{aligned} G_{NP} &= \frac{-2\,\Gamma(3+2\epsilon)\Gamma(-\epsilon)\Gamma(1-\epsilon)}{\Gamma(1-2\epsilon)} \\ &\times \int_0^1 dt_1 dt_2 \int \prod_{i=1}^4 dz_i\, z_4^{1+\epsilon}\delta(1-\sum_{j=1}^4 z_j)\,\mathcal{F}_2(\vec{z})^{-3-2\epsilon} \\ \mathcal{F}_2(\vec{z}) &= -s_{12}\,z_2 z_3 - T\,z_1 z_4 - P_3^2\,z_2 z_4 - P_4^2\,z_3 z_4\,, \\ &\text{where} \\ T &= s_{23}\,t_2(1-t_1) + s_{13}\,t_1(1-t_2)\,,\; s_{ij} = (p_i+p_j)^2\,, \\ P_3^2 &= s_{12}\,(1-t_1)(1-t_2)\,,\; P_4^2 = s_{12}\,t_1 t_2\,. \end{aligned} \tag{H.5}$$

Appendix H. Another representation of the non-planar two-loop box105

With the substitutions

$$z_4 = t_3 t_4\,,\; z_3 = t_3\,(1-t_4)\,,\; z_2 = (1-t_3)\,t_5\,,\; z_1 = (1-t_3)\,(1-t_5) \tag{H.6}$$

we finally obtain

$$\begin{aligned} G_{NP} &= \frac{-2\,\Gamma(3+2\epsilon)\Gamma(-\epsilon)\Gamma(1-\epsilon)}{\Gamma(1-2\epsilon)} \\ &\times \int_0^1 dt_1 dt_2 dt_3 dt_4 dt_5\, t_3^{-1-\epsilon}(1-t_3)\, t_4^{1+\epsilon}\, \left[\,\mathcal{F}_2(\vec{t})\,\right]^{-3-2\epsilon} \\ \mathcal{F}_2(\vec{t}) &= -s_{12}\,\bar{t}_3\bar{t}_4 t_5 - T\,\bar{t}_3 t_4 \bar{t}_5 - P_3^2\,\bar{t}_3 t_4 t_5 - P_4^2\, t_3 t_4 \bar{t}_4\,,\;\; \bar{t}_i = 1 - t_i\,. \end{aligned}$$

In this form the integrand can be fed into the sector decomposition routine for "general integrands". The corresponding template file `templatexbox.m` can be found in `SecDec/general/demos`.

Appendix I

Phase Space Parametrisations

I.1 A $2 \to 3$ Phase Space Parametrisation

The $D-$dimensional phase space for $p_1 + p_2 \to p_3 + p_4 + p_5$ is given by

$$\begin{aligned} d\Phi_3 &= \Big[\prod_{j=3}^{5} \frac{d^D p_j}{(2\pi)^{D-1}}\Big]\, \delta^+(p_3^2 - m_3^2)\delta^+(p_4^2 - m_4^2)\delta^+(p_5^2 - m_5^2) \\ &\times \quad (2\pi)^D \delta^{(D)}\Big(p_1 + p_2 - \sum_{j=3}^{5} p_j\Big)\,. \end{aligned}$$

Using

$$\begin{aligned} \int d^D p_j\, \delta^+(p_j^2 - m_j^2) &= \int d^{D-1}\vec{p}_j\, dE_j\, \delta(E_j^2 - \vec{p}_j^{\,2} - m_j^2)\, \theta(E_j) \\ &= \frac{1}{2}\int dE_j\, d\Omega_{D-2}^{(j)}\, |\vec{p}_j|^{D-3}\Big|_{|\vec{p}_j| = \sqrt{E_j^2 - m_j^2}}\,, \\ \int d\Omega_{D-2} &= \frac{2\pi^{\frac{D-1}{2}}}{\Gamma(\frac{D-1}{2})}\,, \end{aligned}$$

and eliminating p_5 by momentum conservation, one obtains

$$\begin{aligned} \int d\Phi_3 &= \frac{1}{(2\pi)^{2D-3}}\frac{1}{4}\int dE_3\, dE_4\; d\Omega_{D-2}^{(3)}\, |\vec{p}_3|^{D-3}\, d\Omega_{D-2}^{(4)}\, |\vec{p}_4|^{D-3} \\ &\quad \delta((p_1 + p_2 - p_3 - p_4)^2 - m_5^2)\,. \end{aligned} \tag{I.1}$$

Having NNLO phase spaces in mind, let us assume that we have two massless particles in the final state (which may become unresolved), so $m_3^2 = m_4^2 = 0$, and p_5 is the momentum of a massive state, $m_5^2 = m^2$, either a single particle or a pseudo-state formed by n additional momenta $\tilde{p}_i$ in the final state, i.e. $p_5 = \sum_{i=5}^{n} \tilde{p}_i$. In

this case we can parametrise the momenta $p_1 \dots p_4$ as

$$\begin{aligned} p_1 &= \frac{\sqrt{s}}{2}\,(1, \vec{0}^{(D-2)}, 1) \\ p_2 &= \frac{\sqrt{s}}{2}\,(1, \vec{0}^{(D-2)}, -1) \\ p_3 &= E_3\,(1, \vec{0}^{(D-4)}, \sin\eta\sin\theta_1, \cos\eta\sin\theta_1, \cos\theta_1) \\ &= E_3\,(1, \vec{0}^{(D-4)}, \vec{n}_3) \\ p_4 &= E_4\,(1, \vec{0}^{(D-4)}, \vec{n}_4) \end{aligned} \tag{I.2}$$

where we choose $\vec{n}_4$ such that

$$\begin{aligned} \vec{n}_4 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} \sin\phi\sin\theta_2 \\ \cos\phi\sin\theta_2 \\ \cos\theta_2 \end{pmatrix} \\ &= \begin{pmatrix} \sin\phi\sin\theta_2 \\ \cos\theta_1\cos\phi\sin\theta_2 + \sin\theta_1\cos\theta_2 \\ \cos\theta_1\cos\theta_2 - \sin\theta_1\cos\phi\sin\theta_2 \end{pmatrix} \end{aligned}$$

Thus

$$\begin{aligned} d\Omega^{(3)}_{D-2} &= d(\cos\theta_1)\,(1-\cos^2\theta_1)^{\frac{D-4}{2}}\,d\Omega^{(\eta)}_{D-3} \\ d\Omega^{(4)}_{D-2} &= d(\cos\theta_2)\,(1-\cos^2\theta_2)^{\frac{D-4}{2}}\,d\phi\,(\sin\phi)^{D-4}\,d\Omega_{D-4}\,. \end{aligned} \tag{I.3}$$

Due to overall rotational invariance around the beam axis, we can integrate out the azimuthal angle η and use $\eta = 0$ in eq. (I.2), leading to $\vec{n}_3 \cdot \vec{n}_4 = \cos\theta_2$. We obtain

$$\begin{aligned} d\Phi_3 = \; & \frac{1}{4}\frac{1}{(2\pi)^{2D-3}} d\Omega_{D-3} d\Omega_{D-4}\, d(\cos\theta_1)\, d(\cos\theta_2)\, d(\cos\phi)\, dE_3\Theta(E_3)\, dE_4\Theta(E_4) \\ & (E_3E_4)^{D-3}\left(\sin^2\theta_1\,\sin^2\theta_2\right)^{\frac{D-4}{2}}(\sin^2\phi)^{\frac{D-5}{2}} \\ & \delta(m^2 - (p_1+p_2-p_3-p_4)^2) \end{aligned} \tag{I.4}$$

$$\delta(m^2 - (p_1+p_2-p_3-p_4)^2) = \delta(s - m^2 - 2\sqrt{s}\,(E_3+E_4) + 2\,E_3E_4(1-\vec{n}_3\cdot\vec{n}_4))\,.$$

Now we have to choose which variable to eliminate using the δ-constraint. In our example, we will eliminate E_4, leading to

$$E_4 = \frac{s - m^2 - 2\sqrt{s}\,E_3}{2(\sqrt{s} - E_3\,(1-\vec{n}_3\cdot\vec{n}_4))}\,. \tag{I.5}$$

The constraint $\Theta(E_4)$ leads to

$$E_3^{\max} = \frac{s-m^2}{2\sqrt{s}} = \frac{\sqrt{s}}{2}\,\beta \;\text{ where } \beta = 1 - \frac{m^2}{s}\;. \tag{I.6}$$

We substitute

$$\begin{aligned} E_3 &= \frac{\sqrt{s}}{2}\,\beta\, x_3 \Rightarrow E_4 = \frac{\sqrt{s}}{2}\,\beta\,\frac{1-x_3}{1-\beta\, x_3(1-x_2)} \\ \cos\theta_1 &= 2\,x_1 - 1\;,\; \cos\theta_2 = 2\,x_2 - 1\;,\; \cos\phi = 2\,x_4 - 1 \end{aligned}$$

to obtain

$$\begin{aligned} d\Phi_3 &= \frac{1}{(2\pi)^{2D-3}} d\Omega_{D-3} d\Omega_{D-4}\, s^{D-3} 2^{D-8} \beta^{2D-5} \\ &\quad \prod_{i=1}^{4} dx_i\; [x_1(1-x_1)x_2(1-x_2)]^{\frac{D-4}{2}}\; [x_3\,(1-x_3)]^{D-3} \\ &\quad [x_4\,(1-x_4)]^{\frac{D-5}{2}}\; [1-\beta\, x_3\,(1-x_2)]^{2-D}\;. \end{aligned} \tag{I.7}$$

The invariants in this parametrisation are given by

$$\begin{aligned} s_{13} &= -s\beta\, x_3\,(1-x_1) \\ s_{23} &= -s\beta\, x_3\, x_1 \\ s_{34} &= \beta\, K\, x_3\,(1-x_2)\;,\; K = \frac{s\beta\,(1-x_3)}{1-\beta\, x_3\,(1-x_2)} \\ s_{35} &= s\,\frac{1-\beta(1-x_2x_3)}{1-\beta\, x_3(1-x_2)} \\ s_{14} &= -K\,\left\{t^- + x_4\,(t^+ - t^-)\right\} = -K\;\tilde{s}_{14} \\ s_{24} &= -K\,\left\{u^+ - x_4\,(u^+ - u^-)\right\} = -K\;\tilde{s}_{24}\;, \end{aligned}$$

where

$$\begin{aligned} t^{\pm} &= \left(\sqrt{x_1(1-x_2)} \pm \sqrt{x_2(1-x_1)}\right)^2 \\ u^{\pm} &= \left(\sqrt{(1-x_1)(1-x_2)} \pm \sqrt{x_1x_2}\right)^2\;. \end{aligned} \tag{I.8}$$

Physical singular limits:

$$\begin{aligned} &x_3 \to 0: \quad 3 \text{ soft}\;,\; x_3 \to 1:\; 4 \text{ soft}\;, \\ &x_1 \to 1: \quad 3 \parallel 1\;,\; x_1 \to 0:\; 3 \parallel 2\;,\; x_2 \to 1:\; 3 \parallel 4\;. \end{aligned}$$

We observe that $\tilde{s}_{14}$ has a line singularity at $x_4 = 0$, $x_1 = x_2$. If s_{14} appears in the denominator of the matrix element squared we are integrating over phase space,

further transformations need to be applied before this integrand can be treated via sector decomposition. If this is the case we decouple the problem at $x_4 = 0$ from the line singularity $x_1 = x_2$ by the transformation [119]

$$x_4 = \frac{t^- (1 - z_4)}{t^- + z_4 (t^+ - t^-)} \quad \Rightarrow \tilde{s}_{14} = \frac{t^+ t^-}{t^- + z_4 (t^+ - t^-)} \,, \tag{I.9}$$

leading to

$$\begin{aligned}
& \int_0^1 dx_4 \left[x_4 (1 - x_4)\right]^{\frac{D-5}{2}} (\tilde{s}_{14})^{-1} \\
= & \int_0^1 dz_4 \left[z_4 (1 - z_4) t^+ t^-\right]^{\frac{D-5}{2}} \left[t^- + z_4 (t^+ - t^-)\right]^{4-D} \\
= & \int_0^1 dz_4 \left[z_4 (1 - z_4)\right]^{\frac{D-5}{2}} \left| x_1(1 - x_2) - x_2(1 - x_1) \right|^{D-5} \\
& \left[(\sqrt{x_1(1 - x_2)} - \sqrt{x_2(1 - x_1)})^2 + 4\, z_4 \sqrt{x_1(1 - x_1) x_2(1 - x_2)} \right]^{4-D} .
\end{aligned}$$

Now we split the x_2-integration range at $x_2 = x_1$ and remap to the unit cube:

$$\int_0^1 dx_2 f(x_1, x_2) = \underbrace{\int_0^{x_1} dx_2 f(x_1, x_2)}_{(1)} + \underbrace{\int_{x_1}^1 dx_2 f(x_1, x_2)}_{(2)}$$

where we substitute $x_2 = x_1\, z_2$ in (1) and $x_2 = x_1 + (1 - x_1)\, z_2$ in (2). Using the fact that the contribution from region (2) equals the first one if we transform $z_2 \to 1 - z_2$ and $x_1 \to 1 - x_1$ and combining with the original phase space given in eq. (I.7), we obtain

$$\begin{aligned}
& \int d\Phi_3 \frac{1}{s_{14}} = \frac{1}{(2\pi)^{2D-3}} d\Omega_{D-3} d\Omega_{D-4}\, s^{D-4} 2^{D-7} \beta^{2D-6} \qquad \text{(I.10)} \\
& \int_0^1 dx_1 dz_2 dx_3 dz_4 [z_4 (1 - z_4)]^{\frac{D-5}{2}} x_3^{D-3} (1 - x_3)^{D-4} \\
& [1 - \beta\, x_3 (1 - x_1 + x_1 z_2)]^{3-D} x_1^{D-4} z_2^{D-5} \left[(1 - x_1)(1 - z_2)(1 - x_1 + x_1 z_2)\right]^{\frac{D-4}{2}} \\
& \Big[(\sqrt{(1 - x_1)(1 - z_2)} - \sqrt{1 - x_1 + x_1 z_2})^2 \\
+ \quad & 4\, z_4 \sqrt{(1 - x_1)(1 - z_2)(1 - x_1 + x_1 z_2)} \Big]^{4-D} .
\end{aligned}$$

I.2 A Phase Space for $pp \to t\bar{t}+$ Double Real Radiation

The $D-$dimensional phase space for $p_1+p_2 \to p_3+p_4+p_5+p_6$, where $p_3^2 = p_4^2 = m_t^2$ and the other particles are massless, is given by

$$\begin{aligned} d\Phi_{2\to 4} &= \Big[\prod_{j=3}^{6} \frac{d^D p_j}{(2\pi)^{D-1}}\Big]\, \delta^+(p_3^2 - m_t^2)\delta^+(p_4^2 - m_t^2) \\ &\quad \delta^+(p_5^2)\delta^+(p_6^2)(2\pi)^D \delta^{(D)}\Big(p_1 + p_2 - \sum_{j=3}^{6} p_j\Big) \end{aligned} \tag{I.11}$$

There are many ways to parametrise this phase space, and different parametrisations are better suited to different calculations. The following parametrisation is useful, in that it gives s_{56} in a simple form and avoids some of the square roots that can occur in the denominator for other parametrisations (eg the 'energy parametrisation', where energies and angles of the outgoing particles form the integration variables). Now we insert unity in the form

$$1 = \int \frac{d^D p_{34}}{(2\pi)^{D-1}} (2\pi)^D \delta^{(D)}\Big(p_3 + p_4 - p_{34}\Big) \frac{ds_{34}}{2\pi}\, \delta(p_{34}^2 - s_{34}) \tag{I.12}$$

Further, we factorize the phase space corresponding to $t\bar{t}$ production from the decay of a particle with momentum p_{34}. Using

$$\begin{aligned} d\tilde{\Phi}_{t\bar{t}} &= \frac{d^D p_3}{(2\pi)^{D-1}} \frac{d^D p_4}{(2\pi)^{D-1}}\, \delta^+(p_3^2 - m_t^2)\delta^+(p_4^2 - m_t^2) \\ &\quad (2\pi)^D \delta^{(D)}\Big(p_3 + p_4 - p_{34}\Big) \end{aligned} \tag{I.13}$$

we can write the full phase space as

$$\begin{aligned} d\Phi_{2\to 4} &= d\tilde{\Phi}_{t\bar{t}}\, \frac{ds_{34}}{2\pi} \frac{d^D p_5}{(2\pi)^{D-1}} \frac{d^D p_6}{(2\pi)^{D-1}}\, \delta^+(p_5^2)\delta^+(p_6^2) \\ &\quad \frac{d^D p_{34}}{(2\pi)^{D-1}} \delta(p_{34}^2 - s_{34})\ (2\pi)^D \delta^{(D)}\Big(p_1 + p_2 - p_{34} - p_5 - p_6\Big) \\ &=: \frac{ds_{34}}{2\pi}\, d\tilde{\Phi}_{t\bar{t}}\, d\tilde{\Phi}(p_5, p_6, p_{34})\ . \end{aligned} \tag{I.14}$$

Note that $d\tilde{\Phi}_{t\bar{t}}$ could be evaluated in four dimensions because there are no singularities associated with the "subprocess" $p_{34} \to t\bar{t}$.

Now we seek a parametrisation in which s_{56} has a simple form. In order to achieve a convenient form of $d\tilde{\Phi}(p_5, p_6, p_{34})$, we view the phase space as the product of a phase space for $p_1 + p_2 \to p_{34} + p_{56}$ and one for the decay $p_{56} \to p_5 + p_6$. In addition, we introduce the variable $v = p_1 \cdot p_{34}/p_2 \cdot p_{34}$, which is invariant under boosts in z-direction if we choose

$$p_1 = \frac{\sqrt{s}}{2}(1, \vec{0}^{(D-2)}, 1) \text{ and } p_2 = \frac{\sqrt{s}}{2}(1, \vec{0}^{(D-2)}, -1) . \tag{I.15}$$

To be concrete, we introduce

$$\begin{aligned} \int d\tilde{\Phi}(p_5, p_6, p_{34}) &= \int dv\, d\tilde{\Phi}_v\, \delta(v - \frac{p_1 \cdot p_{34}}{p_2 \cdot p_{34}}) \times \\ &\quad \int ds_{15}\, ds_{25}\, \delta(s_{15} + 2p_1 \cdot p_5)\, \delta(s_{25} + 2p_2 \cdot p_5)\, \frac{ds_{56}}{2\pi}\, d\tilde{\Phi}_{56} \end{aligned} \tag{I.16}$$

$$\begin{aligned} d\tilde{\Phi}_v &= \frac{d^D p_{34}}{(2\pi)^{D-1}} \delta(p_{34}^2 - s_{34})\, \frac{d^D p_{56}}{(2\pi)^{D-1}} \delta(p_{56}^2 - s_{56}) \\ &\quad (2\pi)^D \delta^{(D)}\Big(p_1 + p_2 - p_{34} - p_{56}) \\ d\tilde{\Phi}_{56} &= \frac{d^D p_5}{(2\pi)^{D-1}} \frac{d^D p_6}{(2\pi)^{D-1}}\, \delta^+(p_5^2)\delta^+(p_6^2)(2\pi)^D \delta^{(D)}(p_5 + p_6 - p_{56}) \end{aligned} \tag{I.17}$$

$d\tilde{\Phi}_v$ corresponds to a $2 \to 2$ phase space $p_1 + p_2 \to p_{34} + p_{56}$. To evaluate $d\tilde{\Phi}_v$ in the CMS frame of $p_1 + p_2$, we choose

$$p_{34} = (E_{34}, \vec{0}^{(D-3)}, |\vec{p}_{34}| \sin\theta_1, |\vec{p}_{34}| \cos\theta_1)$$

and eliminate p_{56} by momentum conservation. Further, we include the constraint $\delta(v - p_1 \cdot p_{34}/p_2 \cdot p_{34})$ which allows us to fix $\cos\theta_1$ in terms of v. We define

$$d\Phi_v = d\tilde{\Phi}_v\, \delta(v - \frac{p_1 \cdot p_{34}}{p_2 \cdot p_{34}})$$

such that

$$\begin{aligned} d\Phi_v &= \frac{1}{(2\pi)^{D-2}}\, d^{D-1}\vec{p}_{34}\, \frac{dE_{34}}{2E_{34}} \delta(E_{34} - \sqrt{\vec{p}_{34}^2 + s_{34}}) \\ &\quad \delta(s + s_{34} - s_{56} - 2(1+v)p_2 \cdot p_{34}) \delta(v - \frac{E_{34} - |\vec{p}_{34}|\cos\theta_1}{E_{34} + |\vec{p}_{34}|\cos\theta_1}) \\ &= \frac{d\Omega_{D-3}}{(2\pi)^{D-2}}\, d|\vec{p}_{34}|\, |\vec{p}_{34}|^{D-3} d(\cos\theta_1)\, (1 - \cos^2\theta_1)^{\frac{D-4}{2}} (1+v)^{-2} \\ &\quad \delta(s + s_{34} - s_{56} - (1+v)\sqrt{s}(E_{34} + |\vec{p}_{34}|\cos\theta_1))\, \delta\left(\cos\theta_1 - \frac{(1-v)E_{34}}{(1+v)|\vec{p}_{34}|}\right) \\ E_{34} &= \sqrt{\vec{p}_{34}^2 + s_{34}} \end{aligned} \tag{I.18}$$

Using $\cos\theta_1^0 = \frac{(1-v)E_{34}}{(1+v)|\vec{p}_{34}|}$ leads to

$$\begin{aligned} d\Phi_v &= \frac{d\Omega_{D-3}}{(2\pi)^{D-2}}\, d|\vec{p}_{34}|\, |\vec{p}_{34}|^{D-3}(1+v)^{-2}(1-\cos^2\theta_1^0)^{\frac{D-4}{2}} \\ &\quad \delta(s + s_{34} - s_{56} - 2\sqrt{s}\sqrt{\vec{p}_{34}^{\,2} + s_{34}}) \\ (1-\cos^2\theta_1^0) &= \frac{4v\,|\vec{p}_{34}|^2 - s_{34}(1-v)^2}{|\vec{p}_{34}|^2(1+v)^2} \end{aligned} \tag{I.19}$$

Solving the δ-constraint for $|\vec{p}_{34}|$ and considering only the positive solution leads to

$$|\vec{p}_{34}|^0 = \frac{\sqrt{s^2 - 2s(s_{34}+s_{56}) + (s_{34}-s_{56})^2}}{2\sqrt{s}} = \frac{\sqrt{\lambda(s, s_{34}, s_{56})}}{2\sqrt{s}} \tag{I.20}$$

and

$$\begin{aligned} d\Phi_v &= \frac{d\Omega_{D-3}}{(2\pi)^{D-2}}\, \frac{(s + s_{34} - s_{56})}{4s(1+v)^2} \left(|\vec{p}_{34}|^2(1-\cos^2\theta_1^0)\right)^{\frac{D-4}{2}} \\ &\quad d|\vec{p}_{34}|\, \delta\left(|\vec{p}_{34}| - \frac{\sqrt{\lambda(s, s_{34}, s_{56})}}{2\sqrt{s}}\right) \end{aligned} \tag{I.21}$$

where

$$\lambda(x,y,z) = x^2 + y^2 + z^2 - 2xy - 2yz - 2zx$$

is the Källen function. Substituting

$$z = \frac{s_{34}}{s} \quad , \quad y_{56} = \frac{s_{56}}{s}$$

leads to

$$\begin{aligned} |\vec{p}_{34}^{\,0}|^2(1-\cos^2\theta_1^0) &= \frac{s}{(1+v)^2}\left(v\, y_{56}^2 - 2v y_{56}(1+z) + (v-z)(1-vz)\right) \\ &= \frac{s\, v}{(1+v)^2}(y_{56}^+ - y_{56})(y_{56}^- - y_{56}) \\ \text{where } y_{56}^\pm &= \frac{(1+z)\sqrt{v} \pm (1+v)\sqrt{z}}{\sqrt{v}} \end{aligned} \tag{I.22}$$

We further substitute

$$y_{56} = x_1\, y_{56}^- \quad , \quad v = \frac{z + x_2(1-z)}{1 - x_2(1-z)}\ ,\ x_2 = \frac{v-z}{(1+v)(1-z)} \tag{I.23}$$

to obtain

$$\begin{aligned} d\Phi_v &= \frac{d\Omega_{D-3}}{(2\pi)^{D-2}}\, s^{\frac{D-4}{2}}\, \frac{(1-x_2(1-z))^2}{4(1+z)} \left(1 - \frac{x_1\, y_{56}^-}{1+z}\right) \\ &\quad \left(x_2(1-x_2)(1-z)^2(1-x_1)(1-x_1\,\frac{y_{56}^-}{y_{56}^+})\right)^{\frac{D-4}{2}} \\ y_{56}^{\pm} &= (1+z)\left(1 \pm \frac{\sqrt{z}}{\sqrt{(1-x_2(1-z))(z+x_2(1-z))}}\right) \\ \frac{v}{(1+v)^2}\, y_{56}^+\, y_{56}^- &= x_2\,(1-x_2)\,(1-z)^2 \end{aligned} \tag{I.24}$$

Now we turn to $d\tilde{\Phi}_{56}$. Regarded in isolation from the other particles involved, $d\tilde{\Phi}_{56}$ just describes the decay $p_{56} \to p_5 + p_6$. The final state can be described by a polar and an azimuthal angle forming Ω_{D-2}, which could be integrated over if $\vec{p}_{56}$ could be freely rotated in all directions. Our case is different: in the CMS of $p_1 + p_2$, we have fixed $\vec{p}_{56} = -\vec{p}_{34}$ to be in the same plane as p_1 and p_2 and $\vec{p}_{34}$ to be at polar angle θ_1 relative to the z-axis defined by the direction of p_1. Therefore the $D-2$ angles cannot be integrated out. We choose to replace the $D-2$ angular variables by $D-4$ angular variables and the two invariants s_{15} and s_{25} using the δ-functions already introduced in eq. (I.16).

$$\begin{aligned} d\Phi_{56} &= d\tilde{\Phi}_{56}\, \delta(s_{15} + 2p_1\cdot p_5)\, \delta(s_{25} + 2p_2 \cdot p_5) \\ &= \frac{1}{(2\pi)^{D-2}} d^D p_5\, \delta^+(p_5^2)\delta^+((p_5 - p_{56})^2)\, \delta(s_{15} + 2p_1\cdot p_5)\, \delta(s_{25} + 2p_2\cdot p_5)\ . \end{aligned} \tag{I.25}$$

For p_5 we introduce a parametrisation which is boost invariant in z-direction:

$$p_5 = (E_5, p_\perp \vec{e_5}, p_z)\ ,$$

where $\vec{e_5}$ is a (D-2)-dimensional unit vector, giving rise to the integrations over $d\phi$ and $d\Omega_{D-4}$ in $d\Phi_{56}$:

$$\begin{aligned} d\Phi_{56} &= \frac{1}{4\,(2\pi)^{D-2}} \frac{dE_5}{E_5}\, \delta\left(E_5 - \sqrt{p_\perp^2 + p_z^2}\right) \\ &\quad dp_\perp^2\ \left(p_\perp^2\right)^{\frac{D-4}{2}}\ dp_z\, d(\cos\phi)\, (\sin^2\phi)^{\frac{D-5}{2}} d\Omega_{D-4} \\ &\quad \delta\left(s_{56} - 2E_5\sqrt{s_{56} + \vec{p}_{56}^2} - 2|\vec{p}_{56}|\,(p_\perp\, \sin\theta_1 \cos\phi + p_z\, \cos\theta_1)\right) \\ &\quad \delta(s_{15} + \sqrt{s}(E_5 - p_z))\, \delta(s_{25} + \sqrt{s}(E_5 + p_z)) \end{aligned} \tag{I.26}$$

The angles $d\Omega_{D-4}$ can be integrated out since the system formed by p_5 and p_6 can be rotated without physical effect in these azimuthal angles. We use $\delta(s_{15}+\sqrt{s}(E_5-p_z))$ to eliminate p_z and $\delta(s_{25}+\sqrt{s}(E_5+p_z))$ to eliminate $p_\perp^2$:

$$\begin{aligned} p_z^0 &= \frac{s_{15}-s_{25}}{2\sqrt{s}} \\ (p_\perp^0)^2 &= \frac{s_{15}\,s_{25}}{s} \end{aligned}$$

and arrive at

$$\begin{aligned} d\Phi_{56} &= \frac{1}{8\,s}\,\frac{1}{(2\pi)^{D-2}} d\Omega_{D-4}\, d(\cos\phi)\,(\sin^2\phi)^{\frac{D-5}{2}}\,\left(\frac{s_{15}s_{25}}{s}\right)^{\frac{D-5}{2}} \\ &\quad \frac{1}{|\vec{p}_{56}|\,\sin\theta_1}\,\delta(\cos\phi-\frac{a}{b}) \qquad\qquad (I.27) \\ &= \frac{1}{4\,s}\,s^{\frac{5-D}{2}}\,\frac{1}{(2\pi)^{D-2}} d\Omega_{D-4}\,(b^2-a^2)^{\frac{D-5}{2}}\,\left(4\,|\vec{p}_{56}|^2\,\sin^2\theta_1\right)^{-\frac{D-4}{2}} \end{aligned}$$

$$\begin{aligned} a &= s_{56}\sqrt{s}+(s_{15}+s_{25})\sqrt{s_{56}+|\vec{p}_{56}|^2}-(s_{15}-s_{25})\,|\vec{p}_{56}|\,\cos\theta_1 \\ b &= 2\,|\vec{p}_{56}|\,\sin\theta_1\,\sqrt{s_{15}s_{25}} \\ |\vec{p}_{56}|^2 &= \frac{1}{4s}\,\lambda(s,s_{34},s_{56}) = \frac{s}{4}\lambda(1,z,y_{56}) \\ |\vec{p}_{56}|^2\,\sin^2\theta_1 &= \frac{s}{(1+v)^2}\left(v\,\lambda(1,z,y_{56})-z\,(1-v)^2\right) = \alpha_3^2 \\ &= \frac{s\,v}{(1+v)^2}(y_{56}^+-y_{56})(y_{56}^--y_{56}) \\ &= s\,x_2(1-x_2)\,(1-x_1)(1-x_1\,\frac{y_{56}^-}{y_{56}^+})\,(1-z)^2 \\ |\vec{p}_{56}|^2\,\cos^2\theta_1 &= \frac{s\,(1-v)^2}{4\,(1+v)^2}\,(1+z-y_{56})^2 \qquad = \qquad \alpha_2^2 \\ E_{56}^2 &= s_{56}+|\vec{p}_{56}|^2 = \frac{s}{4}\,(1-z+y_{56})^2 = \alpha_1^2(s_{56}) \end{aligned}$$

where eqs. (I.19), (I.20) and $|\vec{p}_{56}| = |\vec{p}_{34}|$ have been used in the last two lines. Note that

$$\alpha_2^2+\alpha_3^2 = \alpha_1^2(s_{56})-s_{56}\;.$$

The limits on s_{15} and s_{25} are obtained from the requirement $a^2/b^2 \le 1$. The result is

$$\begin{aligned} 0 &\ge s_{15} \ge -\sqrt{s}\,(\alpha_1(s_{56})+\sqrt{\alpha_2^2}) = -s_{15}^- \\ y_{15}^- &= \frac{1}{\sqrt{s}}\,(\alpha_1(s_{56})+\sqrt{\alpha_2^2}) = \frac{1}{\sqrt{s}}\,(E_{56}+|\vec{p}_{56}|\,|\cos\theta_1|) \end{aligned}$$

and, substituting

$$s_{15} = x_3\, s\, (-y_{15}^-)\ , \tag{I.28}$$

$$\begin{aligned} s_{25}^{\pm} &= \frac{s}{s_{15}^-}\left(\sqrt{(1-x_3)s_{56}} \pm \sqrt{x_3\,\alpha_3^2}\right)^2 \qquad (I.29)\\ &= \frac{1}{y_{15}^-}\left(\sqrt{(1-x_3)s_{56}} \pm \sqrt{x_3\,|\vec{p}_{56}|^2\,\sin^2\theta_1}\right)^2 \\ &= \frac{1}{y_{15}^-}\left(\sqrt{x_1(1-x_3)y_{56}^-} \pm \sqrt{x_3\,x_2(1-x_2)\,(1-x_1)\,(1-x_1\,\frac{y_{56}^-}{y_{56}^+})(1-z)^2}\right)^2 . \end{aligned}$$

Therefore we also substitute

$$s_{25} = -s_{25}^- - x_4\,(s_{25}^+ - s_{25}^-)\ . \tag{I.30}$$

After these substitutions we obtain

$$\begin{aligned} b^2 - a^2 &= s\,x_4\,(1-x_4)\,(s_{25}^+ - s_{25}^-)^2\,(y_{15}^-)^2 \\ &= 16\,s^2\,y_{56}\,x_3\,(1-x_3)\,x_4\,(1-x_4)\,|\vec{p}_{56}|^2\,\sin^2\theta_1 \\ &= 4\,s^3\,y_{56}^-\,x_1\,(1-x_1)\,(1-x_1\,\frac{y_{56}^-}{y_{56}^+}) \\ &\times\ x_2\,(1-x_2)\,x_3\,(1-x_3)\,x_4\,(1-x_4)\,(1-z)^2 \qquad (I.31) \end{aligned}$$

$$\begin{aligned} d\Phi_{56} &= \frac{2^{D-5}}{8\,s}\,\frac{s^{\frac{D-6}{2}}}{(2\pi)^{D-2}}\,d\Omega_{D-4}\,\frac{1}{(1-z)\sqrt{(1-x_1)(1-x_1\,\frac{y_{56}^-}{y_{56}^+})\,x_2(1-x_2)}} \\ &\quad \left(y_{56}^-\,x_1\,x_3(1-x_3)x_4(1-x_4)\right)^{\frac{D-5}{2}} \qquad (I.32) \end{aligned}$$

$$\begin{aligned} y_{56}^- &= \frac{x_2\,(1-x_2)\,(1-z)^2\,(1+v)^2}{v\,y_{56}^+} \\ \frac{(1+v)^2}{v} &= \frac{(1+z)^2}{(1-x_2(1-z))(z+x_2(1-z))} \end{aligned}$$

Now we turn to $d\tilde{\Phi}_{t\bar{t}}$. Again, we would like to replace the $D-2$ angles describing

the $t\bar{t}$ final state by invariants. Therefore we introduce

$$\begin{aligned} d\tilde{\Phi}_{34} &= ds_{13}ds_{23}\, d\tilde{\Phi}_{t\bar{t}}\, \delta(s_{13}+2p_1\cdot p_3-m^2)\,\delta(s_{23}+2p_2\cdot p_3-m^2) \\ &= ds_{13}ds_{23}\, d\Phi_{34} \\ d\Phi_{34} &= \frac{1}{(2\pi)^{D-2}} d^D p_3\, \delta^+(p_3^2-m^2)\delta^+((p_3-p_{34})^2-m^2) \\ &\quad \delta(s_{13}+2p_1\cdot p_3-m^2)\,\delta(s_{23}+2p_2\cdot p_3-m^2)\ . \end{aligned} \tag{I.33}$$

Proceeding in a way analogous to $d\Phi_{56}$ we obtain

$$d\Phi_{34} = \frac{1}{4\,s}\, s^{\frac{5-D}{2}}\, \frac{1}{(2\pi)^{D-2}} d\Omega_{D-4}\, (b^2-a^2)^{\frac{D-5}{2}}\, \left(4\,|\vec{p}_{34}|^2\,\sin^2\theta_1\right)^{-\frac{D-4}{2}}$$

$$\begin{aligned} a/\sqrt{s} &= s_{34} - 2E_3\,E_{34} - 2p_{z,0}\,|\vec{p}_{34}|\,\cos\theta_1 \\ b^2/s &= 4\,|\vec{p}_{34}|^2\,\sin^2\theta_1\, p_{\perp,0}^2 \end{aligned}$$

Defining

$$\xi_1 = m^2 - s_{13}\ ,\ \xi_2 = m^2 - s_{23} \tag{I.34}$$

we have

$$\begin{aligned} E_3 &= \frac{1}{2\sqrt{s}}(\xi_1+\xi_2) \\ p_{z,0} &= \frac{1}{2\sqrt{s}}(\xi_2-\xi_1) \\ p_{\perp,0}^2 &= \frac{1}{s}\xi_1\xi_2 - m^2 \end{aligned} \tag{I.35}$$

leading to

$$\begin{aligned} a &= s_{34}\sqrt{s} + \xi_1\,(\alpha_2-\alpha_1) - \xi_2\,(\alpha_1+\alpha_2) \\ b &= 2\alpha_3\sqrt{\xi_1\xi_2 - m^2 s} \\ \alpha_1^2 &= E_{34}^2 = s_{34} + |\vec{p}_{34}|^2 = \frac{s}{4}\,(1-z+y_{56})^2 = \alpha_1^2(s_{34}) \\ \alpha_2 &= |\vec{p}_{34}|\,\cos\theta_1 \\ \alpha_3 &= |\vec{p}_{34}|\,\sin\theta_1 = \sqrt{s}(1-z)\sqrt{(1-x_1)(1-x_1\,\frac{y_{56}^-}{y_{56}^+})\,x_2(1-x_2)} \end{aligned} \tag{I.36}$$

Solving the constraint $b^2 - a^2 \geq 0$ leads to

$$\xi_1^{\pm} = \frac{\sqrt{s}}{2}(\alpha_1+\alpha_2)\left(1 \pm \sqrt{1-\frac{4m^2}{s_{34}}}\right) \tag{I.37}$$

$$\xi_2^{\pm} = \frac{1}{(\alpha_1+\alpha_2)^2}\left\{(\xi_1^+ - \xi_1^-)\left(\sqrt{s_{34}\,(1-x_5)} \pm \sqrt{\alpha_3^2\,x_5}\right)^2 + \xi_1^-\,(s_{34}+\alpha_3^2)\right\} \tag{I.38}$$

$$\begin{aligned} b^2 - a^2 &= (\alpha_1+\alpha_2)^2\,(\xi_2^+ - \xi_2^-)^2\,x_6\,(1-x_6) \\ &= 16\,s\,\alpha_3^2\,(s_{34}-4m^2)\,x_5(1-x_5)\;x_6\,(1-x_6)\,, \end{aligned} \tag{I.39}$$

where we have made the substitutions

$$\begin{aligned} \xi_1 &= \xi_1^- + (\xi_1^+ - \xi_1^-)\,x_5 \\ \xi_2 &= \xi_2^- + (\xi_2^+ - \xi_2^-)\,x_6 \end{aligned} \tag{I.40}$$

The phase space then becomes

$$\begin{aligned} d\Phi_{34} = \frac{2^{D-5}}{8\,s}\,\frac{s^{\frac{D-6}{2}}}{(2\pi)^{D-2}}d\Omega_{D-4}\,&\frac{1}{(1-z)\sqrt{(1-x_1)(1-x_1\,\frac{y_{56}^-}{y_{56}^+})\,x_2(1-x_2)}} \\ &\left((z-\frac{4m^2}{s})\,x_5(1-x_5)\,x_6(1-x_6)\right)^{\frac{D-5}{2}}\,. \end{aligned}$$

In summary, we have made the substitutions

$$\begin{aligned} s_{34} &= s\,z \\ s_{56} &= x_1\,s\,y_{56}^- \\ v &= \frac{z+x_2(1-z)}{1-x_2(1-z)} \quad , \quad v = \frac{p_1\cdot p_{34}}{p_2\cdot p_{34}} \quad , \quad \frac{dv}{dx_2} = \frac{1-z^2}{(1-x_2(1-z))^2} \\ s_{15} &= -x_3\,s\,y_{15}^- \\ s_{25} &= s_{25}^- + x_4\,(s_{25}^+ - s_{25}^-) \\ \xi_1 &= m^2 - s_{13} = \xi_1^- + x_5\,(\xi_1^+ - \xi_1^-) \\ \xi_2 &= m^2 - s_{23} = \xi_2^- + x_6\,(\xi_2^+ - \xi_2^-) \end{aligned} \tag{I.41}$$

Collecting the Jacobians contributing to $d\tilde{\Phi}(p_5, p_6, p_{34})$ we therefore obtain

$$\begin{aligned} d\tilde{\Phi}(p_5, p_6, p_{34}) &= dv\, ds_{15}\, ds_{25}\, \frac{ds_{56}}{2\pi}\, d\Phi_v\, d\Phi_{56} \\ &= \frac{2^{D-5}}{16\pi} \frac{s^{D-3}}{(2\pi)^{2D-4}} d\Omega_{D-3}\, d\Omega_{D-4}\, (1-z)^{D-3} \prod_{i=1}^{4} dx_i \\ &\quad (y_{56}^-)^{\frac{D-2}{2}} \left(1 - \frac{x_1\, y_{56}^-}{1+z}\right) \Big(x_4(1-x_4)\Big)^{\frac{D-5}{2}} \\ &\quad \left(x_1\,(1-x_1)\, x_2\,(1-x_2)\, x_3\,(1-x_3)\,(1 - x_1\, \frac{y_{56}^-}{y_{56}^+})\right)^{\frac{D-4}{2}} \end{aligned} \tag{I.42}$$

Combining now with $d\tilde{\Phi}_{t\bar{t}}$ we have

$$\begin{aligned} d\tilde{\Phi}_{34} &= ds_{13} ds_{23}\, d\Phi_{34} \\ &= dx_5\, dx_6\, (\xi_1^+ - \xi_1^-)\, (\xi_2^+ - \xi_2^-)\, d\Phi_{34} \end{aligned} \tag{I.43}$$

$$\begin{aligned} (\xi_1^+ - \xi_1^-)\, (\xi_2^+ - \xi_2^-) &= 4\, s^2 \frac{(1-z)}{\sqrt{z}}\, (z - \frac{4m^2}{s}) \\ &\times \sqrt{(1-x_1)(1 - x_1\, \frac{y_{56}^-}{y_{56}^+})\, x_2(1-x_2)}\, \sqrt{x_5(1-x_5)} \end{aligned}$$

Therefore

$$\begin{aligned} d\Phi_{2\to 4} &= \frac{ds_{34}}{2\pi}\, d\tilde{\Phi}_{34}\; d\tilde{\Phi}(p_5, p_6, p_{34}) \qquad \text{(I.44)} \\ &= \frac{s^{\frac{D}{2}-1}}{4\pi} \frac{4^{D-5}}{(2\pi)^{D-2}} d\Omega_{D-4}\, dz\, dx_5\, dx_6\, d\tilde{\Phi}(p_5, p_6, p_{34})\; \frac{1}{\sqrt{z}} \\ &\quad \left(z - \frac{4m^2}{s}\right)^{\frac{D-3}{2}} \Big(x_5(1-x_5)\Big)^{\frac{D-4}{2}} \Big(x_6(1-x_6)\Big)^{\frac{D-5}{2}} \\ &= 2^{2D-14}\, \frac{s^{\frac{3D}{2}-4}}{(2\pi)^{3D-4}} d\Omega_{D-3}\, d\Omega_{D-4}\, d\Omega_{D-4}\, dz \prod_{i=1}^{6} dx_i \\ &\quad \frac{(1-z)^{D-3}}{\sqrt{z}} \left(z - \frac{4m^2}{s}\right)^{\frac{D-3}{2}} (y_{56}^-)^{\frac{D-2}{2}} \left(1 - \frac{x_1\, y_{56}^-}{1+z}\right) \\ &\quad \left(x_1\,(1-x_1)\, x_2\,(1-x_2)\, x_3\,(1-x_3)\, x_5(1-x_5)(1 - x_1\, \frac{y_{56}^-}{y_{56}^+})\right)^{\frac{D-4}{2}} \\ &\quad \Big(x_4(1-x_4)\, x_6(1-x_6)\Big)^{\frac{D-5}{2}}\,, \qquad \text{(I.45)} \end{aligned}$$

where y_{56}^- is given by eq. (I.24). To write the invariants in a concise form, we intro-

duce

$$\begin{aligned}
\tilde{y}_{56}^{\pm} &= y_{56}^{\pm}/(1+z) = 1 \pm \frac{\sqrt{z}}{\sqrt{(1-x_2(1-z))(z+x_2(1-z))}} \\
\text{and} &\quad \text{we use} \\
\frac{\alpha_1(s_{56})+\alpha_2}{\sqrt{s}} &= x_1\,\tilde{y}_{56}^{-} + (1-x_2)\,(1-z)\,(1-x_1\,\tilde{y}_{56}^{-}) \\
\frac{\alpha_1(s_{56})-\alpha_2}{\sqrt{s}} &= x_1\,\tilde{y}_{56}^{-} + x_2\,(1-z)\,(1-x_1\,\tilde{y}_{56}^{-}) \\
\frac{\alpha_1(s_{34})+\alpha_2}{\sqrt{s}} &= \Big(1-x_2\,(1-z)\Big)\,(1-x_1\,\tilde{y}_{56}^{-}) \\
\frac{\alpha_1(s_{34})-\alpha_2}{\sqrt{s}} &= \Big(1-(1-x_2)\,(1-z)\Big)\,(1-x_1\,\tilde{y}_{56}^{-})
\end{aligned} \tag{I.46}$$

Note that $\alpha_2 \to -\alpha_2$ is related to $x_2 \to (1-x_2)$ and that $\alpha_1(s_{56}) + \alpha_1(s_{34}) = \sqrt{s}$. We obtain

$$\begin{aligned}
s_{34} &= s\,z \\
s_{56} &= s\,x_1\,(1+z)\,\tilde{y}_{56}^{-} \\
s_{15} &= -x_3\,s_{15}^{-} = -x_3\,\sqrt{s}\,(\alpha_1(s_{56})+\alpha_2) \\
s_{25} &= -s_{25}^{-} - x_4\,(s_{25}^{+} - s_{25}^{-}) \\
s_{16} &= -(1-x_3)\,s_{15}^{-} = -(1-x_3)\,\sqrt{s}\,(\alpha_1(s_{56})+\alpha_2) \\
s_{26} &= -s_{25} - \sqrt{s}\,(\alpha_1(s_{56})-\alpha_2) \\
s_{13} &= m^2 - \xi_1^{-} - x_5\,(\xi_1^{+} - \xi_1^{-}) = m^2 - \xi_1^{-} - x_5\,\sqrt{s}\,(\alpha_1(s_{34})+\alpha_2)\left(1-\frac{4m^2}{z\,s}\right) \\
s_{23} &= m^2 - \xi_2^{-} - x_6\,(\xi_2^{+} - \xi_2^{-}) \\
s_{14} &= 2m^2 - s_{13} - \sqrt{s}\,(\alpha_1(s_{34})-\alpha_2) \\
s_{24} &= 2m^2 - s_{23} - \sqrt{s}\,(\alpha_1(s_{34})+\alpha_2) \\
s_{134} &= s\,\left(-x_2\,(1-z) + x_1\,\tilde{y}_{56}^{-}(1-(1-x_2)(1-z))\right) \\
s_{234} &= s\,\left(-(1-x_2)\,(1-z) + x_1\,\tilde{y}_{56}^{-}(1-x_2(1-z))\right)
\end{aligned}$$

$$\begin{aligned}
s_{35} &= m^2 - \frac{1}{s}\left(s_{25}\,\xi_1 + s_{15}\,\xi_2 + \frac{a_3 a_5}{2\alpha_3^2}\right) \\
&\quad -8s\sqrt{y_{56}(z-\frac{4m^2}{s})x_3(1-x_3)x_4(1-x_4)x_5(1-x_5)x_6(1-x_6)} \\
a_3 &= s_{34}\sqrt{s} - \xi_1\,(\alpha_1(s_{34})-\alpha_2) - \xi_2\,(\alpha_1(s_{34})+\alpha_2) \\
a_5 &= s_{56}\sqrt{s} + s_{15}\,(\alpha_1(s_{56})-\alpha_2) + s_{25}\,(\alpha_1(s_{56})+\alpha_2)
\end{aligned} \tag{I.47}$$

Choosing $s_{34}, s_{56}, s_{15}, s_{25}, s_{13}, s_{23}, s_{16}, s_{35}$ as independent variables, we can verify the relations

$$\begin{aligned}
s_{36} &= 4m^2 - s_{13} - s_{23} - s_{34} - s_{35} \\
s_{45} &= 2m^2 - s_{15} - s_{25} - s_{35} - s_{56} \\
s_{46} &= -2m^2 + s + s_{13} + s_{15} + s_{23} + s_{25} + s_{35} \\
s_{26} &= -s - s_{15} - s_{16} - s_{25} + s_{34} - s_{56} \\
s_{14} &= 2m^2 - s - s_{13} - s_{15} - s_{16} \\
s_{24} &= 2m^2 + s_{15} + s_{16} - s_{23} - s_{34} + s_{56}
\end{aligned} \tag{I.48}$$

Note that $4m^2 \leq s_{34} \leq s$, therefore we substitute

$$z = 1 - \beta\, x_7\ ,\ \beta = (1 - \frac{4m^2}{s})\ , \tag{I.49}$$

such that $\tilde{y}_{56}^{\pm}$ becomes

$$\tilde{y}_{56}^{\pm} = \frac{y_{56}^{\pm}}{2 - x_7\,\beta} = 1 \pm \frac{\sqrt{1 - x_7\,\beta}}{\sqrt{(1 - x_2\,x_7\,\beta)(1 - (1 - x_2)\,x_7\,\beta)}}$$

We finally obtain for the phase space

$$\begin{aligned}
d\Phi_{2\to 4} &= 2^{2D-14}\,\beta^{\frac{3D-7}{2}}\,\frac{s^{\frac{3D}{2}-4}}{(2\pi)^{3D-4}} d\Omega_{D-3}\,d\Omega_{D-4}\,d\Omega_{D-4}\prod_{i=1}^{7} dx_i \\
&\quad \frac{x_7^{D-3}}{\sqrt{1 - x_7\,\beta}}\,(1 - x_7)^{\frac{D-3}{2}}\,(y_{56}^{-})^{\frac{D-2}{2}}\,\left(1 - x_1\,\tilde{y}_{56}^{-}\right) \\
&\quad \left(x_1\,(1 - x_1)\,x_2\,(1 - x_2)\,x_3\,(1 - x_3)\,x_5(1 - x_5)(1 - x_1\,\frac{y_{56}^{-}}{y_{56}^{+}})\right)^{\frac{D-4}{2}} \\
&\quad \Big(x_4(1 - x_4)\,x_6(1 - x_6)\Big)^{\frac{D-5}{2}}\ .
\end{aligned} \tag{I.50}$$

The variables x_5 and x_6 can be excluded from the decomposition as they never lead to a singularity. $E_5 \to 0$ corresponds to $x_1 \to 0, x_3 \to 0$, $E_6 \to 0$ corresponds to $x_1 \to 0, x_3 \to 1$. $x_7 = 0$ means $E_5 = E_6 = 0$.

Appendix J

Troubleshooting

Below we give possible reasons and solutions for problems which may arise during use of the program.

- The function $\mathcal{F}$ is zero:

  verify the on-shell conditions `onshell={...}` in the file `mytemplate.m` where you defined the integrand. By default, the external legs have been set to be light-like ($p_i^2 = ssp[i] = 0$). If you calculate a massless two-point integral or a one-scale three-point integral, at least one scale must be different from zero (e.g. set $ssp[1] = -1$ for a two-point function with external momentum p_1, which amounts to factoring out the overall scale).

  Remember that the program by default replaces p_i^2 by $ssp[i]$, $(p_i + p_j)^2$ by $sp[i, j]$. (This is done in src/deco/calcFU.m). If symbols different from p_i are used for the external momenta, the user has to define their numerical values in his template file `mytemplate.m` in the list `onshell`.

  Example: for external vectors called p, q, define numerical values for the invariants formed by p and q, e.g. $onshell=\{p^2 \to -1, q^2 \to 0, p*q \to -0.5\}$. Alternatively, you can map to the predefined names for the invariants, e.g. $onshell=\{p^2 \to ssp[1], q^2 \to ssp[2], p*q \to (sp[1,2] - ssp[1] - ssp[2])/2\}$. This latter solution allows you to leave the invariants symbolic and specify numerical values only at the numerical integration stage, by assigning the corresponding numerical values in `param.input`.

The user can check if the functions $\mathcal{F}, \mathcal{U}$ and numerator look as expected by looking at the file `FUN.m` in the `integralname/` subdirectory.

- The numerical integration takes very long:
  apart from the fact that this is to be expected for complicated integrands, other reasons could be

  – the integrand still contains undefined symbols at the numerical integration stage because the numerical values for the constants have not been properly defined (e.g. values for which $\mathcal{F}$ is not of definite sign, respectively the general function develops a singularity within the integration range). Things to do are: check the function $\mathcal{F}$ in the file `FUN.m` in the `integralname/` subdirectory (in the loop case); check the log files of the numerical integration in the subdirectories
  `integralname/polestructure/epstothe[i]`, where "polestructure" is of the form e.g. `2l0h0`, denoting 2 logarithmic poles and 0 linear, 0 higher poles.

  – you chose a very large number of Monte Carlo points and/or a very large number of iterations in the input file

  – for functions defined in `SecDec/general`: verify if there is a singularity for $x_i \to 1$ rather than only for $x_i \to 0$ and if so, split this variable at 1/2 by adding its label to the `splitlist`.

- the results do not appear in an editor window:
  either you did not specify an editor in `param.input` (last entry) or your system is unable to open the editor window. In this case just look at the result file located in the `integralname` subdirectory (where `integralname` is the name for the calculated integral or graph, specified by you in `param.input`, third item). The result file is called `integralname_[point]full.res`.

- you get the message "path for Mathematica not automatically found":
  Insert the path to Mathematica on your system manually for the variable `$mathpath` in the file `perlsrc/mathlaunch.pl`.

Appendix K

Choosing a Set of Variables to Decompose

K.1 Choosing a Minimal Subset for Iterated Decomposition

There are a number of criteria which are used to choose which set of variables, $\mathcal{S}$, is used at each stage in the decomposition. These are:

1. Number of variables in $\mathcal{S}$. Clearly as a new sector is produced for each variable in $\mathcal{S}$, sets with fewer variables are favoured.

2. How many functions $\mathcal{S}$ nullifies when $\mathcal{S} \to \{0, 0, ..., 0\}$. The more functions nullified, the quicker each function becomes factorised and so fewer iterations are expected.

3. What is the maximum order of $\mathcal{S}$ in the function being decomposed? This is defined by setting each variable in $\mathcal{S} \to X$, and finding the highest power of X. The smaller this value, the fewer additional powers of integration variables are introduce and this generally leads to fewer sectors.

K.2 Decomposing to Avoid Infinite Recursion

In order to avoid infinite recursion in sectors where it is suspected to occur, an heuristic method is employed in an attempt to sidestep this. Consider the example

$$f(\vec{x}) = x_1^2 + x_2^2 x_3 \tag{K.1}$$

A natural choice of subset to decompose here would be $\mathcal{S}_1 = \{x_1, x_3\}$, which leads to

$$f_1(\vec{x}) = x_1 + x_2^2 x_3, \tag{K.2}$$

$$f_2(\vec{x}) = x_1^2 x_3 + x_2^2 \tag{K.3}$$

It is easy to see that following the same prescription for f_2 will lead to

$$f_{2,2}(\vec{x}) = x_1^2 + x_2^2 x_3 = f(\vec{x}) \tag{K.4}$$

and so we fall into an infinite loop. A different choice of subset $\mathcal{S}_2 = \{x_1, x_2\}$ leads to

$$f_1(\vec{x}) = 1 + x_2^2 x_3 \tag{K.5}$$

$$f_2(\vec{x}) = x_1^2 + x_3 \tag{K.6}$$

and from here it is clear that the iteration will terminate. To this end the method used to sidestep infinite recursion is to decompose first all squared variables, and then follow the standard method outlined above. This has worked in all two-loop and three-loop applications considered, but we do not guarantee that it works in every up to three-loop diagram. An example where it fails is a four-loop massive propagator correction of figure K.1 at threshold ($p^2 = m^2$):

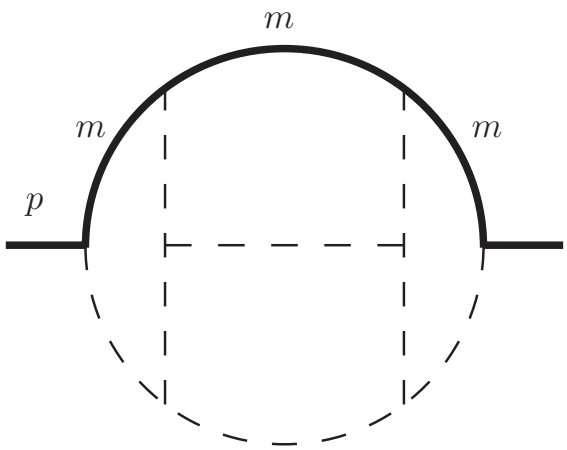


Figure K.1: A four-loop massive two-point integral

Bibliography

[1] Michael E. Peskin and Daniel V. Schroeder. An Introduction to quantum field theory. 1995.

[2] R.Keith Ellis, W.James Stirling, and B.R. Webber. QCD and collider physics. *Camb.Monogr.Part.Phys.Nucl.Phys.Cosmol.*, 8:1–435, 1996.

[3] Raymond Brock et al. Handbook of perturbative QCD: Version 1.0. *Rev.Mod.Phys.*, 67:157–248, 1995.

[4] Davison E. Soper. Basics of QCD perturbation theory. pages 267–316, 2000.

[5] T. Kinoshita. Mass singularities of Feynman amplitudes. *J. Math. Phys.*, 3:650–677, 1962.

[6] T. D. Lee and M. Nauenberg. Degenerate systems and mass singularities. *Phys. Rev.*, 133:B1549–B1562, 1964.

[7] William B. Kilgore. Regularization Schemes and Higher Order Corrections. *Phys.Rev.*, D83:114005, 2011.

[8] T. van Ritbergen, J.A.M. Vermaseren, and S.A. Larin. The Four loop beta function in quantum chromodynamics. *Phys.Lett.*, B400:379–384, 1997.

[9] M. Czakon. The Four-loop QCD beta-function and anomalous dimensions. *Nucl.Phys.*, B710:485–498, 2005.

[10] K. et al Chetyrkin. Talk given at radcor 2011, to be published in the proceedings.

[11] M. Gockeler, R. Horsley, A.C. Irving, D. Pleiter, P.E.L. Rakow, et al. A Determination of the Lambda parameter from full lattice QCD. *Phys.Rev.*, D73:014513, 2006.

[12] John C. Collins, Davison E. Soper, and George F. Sterman. Factorization of Hard Processes in QCD. *Adv.Ser.Direct.High Energy Phys.*, 5:1–91, 1988. To be publ. in 'Perturbative QCD' (A.H. Mueller, ed.) (World Scientific Publ., 1989).

[13] R.P. Feynman. Photon-hadron interactions. 1973.

[14] A.D. Martin, W.J. Stirling, R.S. Thorne, and G. Watt. Heavy-quark mass dependence in global PDF analyses and 3- and 4-flavour parton distributions. *Eur.Phys.J.*, C70:51–72, 2010.

[15] Marco Guzzi, Pavel M. Nadolsky, Hung-Liang Lai, and C.-P. Yuan. Heavy-flavor contributions at NNLO in CTEQ PDF analysis. 2011.

[16] J. Rojo, S. Forte, R.D. Ball, L. Del Debbio, M. Ubiali, et al. The impact of heavy quark mass effects in the NNPDF global analysis. *PoS*, DIS2010:173, 2010.

[17] A.G. Grozin. Integration by parts: An Introduction. *Int.J.Mod.Phys.*, A26:2807–2854, 2011.

[18] S. Laporta. High precision calculation of multiloop Feynman integrals by difference equations. *Int.J.Mod.Phys.*, A15:5087–5159, 2000.

[19] Charalampos Anastasiou and Achilleas Lazopoulos. Automatic integral reduction for higher order perturbative calculations. *JHEP*, 0407:046, 2004.

[20] A.V. Smirnov. Algorithm FIRE – Feynman Integral REduction. *JHEP*, 0810:107, 2008.

[21] C. Studerus. Reduze-Feynman Integral Reduction in C++. *Comput.Phys.Commun.*, 181:1293–1300, 2010.

[22] J. B. Tausk. Non-planar massless two-loop Feynman diagrams with four on-shell legs. *Phys. Lett.*, B469:225–234, 1999.

[23] Vladimir A. Smirnov. Analytical result for dimensionally regularized massless on-shell double box. *Phys. Lett.*, B460:397–404, 1999.

[24] G. Heinrich, T. Huber, D. A. Kosower, and V. A. Smirnov. Nine-Propagator Master Integrals for Massless Three-Loop Form Factors. *Phys. Lett.*, B678:359–366, 2009.

[25] C. Duhr, H. Gangl, and J.R. Rhodes. From polygons and symbols to polylogarithmic functions. 2011.

[26] Vittorio Del Duca, Lance J. Dixon, James M. Drummond, Claude Duhr, Johannes M. Henn, et al. The one-loop six-dimensional hexagon integral with three massive corners. *Phys.Rev.*, D84:045017, 2011.

[27] A. von Manteuffel and C. Studerus. Talk given at acat 2011, to be published in the proceedings.

[28] J. Fleischer, A.V. Kotikov, and O.L. Veretin. Analytic two loop results for selfenergy type and vertex type diagrams with one nonzero mass. *Nucl.Phys.*, B547:343–374, 1999.

[29] J. A. M. Vermaseren. New features of FORM. 2000.

[30] S. Moch and P. Uwer. XSummer: Transcendental functions and symbolic summation in form. *Comput.Phys.Commun.*, 174:759–770, 2006.

[31] V. de Alfaro, B. Jakšić, and T. Regge. Differential properties of Feynman Amplitudes. *High-Energy Physics and Elementary Particles*, 1965.

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

[33] Ettore Remiddi. Differential equations for Feynman graph amplitudes. *Nuovo Cim.*, A110:1435–1452, 1997.

[34] T. Gehrmann and E. Remiddi. Differential equations for two loop four point functions. *Nucl.Phys.*, B580:485–518, 2000.

[35] T. Gehrmann and E. Remiddi. Using differential equations to compute two loop box integrals. *Nucl.Phys.Proc.Suppl.*, 89:251–255, 2000.

[36] T. Gehrmann and E. Remiddi. Two-loop master integrals for $\gamma^* \to 3$ jets: The planar topologies. *Nucl. Phys.*, B601:248–286, 2001.

[37] T. Gehrmann and E. Remiddi. Two-loop master integrals for $\gamma^* \to 3$ jets: The non- planar topologies. *Nucl. Phys.*, B601:287–317, 2001.

[38] T. Gehrmann and E. Remiddi. Progress on two loop nonpropagator integrals. 2001.

[39] R.N. Lee. Space-time dimensionality D as complex variable: Calculating loop integrals using dimensional recurrence relation and analytical properties with respect to D. *Nucl.Phys.*, B830:474–492, 2010.

[40] O. V. Tarasov. Connection between Feynman integrals having different values of the space-time dimension. *Phys. Rev.*, D54:6479–6490, 1996.

[41] Roman N. Lee, Alexander V. Smirnov, and Vladimir A. Smirnov. Dimensional recurrence relations: an easy way to evaluate higher orders of expansion in ϵ. 2010.

[42] R.N. Lee, A.V. Smirnov, and V.A. Smirnov. Master Integrals for Four-Loop Massless Propagators up to Transcendentality Weight Twelve. 2011.

[43] D.H. Bailey and H.R.P Ferguson. A Polynomial Time, Numerically Stable Integer Relation Algorithm. *RNR Techn. Rept*, RNR-91-032, 1992.

[44] S. Arno, D.H. Bailey, and H.R.P Ferguson. Analysis of PSLQ, an integer relation finding algorithm. *Math. Comput*, 68:351–369, 1999.

[45] Henrik Kragh Sørensen. Exploratory experimentation in experimental mathematics: A glimpse at the PSLQ algorithm. *PhiMSAMP*, London: College Publications:341–360, 2010.

[46] Francis Brown. The Massless higher-loop two-point function. *Commun.Math.Phys.*, 287:925–958, 2009.

[47] A B Goncharov. Multiple polylogarithms and mixed tate motives. *Arxiv preprint math0103059*, page 82, 2001.

[48] R.Keith Ellis, D.A. Ross, and A.E. Terrano. The Perturbative Calculation of Jet Structure in e+ e- Annihilation. *Nucl.Phys.*, B178:421, 1981.

[49] Zoltan Kunszt and Davison E. Soper. Calculation of jet cross-sections in hadron collisions at order alpha-s**3. *Phys.Rev.*, D46:192–221, 1992.

[50] S. Catani and M.H. Seymour. The Dipole formalism for the calculation of QCD jet cross-sections at next-to-leading order. *Phys.Lett.*, B378:287–301, 1996.

[51] S. Catani and M.H. Seymour. A General algorithm for calculating jet cross-sections in NLO QCD. *Nucl.Phys.*, B485:291–419, 1997.

[52] Rikkert Frederix, Thomas Gehrmann, and Nicolas Greiner. Automation of the Dipole Subtraction Method in MadGraph/MadEvent. *JHEP*, 0809:122, 2008.

[53] S. Frixione, Z. Kunszt, and A. Signer. Three jet cross-sections to next-to-leading order. *Nucl. Phys.*, B467:399–442, 1996.

[54] Rikkert Frederix, Stefano Frixione, Fabio Maltoni, and Tim Stelzer. Automation of next-to-leading order computations in QCD: The FKS subtraction. *JHEP*, 0910:003, 2009.

[55] David A. Kosower. Antenna factorization of gauge theory amplitudes. *Phys.Rev.*, D57:5410–5416, 1998.

[56] John M. Campbell, M.A. Cullen, and E.W.Nigel Glover. Four jet event shapes in electron - positron annihilation. *Eur.Phys.J.*, C9:245–265, 1999.

[57] A. Gehrmann-De Ridder, T. Gehrmann, and E. W. Nigel Glover. Antenna subtraction at NNLO. *JHEP*, 09:056, 2005.

[58] A. Gehrmann-De Ridder, T. Gehrmann, and E.W.Nigel Glover. Antenna subtraction method for jet calculations at NNLO. *Nucl.Phys.Proc.Suppl.*, 157:32–36, 2006.

[59] Stefan Weinzierl. The Infrared structure of e+ e- $\to$ 3 jets at NNLO reloaded. *JHEP*, 0907:009, 2009.

[60] A. Daleo, T. Gehrmann, and D. Maitre. Antenna subtraction with hadronic initial states. *JHEP*, 0704:016, 2007.

[61] Alejandro Daleo, Aude Gehrmann-De Ridder, Thomas Gehrmann, and Gionata Luisoni. Antenna subtraction at NNLO with hadronic initial states: initial-final configurations. *JHEP*, 1001:118, 2010.

[62] Radja Boughezal, Aude Gehrmann-De Ridder, and Mathias Ritzmann. NNLO antenna subtraction with two hadronic initial states. *PoS*, RADCOR2009:052, 2010.

[63] Alejandro Daleo, Aude Gehrmann-De Ridder, Thomas Gehrmann, and Gionata Luisoni. NNLO Antenna Subtraction with One Hadronic Initial State. *PoS*, RADCOR2009:062, 2010.

[64] Radja Boughezal, Aude Gehrmann-De Ridder, and Mathias Ritzmann. Antenna subtraction at NNLO with hadronic initial states: double real radiation for initial-initial configurations with two quark flavours. *JHEP*, 1102:098, 2011.

[65] Gionata Luisoni, Alejandro Daleo, Aude Gehrmann-De Ridder, and Thomas Gehrmann. NNLO antenna subtraction with one hadronic initial state. *PoS*, DIS2010:122, 2010.

[66] Radja Boughezal, Aude Gehrmann-De Ridder, and Mathias Ritzmann. Antenna subtraction for two hadronic initial states at NNLO. *PoS*, DIS2010:101, 2010.

[67] Thomas Gehrmann and Pier Francesco Monni. Antenna subtraction at NNLO with hadronic initial states: real-virtual initial-initial configurations. 2011.

[68] A. Gehrmann-De Ridder and M. Ritzmann. NLO Antenna Subtraction with Massive Fermions. *JHEP*, 0907:041, 2009.

[69] G. Abelof and A. Gehrmann-De Ridder. Antenna subtraction for the production of heavy particles at hadron colliders. *JHEP*, 1104:063, 2011.

[70] Aude Gehrmann-De Ridder, Mathias Ritzmann, and Peter Skands. Timelike Dipole-Antenna Showers with Massive Fermions. 2011.

[71] K. Fabricius, I. Schmitt, G. Kramer, and G. Schierholz. Higher Order Perturbative QCD Calculation of Jet Cross-Sections in e+ e- Annihilation. *Z.Phys.*, C11:315, 1981.

[72] W.T. Giele and E.W.Nigel Glover. Higher order corrections to jet cross-sections in e+ e- annihilation. *Phys.Rev.*, D46:1980–2010, 1992.

[73] A. Gehrmann-De Ridder and E.W.Nigel Glover. A Complete O (alpha alpha-s) calculation of the photon + 1 jet rate in e+ e- annihilation. *Nucl.Phys.*, B517:269–323, 1998.

[74] B.W. Harris and J.F. Owens. The Two cutoff phase space slicing method. *Phys.Rev.*, D65:094032, 2002.

[75] M. Czakon. A novel subtraction scheme for double-real radiation at NNLO. *Phys. Lett.*, B693:259–268, 2010.

[76] M. Czakon. Double-real radiation in hadronic top quark pair production as a proof of a certain concept. *Nucl.Phys.*, B849:250–295, 2011.

[77] Charalampos Anastasiou, Gunther Dissertori, and Fabian Stockli. NNLO QCD predictions for the $H \to WW \to ll\nu\nu$ signal at the LHC. *JHEP*, 09:018, 2007.

[78] Charalampos Anastasiou, Franz Herzog, and Achilleas Lazopoulos. The fully differential decay rate of a Hoggs boson to bottom-quarks at NNLO in QCD. 2011.

[79] Kirill Melnikov and Frank Petriello. Electroweak gauge boson production at hadron colliders through $O(\alpha_s^2)$. *Phys. Rev.*, D74:114017, 2006.

[80] Stefano Catani and Massimiliano Grazzini. An NNLO subtraction formalism in hadron collisions and its application to Higgs boson production at the LHC. *Phys.Rev.Lett.*, 98:222002, 2007.

[81] Stefano Catani, Leandro Cieri, Giancarlo Ferrera, Daniel de Florian, and Massimiliano Grazzini. Vector boson production at hadron colliders: A Fully exclusive QCD calculation at NNLO. *Phys.Rev.Lett.*, 103:082001, 2009.

[82] Stefano Catani, Leandro Cieri, Daniel de Florian, Giancarlo Ferrera, and Massimiliano Grazzini. Diphoton production at hadron colliders: a fully-differential QCD calculation at NNLO. 2011.

[83] Klaus Hepp. Proof of the Bogolyubov-Parasiuk theorem on renormalization. *Commun.Math.Phys.*, 2:301–326, 1966.

[84] E. R. Speer. Mass Singularities of Generic Feynman Amplitudes. *Annales Poincare Phys. Theor.*, 26:87–105, 1977.

[85] M. Roth and Ansgar Denner. High-energy approximation of one-loop Feynman integrals. *Nucl. Phys.*, B479:495–514, 1996.

[86] Ansgar Denner and S. Pozzorini. An algorithm for the high-energy expansion of multi-loop diagrams to next-to-leading logarithmic accuracy. *Nucl. Phys.*, B717:48–85, 2005.

[87] T. Binoth and G. Heinrich. An automatized algorithm to compute infrared divergent multi-loop integrals. *Nucl. Phys.*, B585:741–759, 2000.

[88] T. Binoth and G. Heinrich. Numerical evaluation of multi-loop integrals by sector decomposition. *Nucl. Phys.*, B680:375–388, 2004.

[89] G. Heinrich and Vladimir A. Smirnov. Analytical evaluation of dimensionally regularized massive on-shell double boxes. *Phys. Lett.*, B598:55–66, 2004.

[90] T. Gehrmann, G. Heinrich, T. Huber, and C. Studerus. Master integrals for massless three-loop form factors: One- loop and two-loop insertions. *Phys. Lett.*, B640:252–259, 2006.

[91] G. Heinrich, T. Huber, and D. Maitre. Master integrals for fermionic contributions to massless three-loop form factors. 2007.

[92] P. A. Baikov, K. G. Chetyrkin, A. V. Smirnov, V. A. Smirnov, and M. Steinhauser. Quark and gluon form factors to three loops. *Phys. Rev. Lett.*, 102:212002, 2009.

[93] M. Czakon, J. Gluza, and T. Riemann. The planar four-point master integrals for massive two- loop Bhabha scattering. *Nucl. Phys.*, B751:1–17, 2006.

[94] R. Boughezal and M. Czakon. Single scale tadpoles and O($G_F m(t)^2 \alpha_s^3$) corrections to the rho parameter. *Nucl. Phys.*, B755:221–238, 2006.

[95] H. M. Asatrian et al. NNLL QCD contribution of the electromagnetic dipole operator to Gamma(anti-B –¿ X/s gamma). *Nucl. Phys.*, B749:325–337, 2006.

[96] A. V. Smirnov and M. Tentyukov. Four Loop Massless Propagators: a Numerical Evaluation of All Master Integrals. *Nucl. Phys.*, B837:40–49, 2010.

[97] P. A. Baikov and K. G. Chetyrkin. Four Loop Massless Propagators: an Algebraic Evaluation of All Master Integrals. *Nucl. Phys.*, B837:186–220, 2010.

[98] R. N. Lee, A. V. Smirnov, and V. A. Smirnov. Analytic Results for Massless Three-Loop Form Factors. *JHEP*, 04:020, 2010.

[99] R. N. Lee and V. A. Smirnov. Analytic Epsilon Expansions of Master Integrals Corresponding to Massless Three-Loop Form Factors and Three-Loop g-2 up to Four-Loop Transcendentality Weight. 2010.

[100] T. Gehrmann, E. W. N. Glover, T. Huber, N. Ikizlerli, and C. Studerus. Calculation of the quark and gluon form factors to three loops in QCD. *JHEP*, 06:094, 2010.

[101] T. Gehrmann, E. W. N. Glover, T. Huber, N. Ikizlerli, and C. Studerus. The quark and gluon form factors to three loops in QCD through to $O(\epsilon^2)$. 2010.

[102] A. V. Smirnov and M. N. Tentyukov. Feynman Integral Evaluation by a Sector decomposiTion Approach (FIESTA). *Comput. Phys. Commun.*, 180:735–746, 2009.

[103] A. V. Smirnov, V. A. Smirnov, and M. Tentyukov. FIESTA 2: parallelizeable multiloop numerical calculations. 2009.

[104] Jonathon Carter and Gudrun Heinrich. SecDec: A general program for sector decomposition. *Comput.Phys.Commun.*, 182:1566–1581, 2011.

[105] Gudrun Heinrich. Sector Decomposition. *Int. J. Mod. Phys.*, A23:1457–1486, 2008.

[106] Andrea Ferroglia, Massimo Passera, Giampiero Passarino, and Sandro Uccirati. All-purpose numerical evaluation of one-loop multi-leg Feynman diagrams. *Nucl. Phys.*, B650:162–228, 2003.

[107] T. Binoth, G. Heinrich, and N. Kauer. A numerical evaluation of the scalar hexagon integral in the physical region. *Nucl. Phys.*, B654:277–300, 2003.

[108] Achilleas Lazopoulos, Thomas McElmurry, Kirill Melnikov, and Frank Petriello. Next-to-leading order QCD corrections to $t\bar{t}Z$ production at the LHC. *Phys. Lett.*, B666:62–65, 2008.

[109] Achilleas Lazopoulos, Kirill Melnikov, and Frank Petriello. QCD corrections to tri-boson production. *Phys. Rev.*, D76:014001, 2007.

[110] Charalampos Anastasiou, Stefan Beerli, and Alejandro Daleo. Evaluating multi-loop Feynman diagrams with infrared and threshold singularities numerically. *JHEP*, 05:071, 2007.

[111] Charalampos Anastasiou, Stefan Beerli, and Alejandro Daleo. The two-loop QCD amplitude $gg \to h, H$ in the Minimal Supersymmetric Standard Model. 2008.

[112] Davison E. Soper. QCD calculations by numerical integration. *Phys. Rev. Lett.*, 81:2638–2641, 1998.

[113] Davison E. Soper. Techniques for QCD calculations by numerical integration. *Phys. Rev.*, D62:014009, 2000.

[114] T. Binoth, J. Ph. Guillet, G. Heinrich, E. Pilon, and C. Schubert. An algebraic / numerical formalism for one-loop multi-leg amplitudes. *JHEP*, 10:015, 2005.

[115] Zoltan Nagy and Davison E. Soper. Numerical integration of one-loop Feynman diagrams for N-photon amplitudes. *Phys. Rev.*, D74:093006, 2006.

[116] Wei Gong, Zoltan Nagy, and Davison E. Soper. Direct numerical integration of one-loop Feynman diagrams for N-photon amplitudes. *Phys. Rev.*, D79:033005, 2009.

[117] Gudrun Heinrich. A numerical method for NNLO calculations. *Nucl. Phys. Proc. Suppl.*, 116:368–372, 2003.

[118] A. Gehrmann-De Ridder, T. Gehrmann, and G. Heinrich. Four-particle phase space integrals in massless QCD. *Nucl. Phys.*, B682:265–288, 2004.

[119] Charalampos Anastasiou, Kirill Melnikov, and Frank Petriello. A new method for real radiation at NNLO. *Phys. Rev.*, D69:076010, 2004.

[120] T. Binoth and G. Heinrich. Numerical evaluation of phase space integrals by sector decomposition. *Nucl. Phys.*, B693:134–148, 2004.

[121] Charalampos Anastasiou, Kirill Melnikov, and Frank Petriello. Real radiation at NNLO: $e^+e^- \to 2$ jets through $\mathrm{O}(\alpha_s^2)$. *Phys. Rev. Lett.*, 93:032002, 2004.

[122] Charalampos Anastasiou, Kirill Melnikov, and Frank Petriello. Higgs boson production at hadron colliders: Differential cross sections through next-to-next-to-leading order. *Phys. Rev. Lett.*, 93:262002, 2004.

[123] Charalampos Anastasiou, Kirill Melnikov, and Frank Petriello. Fully differential higgs boson production and the di-photon signal through next-to-next-to-leading order. *Nucl. Phys.*, B724:197–246, 2005.

[124] Charalampos Anastasiou, Kirill Melnikov, and Frank Petriello. The electron energy spectrum in muon decay through O(α^2). *JHEP*, 09:014, 2007.

[125] Kirill Melnikov and Frank Petriello. The W boson production cross section at the LHC through O(α_s^2). *Phys. Rev. Lett.*, 96:231803, 2006.

[126] Kirill Melnikov. O(α_s^2) corrections to semileptonic decay $b \to cl\bar{\nu}_l$. 2008.

[127] Sandip Biswas and Kirill Melnikov. Second order QCD corrections to inclusive semileptonic $b \to Xcl\bar{\nu}_l$ decays with massless and massive lepton. *JHEP*, 02:089, 2010.

[128] Ryan Gavin, Ye Li, Frank Petriello, and Seth Quackenbush. FEWZ 2.0: A code for hadronic Z production at next-to- next-to-leading order. 2010.

[129] Charalampos Anastasiou, Franz Herzog, and Achilleas Lazopoulos. On the factorization of overlapping singularities at NNLO. 2010.

[130] Vladimir A. Smirnov. *Feynman integral calculus*. Springer, 2006.

[131] Christian Bogner and Stefan Weinzierl. Resolution of singularities for multi-loop integrals. *Comput. Phys. Commun.*, 178:596–610, 2008.

[132] H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. Ann. Math. 79, (1964), 109.

[133] Janusz Gluza, Krzysztof Kajda, Tord Riemann, and Valery Yundin. Numerical Evaluation of Tensor Feynman Integrals in Euclidean Kinematics. 2010.

[134] A. V. Smirnov and V. A. Smirnov. Hepp and Speer Sectors within Modern Strategies of Sector Decomposition. *JHEP*, 05:004, 2009.

[135] Takahiro Ueda and Junpei Fujimoto. New implementation of the sector decomposition on FORM. *PoS*, ACAT08:120, 2008.

[136] Toshiaki Kaneko and Takahiro Ueda. A geometric method of sector decomposition. *Comput. Phys. Commun.*, 181:1352–1361, 2010.

[137] Toshiaki Kaneko and Takahiro Ueda. Sector decomposition via computational geometry. 2010.

[138] Christian Bogner and Stefan Weinzierl. Feynman graph polynomials. *Int.J.Mod.Phys.*, A25:2585–2618, 2010.

[139] L. D. Landau. On analytic properties of vertex parts in quantum field theory. *Nucl. Phys.*, 13:181–192, 1959.

[140] R. J. Eden, P. V. Landshoff, David I. Olive, and J. C. Polkinghorne. *The Analytic S-Matrix*. Cambridge University Press, 1966.

[141] Fyodor V. Tkachov. Landau equations and asymptotic operation. *Int. J. Mod. Phys.*, A14:683–715, 1999.

[142] I.M. Gelfand and G.E. Shilov. *Generalized Functions*, volume 1. Academic Press, New York, 1964.

[143] Mathematica, Copyright by Wolfram Research.

[144] Setsuya Kawabata. A New version of the multidimensional integration and event generation package BASES/SPRING. *Comp. Phys. Commun.*, 88:309–326, 1995.

[145] T. Hahn. CUBA: A library for multidimensional numerical integration. *Comput. Phys. Commun.*, 168:78–95, 2005.

[146] Mark Sofroniou. Format.m: C, FORTRAN77, Maple and TeX Code Generation Package. *http://library.wolfram.com/infocenter/MathSource/60/*, 2005.

[147] Frans Slothouber and et al. ROBODoc 4.99.40. *http://www.xs4all.nl/ rfsber/Robo/robodoc.html*.

[148] Thomas Hahn. Feynman Diagram Calculations with FeynArts, FormCalc, and LoopTools. *PoS*, ACAT2010:078, 2010.

[149] Andrei I. Davydychev and M.Yu. Kalmykov. Massive Feynman diagrams and inverse binomial sums. *Nucl.Phys.*, B699:3–64, 2004.

[150] T. Huber and Daniel Maitre. HypExp, a Mathematica package for expanding hypergeometric functions around integer-valued parameters. *Comput. Phys. Commun.*, 175:122–144, 2006.

[151] Tobias Huber and Daniel Maitre. HypExp 2, Expanding Hypergeometric Functions about Half- Integer Parameters. *Comput. Phys. Commun.*, 178:755–776, 2008.

[152] S. Bethke et al. Experimental Investigation of the Energy Dependence of the Strong Coupling Strength. *Phys. Lett.*, B213:235, 1988.

[153] Gregor Welsh. An Evaluation of Four-Loop Planar One Scale Three Point Master Integrals with up to Ten Massless Propagators. 2011. M.Sci. Dissertation, Durham University (Advisors: Nigel Glover and Gudrun Heinrich).

[154] Mark Zentile. Multi-loop Feynman Diagrams: Two-leg, Massless Propagator Integrals up to Five Loops. 2011. M.Phys. Dissertation, Durham University (Advisors: Nigel Glover and Gudrun Heinrich).

[155] Stefan Beerli. A New method for evaluating two-loop Feynman integrals and its application to Higgs production. 2008. Ph.D. Thesis, ETH (Advisor: Zoltan Kunszt).