

Efficient Algorithms for NLO QCD Event Generators

Dissertation
zur Erlangung des Grades
“Doktor der Naturwissenschaften”
am Fachbereich Physik
der Johannes Gutenberg-Universität
in Mainz

Christopher Schwan
geb. in Bad Kreuznach



Mainz, 2014

Erster Gutachter: [REDACTED]

Zweiter Gutachter: [REDACTED]

Datum der mündlichen Prüfung: 02.12.2014

Dissertation an der Universität Mainz (D77)

Abstract

In this thesis we present techniques that can be used to speed up the calculation of perturbative matrix elements for observables with many legs ($n = 3, 4, 5, 6, 7, \dots$). We investigate several ways to achieve this, including the use of Monte Carlo methods, the leading-color approximation, numerically less precise but faster operations, and SSE-vectorization. An important idea is the use of “random polarizations” for which we derive subtraction terms for the real corrections in next-to-leading order calculations. We present the effectiveness of all these methods in the context of electron-positron scattering to n jets, n ranging from two to seven.

Zusammenfassung

In dieser Arbeit stellen wir Techniken vor, die die Berechnung von perturbativen Matrixelementen für Observablen mit vielen Beinchen ($n = 3, 4, 5, 6, 7, \dots$) beschleunigt. Wir untersuchen einige Wege dies zu erreichen, unter anderem die Verwendung von Monte Carlo Methoden, Farbnäherung, numerisch weniger genaue aber schnellere Operationen und SSE-Vektorisierung. Eine wichtige Idee ist die Verwendung von “Random Polarizations” für welche wir Subtraktionsterme für die reelle Korrektur in nächst-führender Ordnung-Rechnungen herleiten. Wir stellen die Effektivität aller Methoden im Kontext der Elektron-Positron Streuung nach n Jets, n von zwei bis sieben laufend, dar.

Acknowledgments

For giving me the opportunity to continue my work I started with my diploma thesis I thank my supervisor, [REDACTED]. It has been four interesting years I do not want to miss!

I also want to thank my colleagues, especially [REDACTED] and [REDACTED], with whom I had many vital discussions, not only about physics. Thank you for this great time! Furthermore I would like to thank my former colleagues [REDACTED] and [REDACTED] that were always encouraging and interested in problems I had to solve.

I owe a big “Thank you” to my parents and friends that were always patient with me when I was preoccupied, still solving a problem that haunted my mind. Thank you for grounding me!

Finally I would like to thank the DPG for funding my work, the conferences and schools I attended.

Contents

1	Efficient Perturbative QCD Calculations	1
1.1	Introduction	1
1.1.1	Leading-Order Perturbative QCD	2
1.1.2	Color Decomposition and Recursion Relations	3
1.2	Leading and Subleading Color Contributions	7
1.2.1	Application to n -Jet Cross Sections	8
1.3	Random Polarizations	14
1.3.1	Definition and Properties	15
1.3.2	Speed-up of Random Polarizations	17
1.4	Real Color-Ordered Feynman Rules	18
1.4.1	Three- and Four-Gluon-Vertices	19
1.4.2	Quark-Gluon-Vertices	19
1.4.3	Speed-up of RCO Feynman Rules	20
1.4.4	Summation over Random Polarizations	23
1.5	Radiative Correction: Real Subtraction	24
1.5.1	Calculating Radiative Corrections	24
1.5.2	The Origin of Soft- and Collinear Divergences	26
1.5.3	The Subtraction Method	29
1.5.4	The Subtraction Terms	31
1.5.5	The Catani-Seymour Subtraction	38
1.5.6	Summary of the Subtraction Terms	39
1.5.7	Check of the Local Behavior	41
1.5.8	Application to n -Jet Cross Sections	42
2	Monte Carlo Integration	49
2.1	Introduction	49
2.1.1	Plain Monte Carlo Integration	50
2.1.2	Convergence Behavior	50
2.1.3	The “Curse of Dimensionality”	51
2.2	Variance Reduction Techniques	51

2.2.1	Importance Sampling	52
2.2.2	Control Variates	53
2.3	The VEGAS Integration Algorithm	53
2.3.1	Automatic PDF construction in VEGAS	54
2.3.2	The Rebinning Algorithm	55
2.3.3	Results and Chi-Square Test	58
2.3.4	Concluding Discussion	58
2.4	Implementation and Related Details	59
2.4.1	Parallelization of VEGAS	60
3	Conclusion and Outlook	65
A	Appendix	67
A.1	Proofs	67
A.1.1	Proof of the Collinear Phase Space Jacobian	67
A.1.2	Proof for the Soft Case for Quarks	68
A.1.3	Proof of the Soft Function	69
A.1.4	The Soft Function for Helicity Eigenstates	70
A.1.5	The Splitting Functions for the Collinear Limit	70
A.1.6	Catani Seymour Momenta in the Soft and Collinear Limits	70
A.1.7	The Soft Reparametrization	71
A.1.8	Proof of the Spin-Correlation Tensor Transversality	72
A.1.9	Proof of the Integrability of the Additional Term	72
A.1.10	Extremal PDF for Importance Sampling	73
A.2	Identities	74
A.2.1	Spinor Identities	74
A.2.2	Polarization Vector Parametrization and Identities	75
A.3	Majorana- and Weyl-Representations	76
A.3.1	Spinors in Weyl Representation	76
A.3.2	Spinors in Majorana Representation	77
A.3.3	Randomly Polarized Spinors	78
A.4	Feynman Rules	78
A.4.1	QCD Feynman Rules	79
A.4.2	Color-stripped QCD Feynman Rules	80
A.5	Phase Space Generation	81
A.5.1	Introduction	81
A.5.2	Soft- and Collinear-Momentum Generation	82
A.6	Technical Details	85
A.6.1	Instruction Counting with perf	85

A.6.2 Efficiently Memorizing Subcurrents	86
--	----

CHAPTER 1

Efficient Perturbative QCD Calculations

1.1 Introduction

In this thesis we present several techniques and methods that we investigated in order to perform a perturbative QCD calculation as efficient as possible. Efficiency is our main aim here, motivated by the fact that perturbative calculations require considerably more computational power when the number of external particles n of an observables \mathcal{O} is high ($n = 4, 5, \dots$). Efficiency makes the difference between a calculation being *doable* in principle and *practically computable* in a reasonable span of time.

Before we go into detail, however, let us first have a look how this work relates to the field of high energy physics. High energy physics is, roughly speaking, the study of the interaction of elementary particles, that are the fundamental building blocks of matter.

Experimentally, the interactions are studied at particle colliders, e.g. the Large Electron-Positron Collider (LEP) which was superseded by the Large Hadron Collider (LHC), both at CERN. They produce, accelerate and collide a specific kind of particles, e.g. electrons/positrons or protons, as in the case of LEP or LHC, respectively. Detectors are placed at the interaction points of the collisions and record the products. By comparing their data with predictions coming from theory one can compare how well a theory matches the physical reality. The theory that contains all elementary particles we have detected¹ and describes their interaction is the so-called Standard Model (of particle physics). Although it describes the interactions quite well, we already know that the SM leaves open important questions, an important question is for example the matter-antimatter

¹Assuming the found Higgs is *the* SM-Higgs particle and neglecting neutrino oscillations which can be incorporated into the SM

asymmetry (see e.g. Ref [1] for an overview). If experiments would detect new yet unknown particles this would be a sign for physics beyond the standard model and would also, depending on the nature of the particle, suggest the form of new physics. To produce new particles with masses that are higher than the ones that have been detected and known so far, colliders need to operate at ever higher energies. This in turn has the consequence that many known particles are produced, many more than new ones which are the “needle in the haystack”. To measure properties of new particles a good description of the “haystack” is needed, i.e. a proper description of the established theory. The dominant interaction in the SM is given by Quantum Chromodynamics (QCD) with which this thesis is mainly concerned.

In particular, this thesis will rely on perturbative QCD (pQCD), which makes use of the fact that the running coupling α_s of QCD is small at high energies ($\alpha_s(Q = 91.18 \text{ GeV}) = 0.118[2]$) and therefore allows a perturbative approach, i.e. an expansion in powers of α_s that is truncated at a certain point. The lowest-order or leading-order (LO) or tree-level or Born approximation will be discussed in Sec. 1.1.1, the next-to-leading order (NLO) corrections in Sec. 1.5 where we mainly discuss the radiative corrections and present newly derived subtraction terms. Chap. 2 will deal with Monte Carlo integration which will be used to perform the phase-space integrations and is an important ingredient in any perturbative calculation.

1.1.1 Leading-Order Perturbative QCD

Let us first review how a leading order (LO) calculation for an observable \mathcal{O} for n outgoing particles is performed in pQCD:

$$\langle \mathcal{O} \rangle \approx \langle \mathcal{O} \rangle^{\text{LO}} = \int_n d\phi_n J_n^{(0)} |\mathcal{A}_n^{(0)}|^2 \mathcal{O}. \quad (1.1)$$

In plain English this means that the expectation value $\langle \mathcal{O} \rangle^{\text{LO}}$ of the observable \mathcal{O} in leading-order approximation is obtained by integrating it together with its “weight”, i.e. the square of the Born-level matrix element $\mathcal{A}^{(0)}$, over the n -particle phase-space ϕ_n (see App. A.5). In fact, the integration is not performed over the whole phase space but a subset that is selected by the so-called jet-function $J_n^{(0)}$ which is either zero or one. The function is zero if any momentum is too “soft”, i.e. its energy too small, or too “collinear” to any other momentum, i.e. if the angle between any two momenta is too small. This renders the expectation value finite and corresponds to what is done when the expectation value $\langle \mathcal{O} \rangle$ is measured in an experiment: If particles are too soft they will not be detected because a detector has a finite energy resolution; if two or more particles are collinear the detector will not be able to distinguish between them and “sees” them as a single particle. In QCD

we have the additional complication that the elementary particles of the theory, the quarks and gluons, are not the ones that are measured. After being produced, the elementary particles confine in colorless bound-states, the hadrons. The final states measured in an actual experiment are the hadrons or their decay products. These come in bunches that are called jets. The jet functions additionally takes care of how partons are related to the measured jets — at LO only one parton gives rise to a single jet.

1.1.2 Color Decomposition and Recursion Relations

QCD is a theory that is in some aspects similar to Quantum Electrodynamics (QED), but a distinguishing feature is that in addition to electrical charge, the quarks (that correspond to the electrons in QED) carry a second charge, called “color”. This charge can be of three different kinds: red, green, and blue; the antiquarks carry anticharges anti-red, anti-green, and anti-blue. The name color was chosen from the fact that a composite particle made from quarks with charges red, green, and blue is charge-neutral, in analogy to additive color mixing which adds these three colors to a neutral one, white.

Color is a complicating feature in calculations of QCD amplitudes that one can simplify by applying color decomposition (CD). In a few words, CD factorizes the QCD amplitude \mathcal{A} into a sum of partial or primitive amplitudes A_i that possess prefactors taking into account the existence of color:

$$\mathcal{A} = \sum_i C_i A_i. \quad (1.2)$$

The important feature is that the amplitudes A_i no longer depend on color, are calculable in a simple, efficient[3] and process independent way. Furthermore, the amplitudes A_i are gauge invariant.

A detailed explanation on algorithms making use of CD for LO calculations can also be found in the author’s diploma thesis[4] which is why we keep the discussion short. We highlight some properties that are important in the remaining sections and discuss the example of Berends-Giele recursion relations[5] in more detail but otherwise refer to literature[6, 7] where a more exhaustive discussion can be found.

The explicit form of Eq. (1.2) depends on the process and the order of the perturbation theory. Because the decomposition in Eq. (1.2) is not unique there are many different choices for QCD at leading order[8–12] and next-to leading[12–14] calculations.

Let us shortly discuss the example of gluon-scattering $gg \rightarrow (n-2)gg$ to highlight why this decomposition makes the calculation of amplitudes easier. For this class of

processes it can be shown that, at leading-order, the color factors C_i are simply traces of n color matrices with adjoint representation indices $a_1, a_2, \dots, a_n \in \{1, 2, \dots, 8\}$ that are permuted by σ_i ,

$$C_i = \text{tr} \left[\prod_{j=1}^n T^{a_{\sigma_i(j)}} \right], \quad (1.3)$$

and the primitive amplitudes are single function depending only on external momenta p_1, p_2, \dots, p_n of the gluons:

$$A_i = A(p_{\sigma_i(1)}, p_{\sigma_i(2)}, \dots, p_{\sigma_i(n)}). \quad (1.4)$$

For this example the sum in Eq. (1.2) runs over the non-cyclical permutations σ_i that permute the indices of particle j : $\sigma_i(j) \in \{1, 2, \dots, n\}$. For a four gluon-process the sum in Eq. (1.2) is relatively short:

$$\begin{aligned} \mathcal{A} = & \text{tr} [T^{a_1} T^{a_2} T^{a_3} T^{a_4}] A(p_1, p_2, p_3, p_4) + \\ & \text{tr} [T^{a_1} T^{a_2} T^{a_2} T^{a_3}] A(p_1, p_2, p_4, p_3) + \\ & \text{tr} [T^{a_1} T^{a_3} T^{a_2} T^{a_4}] A(p_1, p_3, p_2, p_4) + \\ & \text{tr} [T^{a_1} T^{a_3} T^{a_4} T^{a_2}] A(p_1, p_3, p_4, p_2) + \\ & \text{tr} [T^{a_1} T^{a_4} T^{a_2} T^{a_3}] A(p_1, p_4, p_2, p_3) + \\ & \text{tr} [T^{a_1} T^{a_4} T^{a_3} T^{a_2}] A(p_1, p_4, p_3, p_2). \end{aligned} \quad (1.5)$$

Note that the sum runs over all permutations with particle 1 fixed; thereby the non-cyclicity condition is implemented. By fixing the position of one particle one can easily deduce that in general the sum contains $(n-1)!$ terms. Note also that the primitive amplitude A is always the same function, only its arguments are permuted.

The amplitude can be computed using the so-called Berends-Giele recursion relation, which constructs the *off-shell currents*

$$\begin{aligned} J_\mu(p_2, \dots, p_n) = & \frac{-ig_{\mu\nu}}{P_{2,n}^2} \left[\sum_{i=2}^{n-1} V_3^{\nu\rho\sigma} (-P_{2,n}, P_{2,i}, P_{i,n}) J_\rho(p_2, \dots, p_{i-1}) J_\sigma(p_i, \dots, p_n) \right. \\ & \left. + \sum_{i=2}^{n-1} \sum_{j=i+1}^{n-2} V_4^{\nu\rho\sigma\delta} J_\rho(p_2, \dots, p_{i-1}) J_\sigma(p_i, \dots, p_{j-1}) J_\delta(p_j, \dots, p_n) \right] \end{aligned} \quad (1.6)$$

where $P_{ij} = \sum_{k=i}^j p_k$ is the sum of momenta and V_3, V_4 the color ordered three- and four-gluon vertices (see App. A.4.2). The recursion stops when the off-shell currents have only one argument in which case they are the polarization vectors:

$$J_\mu(p_i) = \varepsilon_\mu(p_i). \quad (1.7)$$

The primitive amplitude is then calculated with

$$A(p_1, p_2, \dots, p_n) = \varepsilon_\mu(p_1) \left[iP_{2,n}^2 \right] J_\mu(p_2, \dots, p_n). \quad (1.8)$$

The factor in the square-brackets is the inverse of the propagator and should be understood as a prescription to remove it in J because it is $-p_1^2$ and therefore on-shell:

$$\sum_{i=1}^n p_i = 0 \quad \Leftrightarrow \quad p_1 = -P_{2,n} \quad \Rightarrow \quad p_1^2 = 0 = P_{2,n}^2. \quad (1.9)$$

If one adds pairs of quarks, the structure of Eq. (1.2) remains, but the color factors and recursion relation change. Because the color-degree of freedom of (anti-)quarks are described with fundamental representation i, j indices the color factors can now also contain so-called open strings, i.e.

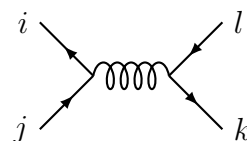
$$\left(T^a \dots T^b \right)_{ij}, \quad (1.10)$$

a product of generators with indices ij from the quark-antiquark pair. For each quark-antiquark-pair there has to be a pair of indices ij . The color factor then is a product of open strings and traces of adjoint representation matrices. To unify the treatment of the color algebra one can use the Fierz-identity

$$T_{ij}^a T_{kl}^a = T_R \left(\delta_{il} \delta_{jk} - \frac{1}{N_C} \delta_{ij} \delta_{kl} \right) \quad (1.11)$$

to rewrite every adjoint representation index into two fundamental representation indices. This is the *color-flow decomposition*[15] (CFD) of QCD amplitudes.

To describe quarks, the recursion relations have to be modified to describe them, as well as the so-called U(1)-gluons that couple two antiquark-quark-pairs without exchanging color. The need for this can be seen when calculating the color-part a Feynman diagram where two antiquark-quark pairs couple via gluon exchange:



$$\propto T_{ij}^a \delta^{ab} T_{kl}^b. \quad (1.12)$$

The generators T_{ij}^a and T_{kl}^b come from the antiquark-quark-gluon vertices, the δ^{ab} from the gluon propagator in between that preserves the gluon type. The result is the Fierz-identity in Eq. (1.12), in which we interpret the Kronecker-deltas as “color-flows”. The first term is a color-flow with a gluon that carries an arbitrary pair color charge and anticharge, the second term corrects the first one which also allows for colorless gluons (red-antired, blue-antiblue, green-antigreen). The two

color-flows of the Feynman diagram are:

$$\delta_{il}\delta_{jk} - \frac{1}{N_C}\delta_{ij}\delta_{kl} = \begin{array}{c} i \quad \quad l \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ j \quad \quad k \end{array} - \frac{1}{N_C} \begin{array}{c} i \quad \quad l \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ j \quad \quad k \end{array} \quad (1.13)$$

Because the second gluon carries no color-charges, it is commonly referred to as U(1)-gluon.

We conclude by listing the advantages of the color decomposition from a point of view where multi-leg computations are important:

1. The color-factors are independent of the momenta and therefore constant over the whole phase-space (p_1, \dots, p_n) . When squaring the amplitude we obtain a *color matrix* M ,

$$|\mathcal{A}|^2 = \sum_i \sum_j A_i^* C_i^* C_j A_j = \vec{A}^\dagger M \vec{A} \quad (1.14)$$

with components $M_{ij} = C_i^* C_j$ that we compute for each scattering process once and for all. Computing the squared amplitude then reduces into a problem of linear algebra for which many optimized routines already exist, e.g. BLAS[16]. The generation of the color matrices is described e.g. in Ref. [4, 17].

2. The primitive amplitudes A can be computed in a diagrammatic way, i.e. by summing up all contributing diagrams, but it is more efficient to make use of the recursion relations which are very simple for the primitive amplitudes. Furthermore the recursion relations can be easily optimized in a way that caches subexpressions appearing more than once are computed only once and then subsequently looked up. For a single primitive amplitude one then obtains a scaling behavior of order n^4 (see Sec. 1.4.3) or even n^3 [18, 19] in the number of external particles. This makes the method very efficient for multi-leg computations where $n = 4, 5, 6, 7, 8$ or even higher.
3. Furthermore, the primitive amplitudes have a much simpler pole-structure. For example, the gluon primitive amplitudes are only singular in the kinematical variables of neighboring momenta, e.g. $A(p_1, p_2, p_3, p_4)$ is only singular when the kinematical variables $s_{12} = 2p_1 \cdot p_2$, s_{23} , s_{34} , and s_{41} go to zero. This behavior is exploited in certain phase-space generators (see App. A.5) that, in general, need to know where the largest contributions of the matrix elements are. We will also need to be aware of this behavior when derive the factorization formulæ for the radiative corrections in Sec. 1.5.

4. Finally one can exploit Eq.(1.2) to derive another approximation, the so-called leading-color (LC) approximation which is motivated by the observation that the diagonal entries of the color matrix are the largest ones. The off-diagonal entries are suppressed by at least a factor $1/N_C^2$ and thus contribute to an observable only of the size of order 10%, see Sec. 1.2.1.2. These entries are commonly referred to as sub-leading color (SLC) contributions and can be treated as a correction to the LC approximation. See Sec. 1.2 for further discussion on this topic.

1.2 Leading and Subleading Color Contributions

In Sec. 1.1.2 we briefly introduced about the color decomposition of QCD, and in this section we describe how we can exploit this in calculations.

The so-called leading color (LC) approximation makes use of the fact that the diagonal entries of the color matrix M defined in Eq. (1.14) contain the largest entries, the off-diagonal ones being smaller by at least a factor $1/N_C^2$, where $N_C = 3$. The diagonal entries are also equal. We can therefore approximate the color matrix M by a diagonal matrix,

$$M \approx C \cdot \mathbb{1} \quad (1.15)$$

where C is a color factor on the diagonal of M that depends on the process, e.g. $C = N_C^{n-1}$ for the process $e^+e^- \rightarrow \bar{q}q + (n-2)g$ and $C = N_C^{n+2}$ for the process $gg \rightarrow ng$. The remaining parts of the color matrix $M' = M - C \cdot \mathbb{1}$ are the subleading-color (SLC) parts and can be treated as a correction to the LC case.

The LC approximation enables us to get rid of the factorial growth of amplitudes because we can now write:

$$\vec{A}^\dagger M \vec{A} \approx \vec{A}^\dagger C \cdot \mathbb{1} \vec{A} = C |\vec{A}|^2 \quad (1.16)$$

where \vec{A} is the vector of partial amplitudes with components A_i that depend on permuted momenta,

$$A_i = A(p_{\sigma_i(1)}, p_{\sigma_i(2)}, \dots, p_{\sigma_i(n)}). \quad (1.17)$$

Because phase space integration treats every momentum likewise, it is symmetric under the relabeling of any two momenta. This allows us to eliminate the permutation σ of the momenta in

$$\begin{aligned} A_i^*(p_{\sigma_i(1)}, p_{\sigma_i(2)}, \dots, p_{\sigma_i(n)}) A_i(p_{\sigma_i(1)}, p_{\sigma_i(2)}, \dots, p_{\sigma_i(n)}) \\ \rightarrow A^*(p_1, p_2, \dots, p_n) A(p_1, p_2, \dots, p_n), \end{aligned} \quad (1.18)$$

so that under the phase space integral we have the identity

$$|\vec{A}|^2 \rightarrow \left(\sum_i 1 \right) |A(p_1, p_2, \dots, p_n)|^2. \quad (1.19)$$

In the example above we just need to compute a single amplitude. The factor $\sum_i 1$ is $(n-1)!$ in the case if every particle is a gluon, and $(n-2)!$ for the processes $e^+e^- \rightarrow \bar{q}q(n-2)g$. At hadron colliders one has, in addition to incoming gluons, to consider processes with incoming quarks from the parton distribution functions. These *channels* then have a different color factor C and must be treated separately.

1.2.1 Application to n -Jet Cross Sections

In this section we compute the total cross sections σ at LO for the process $e^+e^- \rightarrow \gamma^* \rightarrow n$ Jets at $Q = \sqrt{s} = 90$ GeV, which is around the Z-pole mass $M_Z = 91.1$ GeV. We chose this value because many errors of an implementation can be easily detected by checking if the result differs by a factor of 90, $90^2, \dots$. The cross section is computed using Eq. (1.1) with matrix elements computed by our own implementation using the recursive techniques discussed in Sec. 1.1.2. For n outgoing jets at lowest order of the electromagnetic coupling and leading color we have to multiply the matrix elements with the following constant:

$$\frac{1}{4}(4\pi\alpha)^2(2\pi\alpha_s)^{n-2} \left(\sum_q Q_q^2 \right) (\hbar c)^2 \frac{1}{(n-2)!} N_C^{n-1} (n-2)! \quad (1.20)$$

in which the individual parts are

- the spin-averaging factor $\frac{1}{4}$,
- the electromagnetic coupling, $g = \sqrt{4\pi\alpha}$,
- the strong coupling, $g_s = \sqrt{2\pi\alpha_s}$, which was divided by $\frac{1}{\sqrt{2}}$ because of the gluon-field normalization,
- the sum of the squares of charges of the light quark flavors, $\sum_q Q_q^2$, i.e. two up-type quarks, up and charm with $Q = \frac{2}{3}$ and three down-type quarks, down, strange and bottom with $Q = -\frac{1}{3}$ yielding $\sum_q Q_q^2 = \frac{11}{9}$,
- the constant $(\hbar c)^2$ which converts the cross section from natural units in which $\hbar = c = 1$ into a cross section measured in pb,
- a symmetry factor $\frac{1}{(n-2)!}$ because at LC we have $n-2$ gluons that can be interchanged, and finally

- the (leading-)color factor $N_C^{n-1}(n-2)!$ in which $C = N_C^{n-1}$ and $\sum_i 1 = (n-2)!$.

The observable for the total cross section is $\mathcal{O} = \frac{1}{2Q^2}$. The spin-summation is performed as described in Sec. 1.3 and the recursion relations use the optimization of Sec. 1.4.

The jet algorithm $J_n^{(0)}$ we will use in this thesis is the Durham algorithm[20]. This algorithm needs a parameter y_{cut} the determines the “jet-size”. Using this parameter, a generated phase space point $\{p_i = (E_i, \vec{p}_i)\}_{i=1}^n$ is *accepted*, i.e. $J_n^{(0)} = 1$, if for all $i \neq j \in \{1, 2, \dots, n\}$ the *resolution variables*

$$y_{ij} = \frac{2 \min(E_i^2, E_j^2) (1 - \cos \theta_{ij})}{Q^2} \quad \cos \theta_{ij} = \frac{\vec{p}_i \cdot \vec{p}_j}{|\vec{p}_i| |\vec{p}_j|} \quad Q = \sum_{i=1}^n p_i \quad (1.21)$$

are larger than the chosen y_{cut} . If Eq. (1.21) is not fulfilled, the event is *cutted*, i.e. $J_n^{(0)} = 0$ in this case.

One can easily see that Eq. (1.21) sets a bound on the invariants $s_{ij} = 2p_i \cdot p_j$ of the accepted phase space points because

$$s_{ij} = 2p_i \cdot p_j = 2E_i E_j (1 - \cos \theta_{ij}) \geq Q^2 y_{ij} > Q^2 y_{\text{cut}} \quad (1.22)$$

and therefore keeps the propagators $1/s_{ij}$ in the matrix elements of the phase-space integral in Eq. (1.1) finite.

We will choose different values for y_{cut} , the lowest $y_{\text{cut}} = 0.0006$, which is on the lower end of common jet-resolutions, see e.g. Ref. [21]. This makes the phase space integration difficult because it enlarges the range of accepted invariants and thus increases the variance of the integrand in the MC integration (see Chap. 2). The phase space integration with higher y_{cut} are thus easier and we make sure our methods are tested in rather strict conditions.

We use the following couplings, evaluated at $Q = m_Z \approx 90.0$ GeV:

$$\alpha_s = 0.118 \quad \alpha = \frac{1}{127.9} \quad (1.23)$$

and the following constant:

$$(\hbar c)^2 = 0.389\,379\,292 \times 10^9 \text{ GeV pb.} \quad (1.24)$$

The results for different parameters y_{cut} at LO and LC can be found in Tab. 1.1 and were also published in Ref. [22].

1.2.1.1 How Much Numerical Precision is Needed?

Our implementation was designed to be able to make use of arbitrary numerical types so we can also answer the question if we really need double-precision operations.

	y_{cut}	N	E_{LC} [pb]	S_{LC} [pb]
2 Jets	0.001	10^6	4.515×10^1	1.769×10^{-2}
2 Jets	0.002	10^6	4.515×10^1	1.769×10^{-2}
2 Jets	0.0006	10^6	4.515×10^1	1.769×10^{-2}
3 Jets	0.002	10^6	3.137×10^1	3.071×10^{-2}
3 Jets	0.001	10^6	4.043×10^1	1.298×10^{-2}
3 Jets	0.0006	10^7	4.786×10^1	1.582×10^{-2}
4 Jets	0.002	10^8	9.210	1.708×10^{-3}
4 Jets	0.001	10^8	1.632×10^1	3.210×10^{-3}
4 Jets	0.0006	10^8	2.374×10^1	4.998×10^{-3}
5 Jets	0.0006	10^8	7.404	4.020×10^{-3}
6 Jets	0.0006	10^8	1.608	4.084×10^{-3}
7 Jets	0.0006	10^8	2.565×10^{-1}	2.703×10^{-3}

Table 1.1: Results for the jet cross section $e^+e^- \rightarrow \gamma^* \rightarrow \bar{q}q(n-2)g$ with different jet resolution parameters y_{cut} . The numerically integrated result is denoted with E_{LC} , the MC error by S_{LC} . The results were obtained using 5 VEGAS iterations with N integrand evaluations each.

	y_{cut}	N	E_{SLC} [pb]	S_{SLC} [pb]	$\frac{E_{\text{SLC}}}{E_{\text{LC}}}$ [%]
2 Jets			0	0	0
3 Jets			0	0	0
4 Jets	0.002	10^8	-1.538×10^{-1}	7.851×10^{-5}	1.67
4 Jets	0.001	10^8	-2.758×10^{-1}	1.381×10^{-4}	1.69
4 Jets	0.0006	10^8	-4.051×10^{-1}	2.024×10^{-4}	1.71
5 Jets	0.0006	10^8	-3.264×10^{-1}	2.340×10^{-4}	4.41
6 Jets	0.0006	10^8	-1.168×10^{-1}	1.761×10^{-4}	7.26
7 Jets	0.0006	10^8	-2.423×10^{-2}	6.201×10^{-5}	9.45

Table 1.2: Results for the jet cross section $e^+e^- \rightarrow \gamma^* \rightarrow \bar{q}q(n-2)g$ with different jet resolution parameters y_{cut} . The numerically integrated result is denoted with E_{SLC} , the MC error by S_{SLC} . To compare the size of the contribution the last column lists the size of the subleading color contribution relative to the leading color approximation. The results were obtained using 5 VEGAS iterations with N integrand evaluations each.

In Sec. 1.4.3 we show that single-precision operations can offer a significant speedup for matrix elements. Here we will discuss if the reduced precision is enough for LO calculations and if its use yields a speedup in a full calculation.

To obtain meaningful results, we had to take special care during phase space generation because the QCD antenna algorithm (see App. A.5) quickly runs into phase space regions where the sum of two lightlike momenta k_a and k_b is very small and therefore lightlike with single-precision arithmetic, i.e. $(k_a + k_b)^2 = 0$ whereas with double-precision arithmetic it behaves correctly. These phase-space points contain very soft and/or collinear momenta and would be rejected by the jet algorithm afterwards, but depending on the implementation of the jet algorithm it may still happen that they pass the test, e.g. because the generated momenta do not fulfill momentum conservation (because they were generated with too little numerical precision, as described above) or contain infinities. In that case the matrix elements also return infinities and “destroy” the integration which also yields infinity. To overcome this problem in our implementation these points can be detected by checking the phase space weight for zero. If this is the case, it must be rejected. Although this works for LO, it does no longer work the real correction at NLO which must allow for soft momenta. See Sec. 1.5.8 for more discussion on this. Additionally, as suggested in Ref. [19], we used Kahan summation[23] in the MC integrator to preserve as much precision as possible when the number of integrand evaluations is large.

The results can be found in Tab. 1.3. Compared to the results obtained with double precision, we see that they agree within the error bound, but differ numerically which is mainly due to the fact that the random numbers are generated differently.²

The speedup is listed in Tab. 1.4 where we note two things: There is a speedup, but it drops for many particles. This is explained as follows: A speedup through less precision was gained by performing the numerical computations with so-called SIMD³ instructions that perform a single instruction on multiple values. An example is the vector sum $p'^\mu = p^\mu + k^\mu$ for which we can use the ADDPS⁴ instruction that

²Single-precision numbers $x \in [0, 1)$ can represent 2^{24} different values, whereas double-precision numbers can represent 2^{53} different values. Random numbers are usually 32 bit integers, which is why single precision random numbers are generated with one integer, double-precision numbers with two of them.

³SIMD stands for *single instruction multiple data*. These are special instructions executed by the CPU that perform the same instruction multiple times on the values of a special register that holds more than one value. In our case we use the SSE (Streaming SIMD Extension) from Intel that operates on 128 bit-wide registers. These can hold four single-precision numbers or two double-precision numbers.

⁴This is an example of a SSE instruction, see³

performs the operation

$$\begin{pmatrix} p'_0 \\ p'_1 \\ p'_2 \\ p'_3 \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} + \begin{pmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{pmatrix} \quad (1.25)$$

as fast as a summation of a single component. It is therefore four times faster compared to case where we compute the vector sum component-wise. The disadvantage of this approach is that the components must fit into SSE 128 bit register, i.e. the components must be single-precision (32 bit). A vector sum with double-precision is realized using two of those registers, i.e. a vector sum in single-precision is twice as fast. There are a number of operations that perform many operations at once, including vector addition, subtraction, division and multiplication with a scalar and even scalar products. This is why amplitudes can be computed faster in single- and double-precision (see also Sec. 1.4.3), but phase-space generation cannot be optimized this way, because it uses mainly scalar operations (sine, cosine, square-root, calculations with random numbers) that cannot be vectorized in the way described above. What happens when we perform phase-space integrations for a high number of particles is that many phase-space points are cut and thus a lot of phase-space points are generated before a matrix element is computed. This is of course dependent on the actual phase-space generator, but also a general behavior of phase space generators since the average energy for a single particle drops with n (assuming a constant center-of-mass energy Q) and is therefore likely soft for high n . The momenta \vec{p}_i also become more collinear because n momenta have to “fit” into a the same space \mathbb{R}^3 .

1.2.1.2 Subleading Color Correction

To justify the use of the leading-color approximation we have to calculate the contribution of the subleading-color correction. These can be found in Tab. 1.2. Note that the results were obtained by computing the subleading-color contributions of the process $e^+e^- \rightarrow \bar{q}q(n-2)g$; for a true full-color result we would also need to add the processes with more than one quark-pair, i.e. two for the case of four and five jets, and two and three quark-pairs for the case of six and seven jets.

The two and three jet result does not receive a SLC correction because these cases are trivial, i.e. LC agrees with FC. The remaining SLC correction are smaller as 10 % which is decreased even further by the previously mentioned processes with more quark-pairs, since their matrix elements are positive.

	Double Precision		Single Precision	
	E [pb]	S [pb]	E [pb]	S [pb]
2 Jets, $N = 10^7$	4.512×10^1	1.277×10^{-2}	4.514×10^1	1.273×10^{-2}
	4.513×10^1	1.248×10^{-2}	4.514×10^1	1.245×10^{-2}
	4.513×10^1	1.243×10^{-2}	4.512×10^1	1.240×10^{-2}
	4.512×10^1	1.243×10^{-2}	4.512×10^1	1.239×10^{-2}
	4.514×10^1	1.243×10^{-2}	4.512×10^1	1.239×10^{-2}
3 Jets, $N = 10^7$	4.784×10^1	1.186×10^{-1}	4.796×10^1	1.195×10^{-1}
	4.780×10^1	4.434×10^{-2}	4.784×10^1	4.419×10^{-2}
	4.789×10^1	3.183×10^{-2}	4.787×10^1	3.178×10^{-2}
	4.784×10^1	2.791×10^{-2}	4.786×10^1	2.788×10^{-2}
	4.784×10^1	2.649×10^{-2}	4.785×10^1	2.648×10^{-2}
4 Jets, $N = 10^7$	2.374×10^1	1.907×10^{-1}	2.381×10^1	2.113×10^{-1}
	2.382×10^1	4.777×10^{-2}	2.368×10^1	4.945×10^{-2}
	2.372×10^1	3.098×10^{-2}	2.376×10^1	3.154×10^{-2}
	2.380×10^1	2.792×10^{-2}	2.378×10^1	2.808×10^{-2}
	2.375×10^1	2.691×10^{-2}	2.371×10^1	2.696×10^{-2}
5 Jets, $N = 10^7$	7.478	2.511×10^{-1}	7.604	4.097×10^{-1}
	7.275	7.362×10^{-2}	7.274	1.195×10^{-1}
	7.461	4.026×10^{-2}	7.369	4.287×10^{-2}
	7.361	2.475×10^{-2}	7.381	2.512×10^{-2}
	7.386	2.133×10^{-2}	7.378	2.085×10^{-2}
6 Jets, $N = 10^8$	1.628	8.009×10^{-2}	1.667	6.629×10^{-2}
	1.603	2.669×10^{-2}	1.593	1.593×10^{-2}
	1.604	1.044×10^{-2}	1.614	1.232×10^{-2}
	1.608	6.233×10^{-3}	1.608	1.192×10^{-2}
	1.608	4.236×10^{-3}	1.604	7.279×10^{-3}
7 Jets, $N = 10^8$	2.369×10^{-1}	1.333×10^{-2}	2.616×10^{-1}	2.524×10^{-2}
	2.376×10^{-1}	6.406×10^{-3}	2.425×10^{-1}	1.193×10^{-2}
	2.597×10^{-1}	8.046×10^{-3}	2.500×10^{-1}	8.890×10^{-3}
	2.562×10^{-1}	6.865×10^{-3}	2.441×10^{-1}	7.383×10^{-3}
	2.533×10^{-1}	5.637×10^{-3}	2.954×10^{-1}	3.034×10^{-2}

Table 1.3: Five VEGAS iterations each with N calls, computed with single- and double-precision code. The expectation values E and their errors S for the jet cross section σ were computed as described in Sec. 1.2.1 with $y_{cut} = 0.0006$.

	Double Precision	Single Precision	Quotient
	t_d [s]	t_s [s]	t_d/t_s
2 Jets	98	56	1.75
3 Jets	193	123	1.57
4 Jets	348	246	1.41
5 Jets	601	457	1.32
6 Jets	10 252	7822	1.31
7 Jets	18 452	15 471	1.19

Table 1.4: Performance comparison of our single- and double-precision implementations for the jet cross section.

1.3 Random Polarizations

We now switch to another topic and introduce the so-called *random polarizations*[24]. We talked about polarizations so far only in the context of recursion relations where they appeared as the trivial case in the off-shell currents with one external leg, see Eq. (1.7). Apart from the momentum the polarizations also depend on the helicity of the massless boson, or spin-three component in the case of fermions. Although one can measure this property, many detectors will not measure this freedom so that in the theoretical calculation one has to sum over them. For n in- and outgoing particles this is a sum over 2^n summands,

$$\sum_{\lambda_1=\pm} \sum_{\lambda_2=\pm} \cdots \sum_{\lambda_n=\pm}, \quad (1.26)$$

and therefore clearly a bottle-neck in multi-leg calculations.

The idea of random polarizations is to replace the sum over spin-/helicity states by an integral over a continuous *helicity angle* ϕ ,

$$\sum_{\lambda=\pm} \varepsilon_\lambda^\mu (\varepsilon_\lambda^\nu)^* = \frac{1}{2\pi} \int_0^{2\pi} d\phi \varepsilon^\mu(\phi) (\varepsilon^\nu(\phi))^*, \quad (1.27)$$

with appropriately defined polarizations $\varepsilon^\mu(\phi)$. The additional integrals are then merged with the phase-space integration and performed together in a *single* Monte Carlo integration. Thereby we avoid the time-consuming summation over all 2^n spin-configurations. Because MC integration converges independently of the number of integration dimensions (see Sec. 2.1.2) we speed up our calculation. However, since we modify the function that we integrate, we also increase the error of the MC integration. How speedup on the one side and the increased error on the other side yield in an effective performance gain x is discussed in Sec. 1.3.2, but first we define these new polarizations.

1.3.1 Definition and Properties

The polarizations satisfying Eq. (1.27) are defined as a linear combination of the two helicity-eigenstates:

$$\varepsilon^\mu(\phi) = e^{+i\phi}\varepsilon_+^\mu + e^{-i\phi}\varepsilon_-^\mu. \quad (1.28)$$

It is easily checked that Eq. (1.27) holds true:

$$\varepsilon^\mu(\phi)(\varepsilon^\nu(\phi))^* = \sum_{\lambda=\pm} \varepsilon_\lambda^\mu(\varepsilon_\lambda^\nu)^* + 2\Re\left(e^{+i\phi}\varepsilon_+^\mu\varepsilon_-^\nu\right); \quad (1.29)$$

performing the integral yields the first term on the right-hand side, the second one vanishes as expected. Note that we made use of the freedom to choose the relative phase as

$$\varepsilon_-^\mu = (\varepsilon_+^\mu)^*. \quad (1.30)$$

Together with the definition (1.28) of the random polarization we can rewrite it as

$$\varepsilon^\mu(\phi) = 2\Re\left\{e^{+i\phi}\varepsilon_+^\mu\right\}, \quad (1.31)$$

in other words the polarization is real under complex-conjugation. In Sec. 1.4 we will show how this speeds up many leg computations even further.

Spinors for massless particles are treated in the same way,

$$|\phi\rangle = e^{+i\phi}|+\rangle + e^{-i\phi}|-\rangle, \quad (1.32)$$

so the completeness relation reads

$$\sum_{\lambda=\pm} |\lambda\rangle\langle\lambda| = \frac{1}{2\pi} \int_0^{2\pi} d\phi |\phi\rangle\langle\phi|. \quad (1.33)$$

To derive real spinors we proceed analogously to the case of vectors, but we need a relation like (1.30). In Weyl-representation this identity does not hold true:

$$|p+\rangle^{\text{Weyl}} \neq (|p-\rangle^{\text{Weyl}})^*. \quad (1.34)$$

However, in Majorana-representation (see also App. A.3) we find

$$|p+\rangle^{\text{Majorana}} = \mp (|p-\rangle^{\text{Majorana}})^*. \quad (1.35)$$

The additional sign may occur depending on the choice of parametrization and the momentum. This does not pose a problem since we can always introduce a phase factor i canceling the sign. This is important because otherwise we would obtain a purely imaginary spinor:

$$|\phi\rangle = 2i\Im\left\{e^{+i\phi}|+\rangle\right\}. \quad (1.36)$$

Note that for massive particles it is not possible to derive real spinors because the Dirac equation for our metric $(+, -, -, -)$,

$$(\gamma^\mu p_\mu - m) u(\vec{p}) = 0, \quad (1.37)$$

is inconsistent with this requirement. If we require $u(\vec{p}) = u^*(\vec{p})$ the Dirac equation implies $\gamma_\mu = \gamma_\mu^*$, i.e. a real representation of the Dirac gamma matrices is required. In App. A.3 we find that the Majorana representation is purely imaginary, so we can factor out an imaginary unit i that leads to a representation of gamma matrices $\gamma_\mu = i\gamma'_\mu$ with metric $(-, +, +, +)$:

$$\{\gamma_\mu, \gamma_\nu\} = 2g_{\mu\nu} \Leftrightarrow \{\gamma'_\mu, \gamma'_\nu\} = -2g_{\mu\nu}. \quad (1.38)$$

With this metric the Dirac equation reads

$$(\gamma'_\mu p^\mu + im) u(\vec{p}) = 0, \quad (1.39)$$

i.e. it is not possible to find a way in which we can treat massive spinors real-valued *and* gain a computational advantage as we did for massless spinors.

1.3.1.1 Random Polarization Vectors

Let us examine Eq. (1.31) further. We parametrize the helicity eigenstate vectors (see also App. A.2.2) by

$$\varepsilon_+^\mu(p, q) = \frac{\langle q- | \gamma^\mu | p- \rangle}{\sqrt{2} \langle q- | p+ \rangle}, \quad (1.40)$$

with $|p-\rangle$ and $|p+\rangle$ being the negative- and positive-helicity (Weyl-)spinors for a lightlike momentum p . The *reference momentum* q may be chosen arbitrarily; a convenient choice is $q = (p^0, -p^1, -p^2, -p^3)$ ensuring the denominator in Eq. (1.40) always being far enough away from zero. We will use this choice throughout this thesis. Interpreting γ^0 as a parity transformation we see that

$$\varepsilon_+^0 \propto \langle q- | \gamma^0 | p- \rangle = \langle q- | q- \rangle = 0 \quad (1.41)$$

so the transversality condition $p \cdot \varepsilon(p) = 0$ of the polarization vector becomes

$$\vec{p} \cdot \vec{\varepsilon}(\vec{p}) = 0. \quad (1.42)$$

The set of vectors $\vec{\varepsilon}$ orthogonal to \vec{p} are unique up to a rotation because the length of $\vec{\varepsilon}$ is e.g. determined by the polarization sum. Therefore the choice of ϕ makes the solution of Eq. (1.42) unique. This formula also shows that random polarizations include the idea presented by Giele, Stavenga, and Winter[19]. Instead of choosing a random helicity angle ϕ in the Monte Carlo integration the authors chose a random vector $\vec{\varepsilon}$ perpendicular to the momentum \vec{p} which realizes Eq. (1.42).

1.3.2 Speed-up of Random Polarizations

In this section we investigate if rewriting the calculations to random polarizations actually give a gain in computational performance. First we summarize the advantages and disadvantages of both methods to devise a way to compare them.

Random polarizations offer faster calculations of a single amplitude because the spin-sum is circumvented, but they introduce new integration dimensions and therefore increase the MC error. A spin-summed amplitude with helicity eigenstates suffers from the exponential growth of the summations, but one can memorize the subcurrents and look them up in subsequent calculations that need them again. Therefore the scaling is, although still exponential, in practice much better than 2^n .

To account for the increased error in random polarizations, we define the speedup x by

$$x = f \left(\frac{\sigma_{\text{HS}}}{\sigma_{\text{RP}}} \right)^2 \quad (1.43)$$

where $f = \frac{t_{\text{HS}}}{t_{\text{RP}}}$ is the quotient of the duration t_{RP} for a single amplitude computed with random polarizations and t_{HS} the duration for the spin-summed amplitude and σ the standard deviations for each method computed with the same size of statistics N . This formula is derived by assuming

$$\frac{\sigma_{\text{HS}}}{\sqrt{N}} = \frac{\sigma_{\text{RP}}}{\sqrt{M}}, \quad (1.44)$$

i.e. we assume that both methods eventually yield the same error when integrated with sample sizes N and M , respectively. The effective sample size $M = fN/x$ accounts for the RP being faster by f so that from the fN we only need a fraction, x , to match the error.

For the process $e^+e^- \rightarrow \gamma^* \rightarrow \bar{q}q + (n-2)g$ at leading order and leading color we find the results in Tab. 1.5. For the simplest case, $e^+e^- \rightarrow \bar{q}q$, we see that random polarizations add a significantly higher error (relative to the helicity eigenstates) but the absolute error of the result is still acceptably low. This difference in the error can be explained by the fact that random polarizations make the integrand non-factorizable, whereas for helicity eigenstates the matrix element is factorizable:

$$\sum_{\lambda_1=\pm} \sum_{\lambda_2=\pm} \sum_{\lambda_3=\pm} \sum_{\lambda_4=\pm} |\mathcal{A}|^2 \propto (1 + \cos^2 \theta). \quad (1.45)$$

As we will explain in Sec. 2.3.4, VEGAS able to find the optimal grid for a factorizable and there yields a much smaller error compared to the non-factorizable case.

	$\frac{\sigma_{\text{HS}}}{\sigma_{\text{RP}}}$	f	x
$e^+e^- \rightarrow \bar{q}q$	0.009	2.98	0.00
$e^+e^- \rightarrow \bar{q}qg$	0.460	5.30	1.12
$e^+e^- \rightarrow \bar{q}qgg$	0.496	11.10	2.73
$e^+e^- \rightarrow \bar{q}qggg$	0.472	17.86	3.98
$e^+e^- \rightarrow \bar{q}qgggg$	1.007	32.36	32.78

Table 1.5: Comparison of random polarizations vs. helicity eigenstates. Note that the standard deviations σ depend not only on the matrix elements, but also on the phase-space generator (the QCD antenna generator, see App. A.5) and on the probability distribution functions (see Sec. 2.3) generated by VEGAS' last iteration (in total five) with $N = 10^7$ for helicity summation and $N = 10^8$ for random polarizations. Note also that the speedup f scales exponentially — every number is about twice the size of its previous number — although not as 2^n , the number of spin-configurations. This is because we can memorize subcurrents that have the same helicity configurations.

For the more involved process we can see that the error is still larger but only by a only a factor of about two, so if the amplitudes can be computed four times faster random polarizations pay off — this is already the case for three jets.

Note that the values for f are very sensitive to the way one implements the amplitude computation and, more importantly, the way the subcurrents are reused. In App. A.6.2 we present an algorithm how to efficiently store and lookup subcurrents.

1.4 Real Color-Ordered Feynman Rules

In the last section we have shown that massless particles may be described by real-valued polarizations. We now discuss how to treat the Feynman rules so a complete perturbative calculation can be performed with real quantities; as we will show in Sec. 1.4.3, numerical computations benefit from this.

In QED this is easily possible; the photon-fermion vertex and both the photon and fermion propagator have imaginary units which we can factor out from the amplitude. As we will show in Sec. 1.4.2 the imaginary numbers in the gamma matrices γ^μ can be avoided as well.

1.4.1 Three- and Four-Gluon-Vertices

In addition to the QED Feynman rules, QCD has additional three- and four-gluon vertices. Since there is a relative imaginary unit between those two, factoring them out is no longer possible on the level of Feynman diagrams.

Opposed to the Feynman rules, the color-stripped three- and four-gluon vertices and the color-stripped propagator both have an imaginary unit in front. Factoring out these and using the real polarizations defined before is sufficient to perform a computation avoiding complex numbers. This was already done before in [19] where Berends-Giele recursion relation are used to compute the amplitudes.

In the following we assume that this phase is factored out; because the phase is canceled when we square the matrix element, we ignore them in the following completely. If we want to work on the level of amplitudes instead of squared amplitudes, e.g. to compare against another implementation, one has to multiply our amplitudes with the factor

$$(i^2)^{n-1} = (-1)^{n-1} \tag{1.46}$$

if n is the number of external particles. This is justified by the following derivation: One external particle is simply the polarization vector and does not need an additional phase. Adding one more particle means connecting it with a propagator and a vertex, yielding i^2 . For each additional particle we have to multiply with i^2 , and thus we arrive with Eq. (1.46).

1.4.2 Quark-Gluon-Vertices

To include fermions, let us first consider how a single fermion line in tree-level structures look like:

$$\langle \phi | \gamma_\mu | \varphi \rangle \quad \langle \phi | \gamma_\mu \frac{p^\alpha \gamma_\alpha}{p^2} \gamma_\nu | \varphi \rangle \quad \dots \quad \langle \phi | \gamma_{\mu_1} \prod_{i=2}^n \left(\frac{p_i^{\alpha_i} \gamma_{\alpha_i}}{p_i^2} \gamma_{\mu_i} \right) | \varphi \rangle \tag{1.47}$$

These expressions correspond to the case where one gluon, two gluons, \dots , and n gluons couple to a fermion line. Note that there are $n - 1$ propagators and n vertices. Together with the γ^0 from $\langle \phi | = |\phi\rangle^\dagger \gamma^0$ we always have an even number of gamma matrices in a fermion-line. Note that in order for the spinors to be real, the spinors need to be in the Majorana representation, as are the gamma matrices. In this representation the matrices are purely imaginary, see App. A.3. This means that a closed fermion line always has an even number of imaginary units i and therefore is real. This ensures that gluon currents coupling to fermion-lines preserve their real-valuedness.

How are the imaginary units in the gamma matrices treated? For that, first define the “real gamma matrices” γ'_μ ,

$$\gamma_\mu = i\gamma'_\mu, \quad (1.48)$$

which are the basis-vectors of the Clifford-Algebra $\text{Cl}_{3,1}$ in which the number of positive-squaring vectors ($\gamma_0^2 = 1$) is exchanged with the number of negative-squaring vectors (e.g. $\gamma_1^2 = -1$):

$$\gamma'_\mu \gamma'_\nu + \gamma'_\nu \gamma'_\mu = -2g_{\mu\nu}. \quad (1.49)$$

Using these matrices we can discuss the (anti-)fermion-currents that appear in Berends-Giele type recursion relations. A fermion off-shell current computes the numerical result of expressions of the following type:

$$\left(\frac{(p+q)_\alpha \gamma^\alpha}{(p+q)^2} \right) F_\mu(q) \gamma^\mu |\phi\rangle = - \left(\frac{(p+q)_\alpha \gamma'^\alpha}{(p+q)^2} \right) F_\mu(q) \gamma'^\mu |\phi\rangle \quad (1.50)$$

with $F^\mu(q)$ the gluon-current coupling to this quark line. Note that every additional gluon adds another pair of gamma matrices so that we can always factor out $i^2 = -1$ and work with the real-valued γ'^μ instead. Anti-fermion currents can be treated as fermion currents (apart from the differing propagator). If both currents couple together, one makes use of the following identity to convert the anti-fermion current into a bra-like object:

$$\left(F_{\mu_1}^1 \dots F_{\mu_{2n}}^{2n} \gamma^{\mu_1} \dots \gamma^{\mu_{2n}} |\phi\rangle \right)^T i\gamma^0 = \langle \phi | \gamma^{\mu_{2n}} \dots \gamma^{\mu_1} \gamma^0 F_{\mu_1}^1 \dots F_{\mu_{2n}}^{2n} \quad (1.51)$$

with real-valued functions $F_{\mu_i}^i$.

1.4.3 Speed-up of RCO Feynman Rules

The speedup obtained by performing the entire computation with double variables instead of `std::complex<double>` variables can be seen in Fig. 1.1. The program used to obtain these numbers is written using C++-templates where the numerical type T is first left unspecified and then two programs are obtained for which T = double and T = `std::complex<double>`. This way we make sure both types are treated exactly alike (apart from optimizations the compiler makes).

The asymptotic speed up of 14.01 is quite high and has different reasons. When both programs are compiled again with the additional compiler optimization flag `CXXFLAGS="-ffast-math"` the difference shrinks about a factor of

$$\frac{14.01}{5.52} \approx 2.54 \quad (1.52)$$

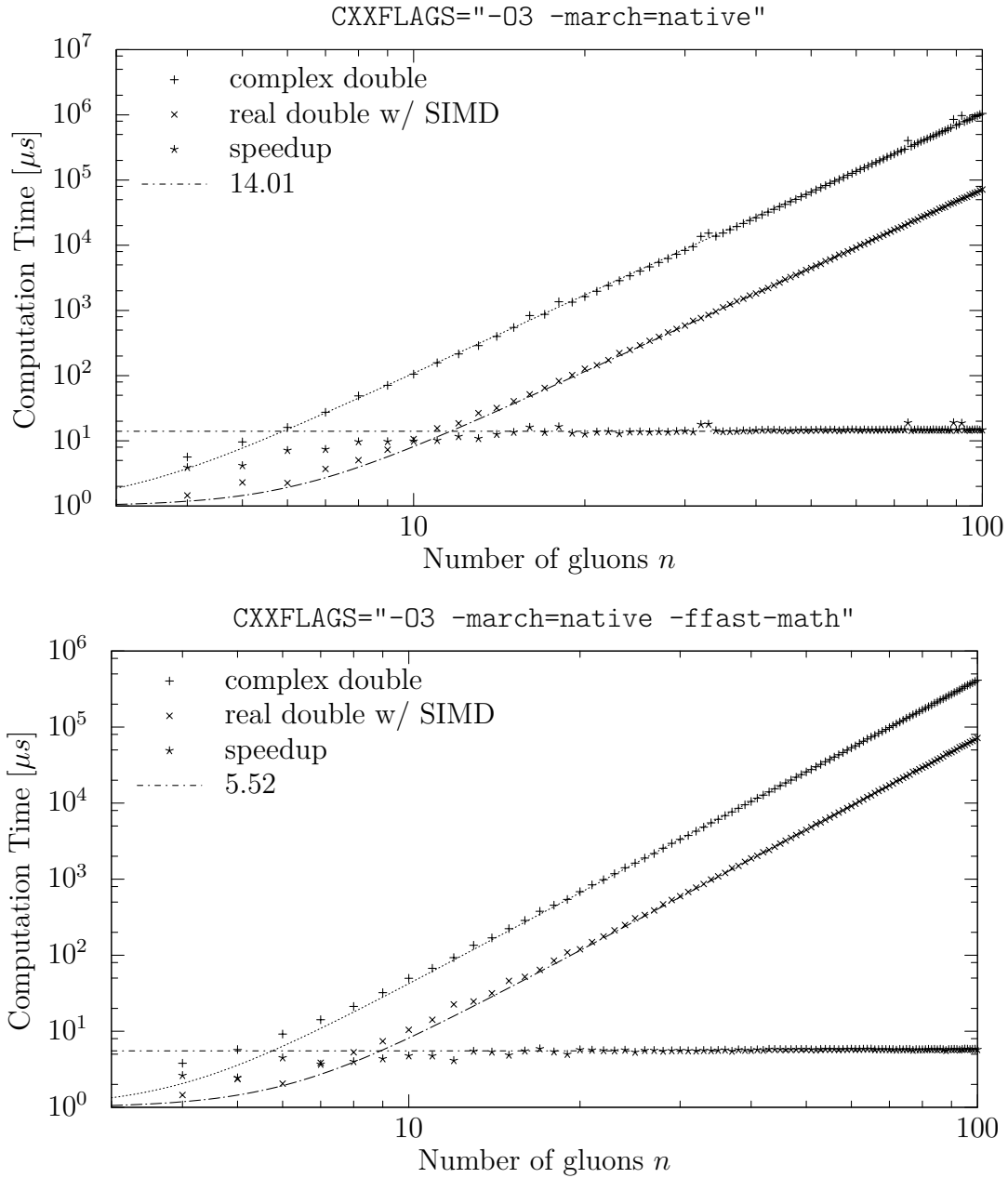


Figure 1.1: Speedup of real color-ordered Feynman rules. Both plots show the evaluation time for a single leading-color and leading order gluon amplitude ($gg \rightarrow (n-2)g$). The times t were fitted to a polynomial[15] $t = an^4 + b$ with constants a and b . The n^4 behavior is only asymptotically true which is why deviations for small n are visible. The speed-up is the quotient between the data sets and fitted by a constant. The “spikes” around $t \approx 30, 70, 90$ appear at different locations when the measurement is performed again or completely disappear; see also second plot. Here the same measurement is shown with the additional compiler optimization `-ffast-math`.

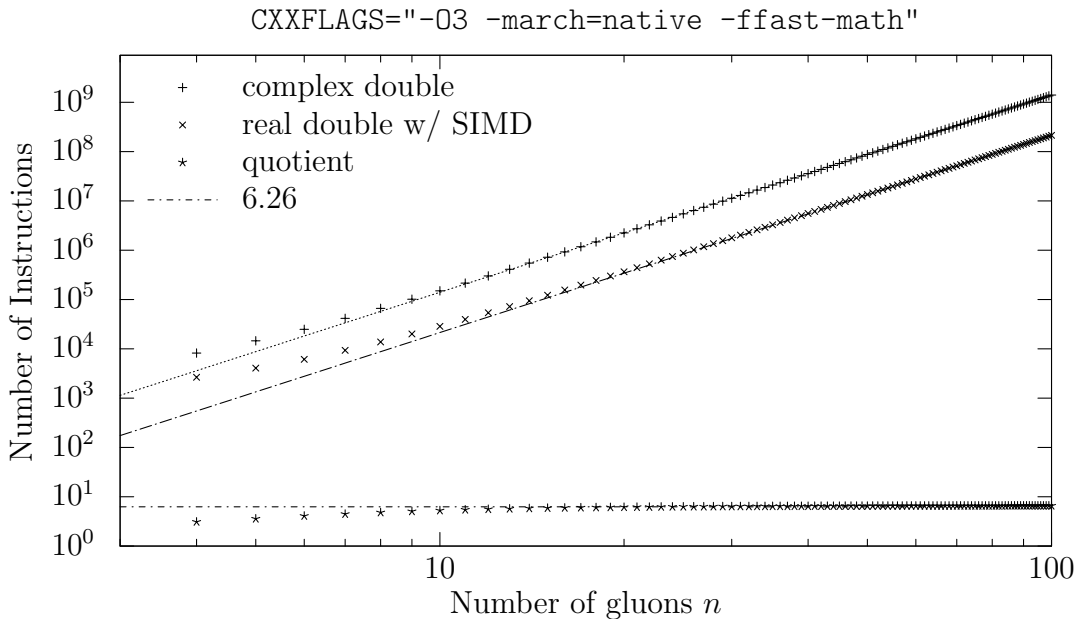


Figure 1.2: *Speedup of real color-ordered Feynman rules in terms of CPU-instructions I . The data sets are fitted with a polynomial $I = an^4 + b$ with constants a and b . The quotient of both data sets is fitted with a constant.*

which can be explained by looking into the assemblies generated by the compiler. There one finds that complex multiplications are not performed directly but by the function `__muldc3`⁵ which takes care of the many special cases when any component of any complex number takes on a special value, such as infinity or “not a number” as required by IEEE 754[25]. Using the compiler switch `-ffast-math` instructs GCC not to adhere to this standard and is thereby able to optimize away the function call by a simple implementation of complex multiplication which is responsible for the speedup of 2.54.

The remaining factor can be understood by looking at Fig. 1.2 where the number of instructions for both the real- and the complex-valued implementation shown. Working with complex numbers results in 6.26 times more instructions which explains the remaining factor 5.52 in the second plot of Fig. 1.1.

Throughout this section all floating-point operations were performed with SSE SIMD instructions, but we found that this resulted only in a moderate speedup of 32%. This is most likely due to the use of double-precision numbers that are 64-bit wide and therefore only two of them fit into a single 128-bit SSE register.

⁵ `__` marks the function as one of GCC’s `libgcc_s.so` library, `mul` means multiplication, `dc` means double precision for complex numbers and `3` marks this function as an operation involving three (complex-)numbers.

The x86_64 architecture that was used to run the programs on also already uses SSE registers and (scalar) SSE operations for floating point computations. If performed with single precision numbers the speedup is 130%, because now many operations can be done with in a single instruction, e.g. vector addition, subtraction, multiplication etc.:

	Precision	Gluons #	Time [ms]	Instructions ⁶ 10 ⁶
w/ SIMD	double	100	72	213
	single	100	39	114
w/o SIMD	double	100	95	355
	single	100	90	356

For double-precision a similar speedup could be achieved by using the AVX 256-bit registers and instructions that can handle four doubles at once. These are, however, only available in the most recent processors.

1.4.4 Summation over Random Polarizations

In the sections before we described how to replace the helicity summation by an integration, i.e. the replacement

$$\sum_{\lambda_1=\pm} \cdots \sum_{\lambda_n=\pm} \longrightarrow \frac{1}{(2\pi)^n} \int_0^{2\pi} d\phi_1 \cdots \int_0^{2\pi} d\phi_n. \quad (1.53)$$

Sometimes, however, it is necessary to sum over certain particles to obtain the exact polarization sum. This is still possible with random polarizations by choosing a random angle η , e.g. $\eta = 0$, and then use the fact that

$$\frac{1}{2} \sum_{\phi \in \{0, \pi\}} (\varepsilon^\mu(\phi + \eta))^* \varepsilon^\nu(\phi + \eta) = \sum_{\lambda=\pm} \varepsilon_\lambda^{\mu*} \varepsilon_\lambda^\nu \quad (1.54)$$

which is derived from Eq. (1.29) where the spin-dependent part cancels between both terms because $e^{i0} = -e^{i\pi}$. From this follows that we can rewrite all integrals in Eq. (1.53) again as sums:

$$\frac{1}{(2\pi)^n} \int_0^{2\pi} d\phi_1 \cdots \int_0^{2\pi} d\phi_n = \frac{1}{2^n} \sum_{\phi_1 \in \{0, \pi\}} \cdots \sum_{\phi_n \in \{0, \pi\}} \quad (1.55)$$

which is, compared to the sums in Eq. (1.53), a basis transformation. However, in this summation we retain the property that the vectors are real-valued and thus can be computed faster than with complex quantities.

⁶The number of instructions was counted using the perf-framework, see App. A.6.1.

1.5 Radiative Correction: Real Subtraction

A full NLO calculation consists of basically two corrections in addition to the LO approximation:

$$\langle \mathcal{O} \rangle^{\text{NLO}} = \langle \mathcal{O} \rangle^{\text{LO}} + \langle \mathcal{O} \rangle_{\text{Virtual}}^{\text{NLO}} + \langle \mathcal{O} \rangle_{\text{Real}}^{\text{NLO}} \quad (1.56)$$

The first part is known from (1.1), the second part is the virtual correction

$$\langle \mathcal{O} \rangle_{\text{Virtual}}^{\text{NLO}} = \int_n d\phi_n J_n^{(0)} 2\Re \left(\mathcal{A}_n^{(0)*} \mathcal{A}_n^{(1)} \right) \mathcal{O} \quad (1.57)$$

which is the interference term between a born amplitude and the one-loop amplitude $\mathcal{A}^{(1)}$. The superscript denotes the number of loops in the amplitude. The interference term can be derived by writing down the series expansion in the coupling parameter $g \ll 1$ (that was suppressed in the formulas above):

$$|\mathcal{A}_n|^2 = \left| g^n \sum_{k=0}^{\infty} g^k \mathcal{A}^{(k)} \right|^2 = g^{2n} \left\{ |\mathcal{A}_n^{(0)}|^2 + g 2\Re \left(\mathcal{A}_n^{(0)*} \mathcal{A}_n^{(1)} \right) + \dots \right\}. \quad (1.58)$$

The calculation of one-loop amplitudes is more complicated than Born amplitudes; there is an additional loop integral that has to be performed and the number of Feynman diagrams that contribute is higher because the topologies are more involved. Loop calculations also require to perform a renormalization procedure, which means redefining quantities such as coupling constants, masses, and field strengths. Ultimately this is needed to relate the constants appearing in the Lagrangians \mathcal{L} with the constants that can be measured in experiments. This can be done by adding additional terms to the Lagrangian, the counter-terms, that render the loop integrations (UV-)finite, i.e. they cancel the divergences that arise from integrations where the loop momentum k is large.

1.5.1 Calculating Radiative Corrections

It turns out that the virtual correction, even after properly renormalizing all quantities such as fields, masses and coupling constant(s), is still (IR-)divergent. The divergence is canceled in infrared safe observables by adding the real or radiative correction

$$\langle \mathcal{O} \rangle_{\text{Real}}^{\text{NLO}} = \int_{n+1} d\phi_{n+1} J_n^{(1)} \left| \mathcal{A}_{n+1}^{(0)} \right|^2 \mathcal{O}. \quad (1.59)$$

The real correction looks very similar to the LO contribution for $n + 1$ instead of n final states. The difference here is that the jet function $J_n^{(1)}$ selects phase-space points that have exactly one soft- and/or collinear (pair of) momenta. The phase-space integral over the additional particle is divergent in a way that the remaining divergences of the virtual correction are canceled. It can be shown that

this cancellation works order for order in perturbation theory for QED[26] and QCD[27, 28].

To work with divergent quantities it is important that these are treated in a mathematically rigorous way; this is usually done by employing dimensional regularization in which the number of space-time dimensions are set to $D = 4 - 2\epsilon$, i.e. away from $D = 4$. This way the divergences appear as inverse powers of ϵ , in an NLO calculation $\frac{1}{\epsilon}$ and $\frac{1}{\epsilon^2}$. There are other regularization prescriptions but dimensional regularization is a common choice because it preserves many symmetries such as gauge in variance.

Dimensional regularization, however, requires us to perform all possibly divergent calculations in D dimensions with D , in general, not being an integer. This prevents us to implement the calculations *directly* in numerical programs which are desirable because of their efficiency when performing the phase-space integration. A conservative way to work around this problem is to perform the calculations analytically in D dimensions, check that the divergences cancel between the virtual and real correction and then finally transform the result into numerical code working in $D = 4$. The downside of this approach is that the resulting codes may become very large for a high number of final states and that for every new process the codes have to be newly generated. A more desirable way would be a method that allows for a “numerical regularization”, i.e. the numerical computation of the finite parts in $D = 4$ and the extraction of the divergences in D dimensions, preferably in a process-independent way.

There are at least three ways to achieve this:

1. Rewriting the integrals in such a way that loop-integral and the integral over the additional particle are performed together, or
2. slicing the phase-space, i.e. treating the parts of phase-space separately that gives rise to the divergences, or
3. using the subtraction method that subtracts out the parts that cause the divergence and adds it back in the loop integral where the divergences cancel.

This work will concentrate on the subtraction method which in turn has found at least three different realizations in literature; the CS-subtraction[29] named after their authors Catani and Seymour, the FKS-subtraction[30, 31] from Frixione, Kunszt and Signer and the Antenna-Subtraction[32]. These methods make use of a universal behavior of the amplitudes when one particle gets soft and/or collinear to another.

The remaining parts of this section is organized as follows: We will first discuss why and where divergences arise in phase-space integrals of the real correction

(Sec. 1.5.2), introduce the idea of the subtraction method in Sec. 1.5.3, and then derive our own subtraction terms in Sec. 1.5.4. These terms are formulated independently of the polarizations — our aim is to use them for the random polarizations that we introduced in Sec. 1.3. This is motivated by the fact that at LO random polarizations offer a speedup (see Sec. 1.3.2) which we would like to have for the radiative corrections as well. We will test this in Sec. 1.5.8 where we apply our subtraction terms to the calculation of n -Jet cross sections.

Additionally, we show that for helicity eigenstates the new terms are basically the same as the ones known from the CS-subtraction method (Sec. 1.5.5) and show which parts are needed in addition for the random polarized spinors/vectors. In Sec. 1.5.7 we will check the local behavior of the new terms.

Please also note that the following section assume we are working with partial amplitudes.

1.5.2 The Origin of Soft- and Collinear Divergences

Let us first illustrate which kind of divergences appear in the real correction and how they arise. Obviously, the squared amplitude $|\mathcal{A}_{n+1}|^2$ is always finite as long as all invariants $s_{ij} = 2p_i \cdot p_j$ that appear inversely in propagators are different from zero. The divergences arise when we perform the phase-space integration where the phase space has points close to these regions $s_{ij} = 0$. The phase-space integration has the form

$$\prod_{i=1}^n \int \frac{d^3 \vec{p}_i}{E(\vec{p}_i)} \delta^{(4)} \left(\sum_{j=1}^m q_j - \sum_{i=1}^n p_i \right) |\mathcal{A}_n(\vec{q}_1, \dots, \vec{q}_m; \vec{p}_1, \dots, \vec{p}_n)|^2 \quad (1.60)$$

with q_j the incoming momenta that are constraint⁷ by the experiment, and p_i the outgoing momenta that are kinematically constrained by the on-shell condition $p_i^2 = m_i^2$ and momentum conservation enforced by the delta-distribution. For decays of particles we have $m = 1$ but at colliders $m = 2$. We will call a set of n momenta

$$\{p_1, p_2, \dots, p_n\} \quad (1.61)$$

fulfilling these conditions a *phase-space point*. All phase-space points comprise the phase space. In this section we will identify the regions of phase-space that give rise to divergences.

⁷The incoming momenta are determined when the colliding particles are elementary one, e.g. electrons and positrons at LEP; when the particle is a composite one such like the protons that are collided at the LHC, the elementary particles that make up the protons are not fully determined.

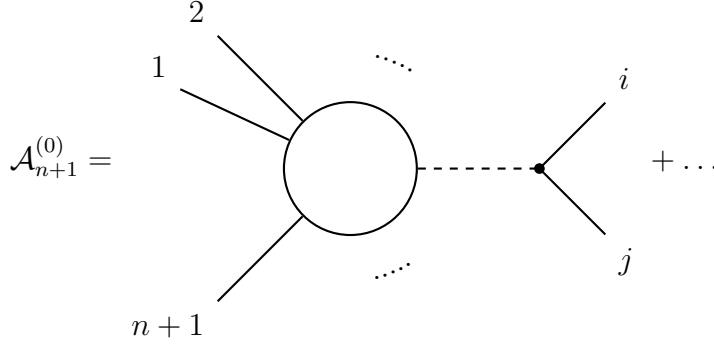


Figure 1.3: *Topology of diagrams that are possibly divergent if particles i and j are collinear. The divergent part is the propagator drawn with a dashed line. The singular part can be extracted from the three-valent vertex and the legs of particles i and j . The blob contains the same set of diagrams as a matrix element with n particles where particles i and j are replaced with a particle $i + j$. The soft divergences of a particle with soft momentum j can be obtained by summing over $j \neq i$ since j produces divergences with every external particle it couples.*

Let us now set $n \rightarrow n + 1$ and look at the structure of the Feynman diagrams, focussing on pairs of two external particles i and j . We divide the diagrams in two classes; the first class contains particles i and j that are directly connected by a three-valent vertex and a corresponding propagator (see Fig. 1.3), the second class simply contains the remaining diagrams and are denoted by the dots in Fig 1.3. Note that in general both set of diagrams are not separately gauge invariant. We will write the contribution of these two classes as

$$\sum_{i \neq j} \frac{A_{ij}}{(p_i + p_j)^2 - m_{ij}^2} + B \quad (1.62)$$

where the propagator of the first class is made explicit and factored off the rest, A_{ij} , and the contribution of the second class is simply written as B . If we square the whole amplitude and perform the phase space integration over one particle, say particle j , this looks like

$$\int \frac{d^{D-1} \vec{p}_j}{E(\vec{p}_j)} \delta^{(D)}(K_j - p_j) \left| \sum_{i \neq j} \frac{A_{ij}}{(p_i + p_j)^2 - m_{ij}^2} + B \right|^2 \quad (1.63)$$

where we have chosen to dimensionally regularize the phase space integral by setting the integration dimension from 3 to $D - 1 = 3 - 2\epsilon$ and abbreviated

$K_j = \sum_i q_i - \sum_{i \neq j} p_i$. Performing the square gives

$$\int \frac{d^{D-1} \vec{p}_j}{E(\vec{p}_j)} \delta^{(D)}(K_j - p_j) \left[\sum_{i \neq j} \sum_{k \neq j} \frac{A_{ij}^* A_{kj}}{[(p_i + p_j)^2 - m_{ij}^2][(p_k + p_j)^2 - m_{kj}^2]} + \sum_{i \neq j} \frac{2\Re(A_{ij}^* B)}{(p_i + p_j)^2 - m_{ij}^2} + |B|^2 \right]. \quad (1.64)$$

Note that A_{ij} and B depend on all momenta. The measure is rewritten using D -dimensional spherical coordinates[33]

$$d^{D-1} \vec{p}_j = p^{D-2} d\lambda \sin^{D-4} \theta d\cos\theta d\Phi \quad (1.65)$$

with the energy $\lambda = E(\vec{p}_j) \in [0, \infty)$, $\cos\theta \in [-1, +1]$ and Φ being the remaining coordinates that we are not interested in. The delta distribution gives an additional restriction on the upper bound of λ which we will neglect because the divergences arise from regions near $\lambda = 0$. The propagator's denominator is rewritten to

$$m_i^2 + m_j^2 + 2p_i \cdot p_j - m_{ij}^2 \quad (1.66)$$

which simplifies, if all masses are zero, to

$$2p_i \cdot p_j = 2\lambda |\vec{p}_i| (1 - \cos\theta_{ij}). \quad (1.67)$$

Inserting everything back into the original expression we arrive with

$$\int d\lambda \lambda^{D-3} d\cos\theta d\Phi \left[\sum_{i \neq j} \sum_{k \neq j} \frac{A_{ij}^* A_{kj}}{4\lambda^2 [|\vec{p}_i|(1 - \cos\theta_{ij})][|\vec{p}_k|(1 - \cos\theta_{ik})]} + \sum_{i \neq j} \frac{2\Re(A_{ij}^* B)}{2\lambda |\vec{p}_i|(1 - \cos\theta_{ij})} + |B|^2 \right] \quad (1.68)$$

from which we can identify the phase space regions that lead to divergences. The last term is trivially finite because it does not lead to singular behavior — B is finite by definition. The integral

$$\int d\lambda \lambda^{D-4} = \frac{\lambda^{D-3}}{D-3} = \frac{\lambda^{1-2\epsilon}}{1-2\epsilon} \quad (1.69)$$

of the second term is finite at the lower bound $\lambda = 0$, but the first term

$$\int d\lambda \lambda^{D-5} = \frac{\lambda^{D-4}}{D-4} = -\frac{\lambda^{-2\epsilon}}{2\epsilon} \quad (1.70)$$

is logarithmically divergent in $D = 4$. These divergences are called soft divergences because they occur for small energies λ .

The second type of divergence arises from the integration over the region where $\cos\theta \approx 1$ in the diagrams where $i = k$, i.e. when the particles i and j are collinear. To investigate on this we first rewrite p_i and p_j with the *Sudakov parametrization*,

$$p_i^\mu = zp^\mu + k^\mu - \frac{k^2}{z} \frac{n^\mu}{2p \cdot n} \quad (1.71a)$$

$$p_j^\mu = (1-z)p^\mu - k^\mu - \frac{k^2}{1-z} \frac{n^\mu}{2p \cdot n}, \quad (1.71b)$$

with

$$p \cdot n \neq 0, \quad k \cdot n = k \cdot p = 0, \quad p^2 = n^2 = 0, \quad k^2 \neq 0. \quad (1.72)$$

The momentum p points in the collinear direction, k^2 is a measure for a-collinearity since

$$(p_i + p_j)^\mu = p^\mu - \frac{k^2}{z(1-z)} \frac{n^\mu}{2p \cdot n} \xrightarrow{k^2 \rightarrow 0} p^\mu \quad (1.73)$$

and the propagator

$$\frac{1}{(p_i + p_j)^2} = -\frac{z(1-z)}{k^2} \quad (1.74)$$

diverges for $k^2 \rightarrow 0$. The vector n is an auxiliary one since we need three directions. We can then show (see App. A.1.1) that

$$\frac{d^3\vec{p}_j}{2E_j} = \frac{k}{1-z} dz dk d\phi + \mathcal{O}(k^2) \quad (1.75)$$

which makes clear that every integrand that falls off faster than $\frac{1}{k}$ causes the integral to diverge. By calculating the splitting kernels[34], i.e. the Feynman diagrams for the decay $(ij) \rightarrow i + j$, it can be shown that $A \sim k$ so that again the mixing term is constant and a logarithmic divergence is produced by

$$\int dk k \frac{z^2(1-z)^2}{k^4} |A|^2 \sim \int \frac{dk}{k}. \quad (1.76)$$

1.5.3 The Subtraction Method

Let us write down the NLO correction in Eq. (1.56) once again:

$$\begin{aligned} \langle \mathcal{O} \rangle_{\text{Real+Virtual}}^{\text{NLO}} &= \int_{n+1} d\phi_{n+1} J_n^{(1)} |\mathcal{A}_{n+1}^{(0)}|^2 \mathcal{O}_{n+1} + \\ &\int_n d\phi_n J_n^{(0)} 2\Re(\mathcal{A}_n^{(0)*} \mathcal{A}_n^{(1)}) \mathcal{O}_n. \end{aligned} \quad (1.77)$$

As argued above, the loop-integral in the virtual correction produces soft- and collinear divergences that are canceled by the real correction. Unfortunately, the

cancellation happens *after* performing the two different integrals, which makes it impossible to evaluate the integrals numerically.

The idea of the subtraction method is to construct an auxiliary term that is subtracted from the real correction such that it renders the integration finite. In terms of Eq. (1.68) these subtraction terms approximate the divergent parts proportional to $|A|^2$. To correct for the subtracted part, it has to be added back in the virtual correction and thereby rendering this part finite as well.

To construct the auxiliary term(s) one has to know where in phase space the divergences arise. As we have explained in Sec. 1.5.2 these are the collinear and soft configurations. Looking at Eq. (1.68) again we already see that the divergent parts of $|\mathcal{A}_{n+1}^{(0)}|^2$ factorizes into a finite matrix element $|\mathcal{A}_n^{(0)}|^2$ with one particle less times a divergent quantity that is simple enough to work with. This is enough to construct the auxiliary term which in the case of Catani-Seymour subtraction is written as

$$J_n^{(0)} \sum_{k \neq i, j} \mathcal{D}_{ij, k} \mathcal{O}_n \quad (1.78)$$

where the Catani-Seymour *Dipoles* $\mathcal{D}_{ij, k}$ contain the n -particle squared matrix elements. This auxiliary term approximates the divergent parts of $J_n^{(1)} |\mathcal{A}_{n+1}^{(0)}|^2 \mathcal{O}_{n+1}$ in the critical phase-space regions so that

$$\int_{n+1} d\phi_{n+1} \left[J_n^{(1)} |\mathcal{A}_{n+1}^{(0)}|^2 \mathcal{O}_{n+1} - J_n^{(0)} \sum_{k \neq i, j} \mathcal{D}_{ij, k} \mathcal{O}_n \right] \quad (1.79)$$

is finite if the observable \mathcal{O}_{n+1} is infrared-safe, i.e. if $\mathcal{O}_{n+1} \rightarrow \mathcal{O}_n$ when two momenta become collinear and/or one momentum soft. The jet-functions in this regions also have to fulfill this property: $J_n^{(1)} \rightarrow J_n^{(0)}$. Furthermore, the dipoles are simple enough so they can be (once and for all) integrated over the additional one-particle phase-space, which regularizes the soft- and collinear divergences of the virtual part:

$$\int_n d\phi_n J_n^{(0)} \left[2\Re \left(\mathcal{A}_n^{(0)*} \mathcal{A}_n^{(1)} \right) + \int_1 \sum_{k \neq i, j} \mathcal{D}_{ij, k} \right]_{\epsilon=0} \mathcal{O}_n = \text{finite}. \quad (1.80)$$

The integration of the dipoles yield the soft- and collinear divergence in the form of $\frac{1}{\epsilon}$ and $\frac{1}{\epsilon^2}$ poles in dimensional regularization with $D = 4 - 2\epsilon$ dimensions. The poles cancel inside the square bracket so the integrand is finite and integration also yields a finite result.

Our task in the following will be to extend the CS-subtraction method for the random polarizations introduced in Sec. 1.3.

1.5.4 The Subtraction Terms

Parts of the following were already published in Ref. [35]. In this thesis we additionally apply it to the Catani-Seymour dipole subtraction and use it to compute radiative corrections of jet production at LEP.

We have to keep in mind that our motivation for deriving new subtraction terms is to apply them to random polarizations. To do so, we will derive the limiting behavior of QCD amplitudes for soft gluons and collinear particles without making use of the helicity-sum identity that does not hold true for random polarizations.

1.5.4.1 The Soft Limit

We will first discuss the case of soft divergences. In Sec. 1.5.2 we showed that soft gluons with momenta $p_j = \lambda q$, $\lambda \rightarrow 0$ cause the phase-space integral to diverge. In particular, the part of the matrix element that is proportional to λ^{-2} is divergent enough; this is the part we need to subtract. In the following we will therefore derive the limiting behavior of an amplitude with a fixed gluon j being soft: $\lim_{p_j \rightarrow 0} \mathcal{A}_{n+1}^{(0)}$.

The diagrams that we have to consider are the ones that couple the soft gluon j to another gluon i or an (anti-)quark i so the diagrams contain a propagator of the type

$$\frac{1}{(p_i + p_j)^2 - m_i^2} = \frac{1}{2p_i \cdot p_j} = \frac{\lambda^{-1}}{2q \cdot p_i}. \quad (1.81)$$

In the remaining diagrams the soft gluon couples either via a four-gluon vertex or deeper inside the diagram so the corresponding propagator contains more momenta, e.g. three

$$\frac{1}{(p_i + p_j + p_l)^2} = \frac{1}{2\lambda q \cdot (p_i + p_l) + 2p_i \cdot p_l} \longrightarrow \frac{1}{2p_i \cdot p_l} \quad (1.82)$$

which are constant in the soft limit $\lambda \rightarrow 0$. We can therefore split the amplitude in the following way:

$$\mathcal{A}_{n+1}^{(0)} = \sum_{i \neq j}^{n+1} \mathcal{A}'_{\xi} \frac{E_{ij}^{\xi}}{2p_i \cdot p_j} + \mathcal{A}'' \quad (1.83)$$

where the sum represents all diagrams with a divergent propagator and \mathcal{A}'' diagrams with propagators that are constant in the soft limit. Note that the \mathcal{A}'_{ξ} still depends on p_j but only via propagators of the type shown in Eq. (1.82) and three-gluon vertices so that $\lim_{\lambda \rightarrow 0} \mathcal{A}'_{\xi} = \text{const.}$

In the case where particle i is a gluon the divergent diagrams with fixed i are

$$\mathcal{A}'_{\mu} \frac{-ig_{\nu}^{\mu}}{2p_i \cdot p_j} V_3^{\nu\rho\sigma} \varepsilon_{\rho}^*(p_i) \varepsilon_{\sigma}^*(p_j) \quad (1.84)$$

with V_3 the three-gluon vertex as given in Eq. (A.75d). The other case in which the soft gluon couples to a quark with mass m_i (which cancels in the propagator) is

$$\bar{u}_\alpha(p_i) V_{\alpha\beta}^\nu \varepsilon_\nu^*(p_j) \frac{i \left[(p_i + p_j)_\mu \gamma_{\beta\gamma}^\mu + m_i \delta_{\beta\gamma} \right]}{2p_i \cdot p_j} \mathcal{A}'_\gamma. \quad (1.85)$$

In the first case the nominator evaluates (coupling constants suppressed) to

$$\begin{aligned} \mathcal{A}'^\mu i f^{abc} \left((-2p_i + p_j) \cdot \varepsilon^*(p_j) \varepsilon_\mu^*(p_i) + (p_i - p_j)^\mu \varepsilon^*(p_j) \cdot \varepsilon^*(p_i) \right. \\ \left. + (2p_j + p_i) \cdot \varepsilon^*(p_i) \varepsilon_\mu^*(p_j) \right) \xrightarrow{\lambda \rightarrow 0} \mathbf{T}_i \mathcal{A}' \cdot \varepsilon^*(p_i) 2p_i \cdot \varepsilon^*(p_j). \end{aligned} \quad (1.86)$$

Terms proportional to p_j vanish in the soft limit and are constant with the divergent propagator. The term proportional to p_i vanishes as well, because the expression \mathcal{A}'^μ contains the same diagrams as an n -particle amplitude (with polarization vector of gluon i removed). The corresponding momenta, however, are $\{p_i\}_{i \neq j}^{n+1}$ which are conserved in the soft limit:

$$\sum_{i \neq j}^{n+1} p_i = -p_j \xrightarrow{p_j \rightarrow 0} 0. \quad (1.87)$$

Therefore the expression $\mathcal{A}' \cdot p_i$ vanishes because it is the statement of gauge invariance. Note that we have defined $\mathbf{T}_i = -i f^{abc}$. The derivation here is completely general with respect to the polarization vectors, the only property we used is the transversality condition $p \cdot \varepsilon(p) = 0$ which holds true both for helicity eigenstates and random polarizations.

In the case of i being a quark we have with $\mathbf{T}_i = -T_{ij}^a$

$$\begin{aligned} \bar{u}_\alpha(p_i) i \gamma_{\alpha\beta}^\nu T_{ij}^a \varepsilon_\nu^*(p_j) i \left((p_i + p_j)_\mu \gamma_{\beta\gamma}^\mu + m_i \delta_{\beta\gamma} \right) \mathcal{A}'_\gamma \\ \xrightarrow{p_j \rightarrow 0} \bar{u}_\alpha(p_i) \gamma_{\alpha\beta}^\nu \varepsilon_\nu^*(p_j) \left(\sum_{\lambda=\pm} u_\beta^\lambda(p_i) \bar{u}_\beta^\lambda(p_i) \right) \mathbf{T}_i \mathcal{A}'_\gamma \\ = \sum_{\lambda=\pm} \left(\bar{u}_\alpha(p_i) \gamma_{\alpha\beta}^\nu \varepsilon_\nu^*(p_j) u_\beta^\lambda(p_i) \right) \mathbf{T}_i \left(\bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma \right). \end{aligned} \quad (1.88)$$

At this point we have to distinguish between quarks in helicity eigenstates with $\lambda_i = \pm$ and random polarized quarks with $\phi_i \in [0, 2\pi)$. For both cases, however, we obtain (see App. A.1.2):

$$2p_i \cdot \varepsilon_\nu^*(p_j) \mathbf{T}_i \bar{u}_\alpha(p_i) \mathcal{A}'_\alpha. \quad (1.89)$$

For the derivation of the case of an antiquark one proceeds analogously.

Eqs. (1.88) and (1.86) can be summarized in a single formula, because in the soft limit

$$\begin{aligned} \mathcal{A}' \cdot \varepsilon^*(p_i) &\xrightarrow{p_j \rightarrow 0} \mathcal{A}_n^{(0)} \\ \bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma &\xrightarrow{p_j \rightarrow 0} \mathcal{A}_n^{(0)}. \end{aligned} \quad (1.90)$$

This is true because, as argued above already, in the soft limit the momenta $\{p_i\}_{i \neq j}^{n+1}$ sum to zero and \mathcal{A}' contain the same diagrams as $\mathcal{A}_n^{(0)}$. Plugging our results back into Eq. (1.83) gives

$$\lim_{p_j \rightarrow 0} \mathcal{A}_{n+1}^{(0)} \sim \varepsilon_\mu^*(p_j) J^\mu \mathcal{A}_n^{(0)}, \quad J^\mu = \sum_{i=1}^n \mathbf{T}_i \frac{p_i^\mu}{p_i \cdot p_j} \quad (1.91)$$

with the *eikonal current* J^μ that factorizes the soft momentum from the amplitude. Here we assume that the particle indices have been renamed so the particle j is excluded.

Squaring the matrix element gives the soft limit that diverges as λ^{-2} :

$$\lim_{p_j \rightarrow 0} \left| \mathcal{A}_{n+1}^{(0)} \right|^2 \sim \sum_{i=1}^n \sum_{k=1}^n \mathcal{A}_n^{(0)*} \mathbf{T}_i \cdot \mathbf{T}_k \mathcal{A}_n^{(0)} S_{ijk}(\varepsilon_j) \quad (1.92)$$

where we defined the soft function

$$S_{ijk}(\varepsilon_j) = (\varepsilon(p_j) \cdot J)(\varepsilon^*(p_j) \cdot J) = \sum_{i=1}^n \sum_{k=1}^n \mathbf{T}_i \cdot \mathbf{T}_k \frac{p_i \cdot \varepsilon(p_j)}{p_i \cdot p_j} \frac{p_k \cdot \varepsilon^*(p_j)}{p_k \cdot p_j}. \quad (1.93)$$

Using color conservation we re-arrange this expression (proof given in App. A.1.3) and write

$$\lim_{p_j \rightarrow 0} \left| \mathcal{A}_{n+1}^{(0)} \right|^2 \sim - \sum_{i=1}^n \sum_{k \neq i}^n \mathcal{A}_n^{(0)*} \mathbf{T}_i \cdot \mathbf{T}_k \mathcal{A}_n^{(0)} S_{ij,k}(\varepsilon_j) \quad (1.94)$$

with the new soft function

$$S_{ij,k}(\varepsilon_j) = \frac{(p_i \cdot \varepsilon_j^*)(p_i \cdot \varepsilon_j)}{(p_i \cdot p_j)^2} - \frac{2\Re \left((p_i \cdot \varepsilon_j^*)(p_k \cdot \varepsilon_j) \right)}{(p_i \cdot p_j)(p_i \cdot p_j + p_j \cdot p_k)}. \quad (1.95)$$

which has sums similar to the Catani-Seymour dipoles.

1.5.4.2 The Collinear Limit

Collinear divergences arise in phase-space regions where two momenta are collinear, i.e. when the angle between them is small. In analogy to Eq. (1.83) we can write the amplitude with collinear momenta i and j as

$$\mathcal{A}_{n+1}^{(0)} = \mathcal{A}'_\xi \frac{E_{ij}^\xi}{2p_i \cdot p_j} + \mathcal{A}'' \quad (1.96)$$

Note that in contrast to Eq. (1.83) there is no sum because collinearity involves two particles, being soft only one. The first term denotes the Feynman diagrams that are the divergent ones and the remaining, denoted by \mathcal{A}'' , are constant in the collinear limit.

To arrive with a factorized formula we rewrite the propagator's tensor/spinor structure with

$$g^{\mu\nu} = \frac{p_{ij}^\mu q_{ij}^\nu + q_{ij}^\mu p_{ij}^\nu}{p_{ij} \cdot q_{ij}} - \sum_{\lambda=\pm} \varepsilon_\lambda^\mu(p_{ij}) \varepsilon_{\lambda'}^{\nu*}(p_{ij}) \quad (1.97)$$

if ξ is Lorentz index or

$$(p_i + p_j)^\mu \gamma_\mu \xrightarrow{p_i \parallel p_j} \not{p}_{ij} = \sum_{\lambda=\pm} u_\lambda(p_{ij}) \bar{u}_\lambda(p_{ij}) \quad (1.98)$$

if ξ is a spinor index so that we obtain, similar to the soft case, the n -particle amplitude $\mathcal{A}_n^{(0)}$ and the splitting functions $\text{Split}_{(ij) \rightarrow i+j}$ which is the Feynman diagram for the process $(ij) \rightarrow i+j$. This gives

$$\lim_{p_i \parallel p_j} \mathcal{A}_{n+1}^{(0)}(\dots, p_i, h_i, \dots, p_j, h_j, \dots) \sim \sum_{\lambda=\pm} \text{Split}_{(ij) \rightarrow i+j}(p_{ij}, \lambda, p_i, h_i, p_j, h_j) \times \mathbf{T}_{(ij) \rightarrow i+j} \mathcal{A}_n^{(0)}(\dots, p_{ij}, \lambda, \dots) \quad (1.99)$$

with splitting functions

$$\text{Split}_{q_{ij} \rightarrow q_i q_j} = \frac{1}{(p_i + p_j)^2 - m_{ij}^2} \bar{u}_i \not{p}_j u_{ij}, \quad (1.100a)$$

$$\text{Split}_{g_{ij} \rightarrow g_i g_j} = \frac{2}{2p_i \cdot p_j} \left(\varepsilon_i \cdot \varepsilon_j p_i \cdot \varepsilon_{ij}^* + \varepsilon_j \cdot \varepsilon_{ij}^* p_j \cdot \varepsilon_i - \varepsilon_i \cdot \varepsilon_{ij}^* p_i \cdot \varepsilon_j \right), \quad (1.100b)$$

$$\text{Split}_{g_{ij} \rightarrow q_i \bar{q}_j} = \frac{1}{2p_i \cdot p_j} \bar{u}_i \not{p}_j v_j, \quad (1.100c)$$

where the indices denote the corresponding momenta and helicity pairs, i.e. $i \hat{=} (p_i, h_i)$, $j \hat{=} (p_j, h_j)$, and $ij \hat{=} (p_{ij}, \lambda)$. The proof is given in App. A.1.5.

Squaring the collinear limit we obtain

$$\lim_{p_i \parallel p_j} \left| \mathcal{A}_{n+1}^{(0)}(\dots, p_i, h_i, \dots, p_j, h_j, \dots) \right|^2 = \left[\mathcal{A}_n^{(0)}(\dots, p_{ij}, \lambda, \dots) \right]_\xi^* \mathbf{T}_{(ij) \rightarrow i+j}^2 \left[P_{(ij) \rightarrow i+j}(\lambda, \lambda') \right]_{\xi\xi'} \left[\mathcal{A}_n^{(0)}(\dots, p_{ij}, \lambda', \dots) \right]_{\xi'} \quad (1.101)$$

with the squares of the splitting functions

$$\left[P_{(ij) \rightarrow i+j}(\lambda, \lambda') \right]_{\alpha\beta} = \sum_{\lambda, \lambda'} u_\alpha^\lambda(p_{ij}) \text{Split}_{(ij) \rightarrow i+j}^* \text{Split}_{(ij) \rightarrow i+j} \bar{u}_\beta^{\lambda'}(p_{ij}), \quad (1.102a)$$

$$\left[P_{(ij) \rightarrow i+j}(\lambda, \lambda') \right]_{\mu\nu} = \sum_{\lambda, \lambda'} \varepsilon_\mu^{\lambda*}(p_{ij}) \text{Split}_{(ij) \rightarrow i+j}^* \text{Split}_{(ij) \rightarrow i+j} \varepsilon_\nu^{\lambda'}(p_{ij}). \quad (1.102b)$$

Note that the squaring canceled the color correlation which remain in the soft case. Instead we obtain a spin correlation because the amplitudes still depend on λ, λ' , the spin of the intermediate particle (ij) .

1.5.4.3 Choice of Momenta

Before we proceed it is convenient to rewrite the soft limit such that it more resembles the collinear case. In particular, to achieve that we have to unify the dependence of the amplitudes on the momenta which is different in both cases. Whereas in the soft limit the amplitudes

$$A(p_1, p_2, \dots, p_{j-1}, p_{j+1}, \dots, p_n) \quad (1.103)$$

are functions of all momenta except the soft one, p_j , in the collinear limit the amplitudes

$$A(p_1, p_2, \dots, p_{i-1}, p_{ij}, p_{i+1}, \dots, p_{j-1}, p_{j+1}, \dots, p_n) \quad (1.104)$$

depend on a new momentum p_{ij} that replaces the collinear momenta p_i and p_j . We have not yet defined p_{ij} and only require that in the collinear limit (for massless particles) it is $p_{ij} = p_i + p_j$ which is also lightlike,

$$(p_i + p_j)^2 = 2p_i \cdot p_j = |\vec{p}_i| |\vec{p}_j| (1 - \cos \theta) = 0, \quad (1.105)$$

because the collinear limit is defined by $\theta = 0$.

We can unify this by using the momenta defined in Ref. [29],

$$\tilde{p}_{ij} = p_i + p_j - \frac{y}{1-y} p_k \quad (1.106a)$$

$$\tilde{p}_k = \frac{1}{1-y} p_k \quad (1.106b)$$

$$y_{ij,k} = \frac{p_i \cdot p_j}{p_i \cdot p_j + p_i \cdot p_k + p_j \cdot p_k}, \quad (1.106c)$$

so that p_i, p_j, p_k are replaced by the $\tilde{p}_{ij}, \tilde{p}_k$. One can show (see App. A.1.6) that in the soft case

$$\tilde{p}_k \rightarrow p_k \quad (1.107a)$$

$$\tilde{p}_{ij} \rightarrow p_i \quad (1.107b)$$

and in the collinear case

$$\tilde{p}_k \rightarrow p_k \quad (1.108a)$$

$$\tilde{p}_{ij} \rightarrow p_i + p_j. \quad (1.108b)$$

Invariants involving momenta p_i, p_j, p_k can be rewritten using the identities listed in Eq. (1.121).

Note that $p_i + p_j + p_k = \tilde{p}_k + \tilde{p}_{ij}$ and $\tilde{p}_{ij}^2 = \tilde{p}_k^2 = 0$, i.e. even away from the soft- and collinear limit the momenta of the n -particle amplitudes are conserved and lightlike — this is important, because it allows us to smoothly approach the limits, i.e. in the neighborhood of these points in phase space.

1.5.4.4 Spin-Correlation for the Soft Function

Replacing the “untilded” momenta in the collinear case is straightforward, in the soft case however we still have the momentum p_i — if we write the soft case in analogy to Eq. (1.102) we obtain

$$[S_{ij,k}]_{\alpha\beta} = u_\alpha(p_i, h_i) S_{i \rightarrow ij} \bar{u}_\beta(p_i, h_i) \quad (1.109a)$$

$$[S_{ij,k}]_{\mu\nu} = \varepsilon_\mu^*(p_i, h_i) S_{i \rightarrow ij} \varepsilon_\nu(p_i, h_i) \quad (1.109b)$$

where instead of p_i we would like to have \tilde{p}_{ij} . This is done by introducing a soft spin-correlation $S_{\lambda\lambda'}$, i.e. matrices that satisfy

$$\sum_{\lambda=\pm} \sum_{\lambda'=\pm} u_\lambda(p_{ij}) S_{\lambda\lambda'} \bar{u}_{\lambda'}(p_{ij}) \xrightarrow{p_{ij} \rightarrow p_i} u_{h_i}(p_i) \bar{u}_{h_i}(p_i), \quad (1.110a)$$

$$\sum_{\lambda=\pm} \sum_{\lambda'=\pm} \varepsilon_{\lambda'}^*(p_{ij}) S_{\lambda\lambda'} \varepsilon_\lambda(p_{ij}) \xrightarrow{p_{ij} \rightarrow p_i} \varepsilon_{h_i}^*(p_i) \varepsilon_{h_i}(p_i). \quad (1.110b)$$

These matrices are given by

$$S_{\lambda\lambda'} = \frac{1}{2p_i \cdot q_{ij}} \frac{1}{2p_{ij} \cdot q_{ij}} \bar{u}_\lambda(p_{ij}) \not{q}_{ij} u_{h_i}(p_i) \bar{u}_{h_i}(p_i) \not{q}_{ij} u_{\lambda'}(p_{ij}), \quad (1.111a)$$

$$S_{\lambda\lambda'} = \varepsilon_\lambda(p_{ij}) \cdot \varepsilon^*(p_i) \varepsilon(p_i) \cdot \varepsilon^*(p_{ij}). \quad (1.111b)$$

For a proof see App. A.1.7.

1.5.4.5 Eliminating Double-Counting of Soft-Collinear Divergences

So far we discussed the soft and the collinear case separately. To construct subtraction terms we have to take into account that these limiting cases can and do overlap since there are phase space points with a soft momentum that is collinear to another momentum at the same time. The limiting cases therefore double-count these which has to be corrected for. This will be done in the following by evaluating the collinear limit of the soft term and soft limit of the collinear terms. We will see that we can simply drop the first term in $S_{ij,k}(\varepsilon_j)$ and therefore avoid double counting.

The collinear limit of the soft term is

$$\lim_{p_i \parallel p_j} S_{ij,k}(\varepsilon_j) \sim \frac{(p_i \cdot \varepsilon_j^*)(p_i \cdot \varepsilon_j)}{(p_i \cdot p_j)^2} \quad (1.112)$$

because the second is not singular enough. The soft limits of the collinear functions are

$$\lim_{p_j \rightarrow 0} [P_{q_{ij} \rightarrow q_i g_j}(\lambda, \lambda')]_{\alpha\beta} \sim \frac{(p_i \cdot \varepsilon_j^*)(p_i \cdot \varepsilon_j)}{(p_i \cdot p_j)^2} u_\alpha(p_i) \bar{u}_\beta(p_i) \quad (1.113a)$$

$$\lim_{p_j \rightarrow 0} [P_{g_{ij} \rightarrow g_i g_j}(\lambda, \lambda')]_{\mu\nu} \sim \frac{(p_i \cdot \varepsilon_j^*)(p_i \cdot \varepsilon_j)}{(p_i \cdot p_j)^2} \varepsilon_\mu^*(p_i) \varepsilon_\nu(p_i) \quad (1.113b)$$

and the collinear function for the remaining splitting is not singular enough. We see that the limits agree. To avoid double-counting we construct a soft function S that differs from $S_{ij,k}(\varepsilon_j)$ by simply leaving out the first term:

$$S(\varepsilon_j) = -\frac{2\Re\left((p_i \cdot \varepsilon_j^*)(p_k \cdot \varepsilon_j)\right)}{(p_i \cdot p_j)(p_i \cdot p_j + p_j \cdot p_k)} \quad (1.114)$$

1.5.4.6 The \mathcal{R} -Operator: Randomly Polarized Dipoles

Using the previously derived subtraction terms with random polarizations is sufficient to obtain a finite integral for real corrections, but the value will differ from the one computed with helicity eigenstates and CS dipoles. The reason for this is that the subtraction terms may, in general, differ by a finite amount that can be shifted between the real- and virtual correction. One possibility is to re-derive also the integrated subtraction terms, but this poses additional restrictions on the terms that are difficult to meet.

Instead, we will construct our new terms in such a way that new integrated terms are not needed and the real corrections with either type of polarizations yield the same result. This we can achieve by making use of Eq. (1.29): The first term is the helicity sum for which we already have the Catani-Seymour subtraction terms and the remaining terms depending on the helicity angles are new and have to be accounted for. Since these terms integrate to zero, we can modify Eq. (1.79) by replacing the CS-dipole $\mathcal{D}_{ij,k}$ with $\mathcal{D}_{ij,k} + \mathcal{D}'_{ij,k}(\theta_i, \theta_j)$ so that it reads

$$\int d\Theta \int_{n+1} d\phi_{n+1} \left[J_n^{(1)} |\mathcal{A}_{n+1}^{(0)}|^2 \mathcal{O}_{n+1} - J_n^{(0)} \sum_{k \neq i,j} (\mathcal{D}_{ij,k} + \mathcal{D}'_{ij,k}) \mathcal{O}_n \right] \quad (1.115)$$

with the integration over the helicity angles

$$\int d\Theta \equiv \frac{1}{(2\pi)^{n+3}} \int_0^{2\pi} d\theta_1 \cdots \int_0^{2\pi} d\theta_{n+3} \quad (1.116)$$

made explicit. The subtraction term $\mathcal{D}_{ij,k}$ cancels the divergent parts that do not depend on helicity angles θ and $\mathcal{D}'_{ij,k}(\theta_i, \theta_j)$ subtracts the parts that depend on the helicity-angles.

The new dipoles can be derived from our subtraction terms if we apply the \mathcal{R} -operator, which simply extracts the helicity-angle dependent parts:

$$\mathcal{R}[f(\theta_i, \theta_j)] = f(\theta_i, \theta_j) - \frac{1}{4} \sum_{\theta_i \in \{0, \pi\}} \sum_{\theta_j \in \{0, \pi\}} f(\theta_i, \theta_j). \quad (1.117)$$

The last term of the \mathcal{R} -operator is according to Sec. 1.4.4 just the sum over the helicities λ_i and λ_j , so we are left with terms that depend on θ_i and/or θ_j .

1.5.5 The Catani-Seymour Subtraction

The Catani-Seymour subtraction terms, or rather their leading-color form, are reproduced if one sums the terms S , and P over helicity eigenstates:

$$\sum_{\lambda_i=\pm} \sum_{\lambda_j=\pm} (P + S). \quad (1.118)$$

For this we introduce new variables[29] P (not to be confused with the splitting function P), z , and y (we drop the indices i, j, k which are made explicit in Ref. [29]):

$$P^2 = s_{ij} + s_{jk} + s_{ik} \quad (1.119a)$$

$$y = \frac{s_{ij}}{s_{ij} + s_{jk} + s_{ik}} \quad (1.119b)$$

$$z = \frac{s_{ik}}{s_{jk} + s_{ik}}. \quad (1.119c)$$

which depend on the invariants

$$s_{ij} = 2p_i \cdot p_j \quad (1.120a)$$

$$s_{ik} = 2p_i \cdot p_k \quad (1.120b)$$

$$s_{jk} = 2p_j \cdot p_k. \quad (1.120c)$$

To rewrite the expression it is convenient to have the following identities,

$$s_{ij} = yP^2 \quad (1.121a)$$

$$s_{ik} = zP^2(1 - y) \quad (1.121b)$$

$$s_{jk} = P^2(1 - y)(1 - z). \quad (1.121c)$$

For the soft function S that is independent of the splitting type we arrive with the following expression (see App. A.1.4):

$$\sum_{\lambda_i=\pm} \sum_{\lambda_j=\pm} S = \frac{1}{P^2} \left(\frac{4}{y} \frac{z(1-y)}{1-z(1-y)} - \frac{4}{1-z(1-y)} (p_k - p_i) \cdot Q_j - \frac{4}{y} p_i \cdot Q_j \right), \quad (1.122)$$

where we also abbreviated

$$Q = \frac{q^\mu}{p \cdot q} \quad (1.123)$$

so that $p \cdot Q = 1$. The contribution P for the quark-gluon splitting is

$$\sum_{\lambda_i=\pm} \sum_{\lambda_j=\pm} P_{(ij) \rightarrow i+j} = \frac{1}{P^2} \left(\frac{2(1+z)}{y} p_i \cdot Q_j + \frac{2}{y} - \frac{2}{1-y} p_k \cdot Q_j \right) \quad (1.124)$$

so that

$$\sum_{\lambda_i=\pm} \sum_{\lambda_j=\pm} (P_{(ij)\rightarrow i+j} + S) = \frac{1}{P^2} \left(\frac{2(1+z)}{y} p_i \cdot Q_j + \frac{2}{y} - \frac{2}{1-y} p_k \cdot Q_j \right. \\ \left. + \frac{4}{y} \frac{z(1-y)}{1-z(1-y)} - \frac{4}{1-z(1-y)} (p_k - p_i) \cdot Q_j - \frac{4}{y} p_i \cdot Q_j \right) \quad (1.125)$$

which is very similar to the Catani-Seymour dipole

$$D = \frac{1}{P^2} \left(\frac{4}{y} \frac{1}{1-z(1-y)} - \frac{2}{y} - \frac{2z}{y} \right). \quad (1.126)$$

The difference is

$$\sum_{\lambda_i=\pm} \sum_{\lambda_j=\pm} (P_{(ij)\rightarrow i+j} + S) - D = \frac{1}{P^2} \left(\frac{2z + 2(z-1)p_i \cdot Q_j}{y} \right. \\ \left. - \frac{2p_k \cdot Q_j}{1-y} - \frac{4(p_k - p_i) \cdot Q_j}{1-z(1-y)} \right), \quad (1.127)$$

where the first term is integrable (see App. A.1.9), the second term is finite as $s_{ij} \rightarrow 0$ and the last term introduces a soft divergence, because in the soft limit $\lambda \rightarrow 0$ for $p_j = \lambda k$, $Q_j \sim \frac{1}{\lambda}$ and $y \sim \frac{1}{\lambda}$. This term must be subtracted because it would introduce an artificial divergence. It stems from S and therefore will also be present in the gluon-gluon split, so we define the corrected S for Catani-Seymour subtraction terms as

$$S \longrightarrow S' = S + \frac{4(p_k - p_i) \cdot Q_j}{1-z(1-y)} \quad (1.128)$$

so all terms have the correct behavior in the collinear and soft limit, if combined with the Catani-Seymour subtraction terms.

A check that the subtraction terms give the desired, integrable behavior is given in Sec. 1.5.7.

1.5.6 Summary of the Subtraction Terms

Here we summarize the subtraction terms $\mathcal{D}_{ij,k}$ and $\mathcal{D}'_{ij,k}(\theta_i, \theta_j)$. The Catani-Seymour subtraction terms are

$$\mathcal{D}_{ij,k} = A_\lambda^* (\{\tilde{p}_i\}_{i=1}^n) \frac{\delta_{\lambda\lambda'}}{P^2} \left(\frac{4}{y} \frac{1}{1-z(1-y)} - \frac{2}{y} - \frac{2z}{y} \right) A_{\lambda'} (\{\tilde{p}_i\}_{i=1}^n) \quad (1.129a)$$

$$\mathcal{D}'_{ij,k} = A_\lambda^* (\{\tilde{p}_i\}_{i=1}^n) \frac{1}{P^2} \left(-g^{\mu\nu} \frac{4}{y} \left(\frac{1}{1-z(1-y)} - 1 \right) + \frac{4}{y^2 P^2} \left(z^2 p_i^\mu p_i^\nu \right. \right. \\ \left. \left. - 2z(1-z) p_i^\mu p_j^\nu + (1-z)^2 p_j^\mu p_j^\nu \right) \right) A_{\lambda'} (\{\tilde{p}_i\}_{i=1}^n) \quad (1.129b)$$

where the first term is applied if i is a(n) (anti-)quark and the second one if i is a gluon. The second term must be converted to a scalar as described in Sec. 1.5.6.1.

The additional subtraction terms for the helicity dependent parts are

$$\mathcal{D}'_{ij,k}(\theta_i, \theta_j) = A_\lambda^* (\{\tilde{p}_i\}_{i=1}^n) \mathcal{R} \left(P_{(ij) \rightarrow i+j} + S' \right) A_{\lambda'} (\{\tilde{p}_i\}_{i=1}^n) \quad (1.130)$$

with the \mathcal{R} -operator as defined in Sec. 1.5.4.6, S' in Eq. (1.128) and $P_{(ij) \rightarrow i+j}$ in Eq. (1.102). The final formula then is Eq. (1.115) with the sum $\sum_{k \neq i,j}$ running over the tuples (i, j, k) and (k, j, i) with

$$j \in \{1, \dots, n\} \quad (1.131a)$$

$$i = j - 1 \pmod{n+1} \quad (1.131b)$$

$$k = j + 1 \pmod{n+1} \quad (1.131c)$$

so e.g. for $e^+e^- \rightarrow q_2 g_3 \bar{q}_1$ the tuples are $(2, 3, 1)$ and $(1, 3, 2)$.

The amplitude $A_\lambda (\{\tilde{p}_i\}_{i=1}^n)$ are n -particle amplitudes with the tilded momenta described in Sec. 1.5.4.3 where particle (ij) is polarized with helicity λ .

1.5.6.1 Spin-Correlation Matrix

The spin-correlation tensor $S^{\mu\nu}$ fulfills the property

$$S_{\mu\nu} \tilde{p}_{ij}^\mu \tilde{p}_{ij}^\nu = 0 \quad (1.132)$$

which is immediately clear for the case $S^{\mu\nu} = g^{\mu\nu}$ and can be shown for $S^{\mu\nu} = v^\mu v^\nu$ with $v^\mu = z_i p_i^\mu - z_j p_j^\mu$; see App. A.1.8. We can therefore use the following formula which is more convenient to calculate the subtraction terms:

$$\begin{aligned} A_\mu^* S^{\mu\nu} A_\nu &= \sum_{\lambda=\pm} \sum_{\lambda'=\pm} (A_\alpha \varepsilon_\lambda^\alpha(\tilde{p}_{ij}))^* \varepsilon_\lambda^\mu(\tilde{p}_{ij}) S_{\mu\nu} \varepsilon_{\lambda'}^\nu(\tilde{p}_{ij}) (\varepsilon_{\lambda'}^\beta(\tilde{p}_{ij}) A_\beta) \\ &= \sum_{\lambda=\pm} \sum_{\lambda'=\pm} \mathcal{A}_\lambda^* S_{\lambda\lambda'} \mathcal{A}_{\lambda'}. \end{aligned} \quad (1.133)$$

In the first step we twice used the identity

$$-g^{\mu\nu} = \sum_{\lambda=\pm} \varepsilon_\lambda^{\mu*}(\tilde{p}_{ij}) \varepsilon_\lambda^\nu(\tilde{p}_{ij}) - \frac{\tilde{p}_{ij}^\mu q^\nu + q^\mu \tilde{p}_{ij}^\nu}{\tilde{p}_{ij} \cdot q} \quad (1.134)$$

and made use of the fact that the contribution of the fraction vanishes because either $A \cdot \tilde{p}_{ij} = 0$ (gauge invariance) or $\tilde{p}_{ij}^\mu S_{\mu\nu} \tilde{p}_{ij}^\nu = 0$, Eq. (1.132). In the second line we abbreviated $\mathcal{A}_\lambda = A_\mu \varepsilon_\lambda^\mu(\tilde{p}_{ij})$ which is now an ordinary- n particle amplitude with particle ij having the momentum \tilde{p}_{ij} and helicity λ , and

$$S_{\lambda\lambda'} = \varepsilon_\lambda^\mu(\tilde{p}_{ij}) S_{\mu\nu} \varepsilon_{\lambda'}^\nu(\tilde{p}_{ij}) \quad (1.135)$$

which we will call spin-correlation matrix.

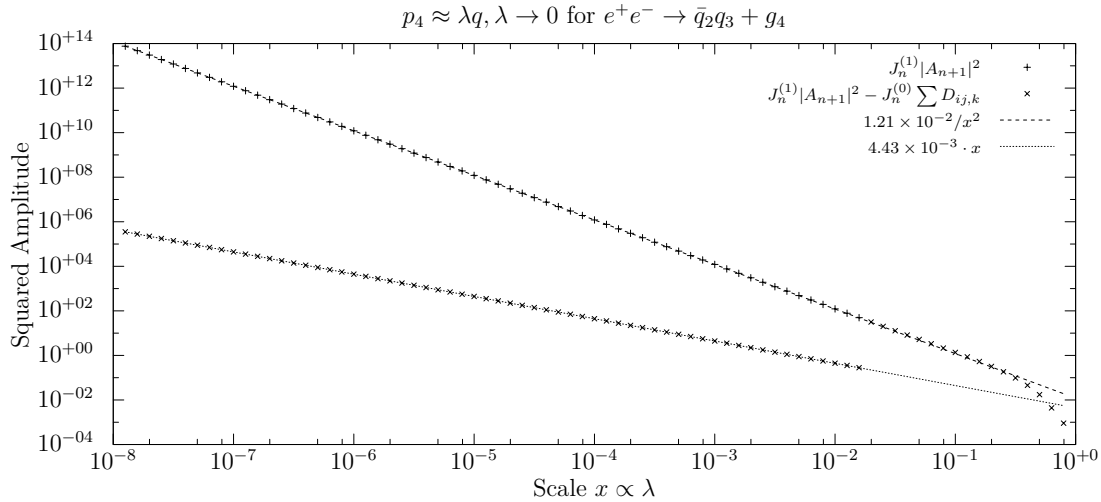


Figure 1.4: *Soft behavior of the unsubtracted and subtracted matrix elements for the two jet corrections for a randomly generated phase space point (see App. A.5.2.1) when a single momentum $p_j = \lambda q$ is rescaled with λ .*

1.5.7 Check of the Local Behavior

To check the local behavior we compare the unsubtracted against the subtracted matrix elements and check if they scale correctly in the soft and collinear limits.

1.5.7.1 Check of the Soft Behavior

The soft limit is generated with a phase space generator as described in App. A.5.2.1. It generates a set of momenta that fulfill momentum conservation and contains one soft momentum $p_j = \lambda q$ where the index j can be freely chosen. We are mainly interested in soft gluons, but it is also possible to check soft (anti-)quarks. It is possible that the generator produces more than one soft particle, but in that case the phase space momentum is vetoed by the jet algorithm (it corresponds to a higher-order correction).

Fig. 1.4 shows the possible situations during the calculation of the production for two jets in e^+e^- collisions at 90 GeV for randomly chosen phase space points where λ is scaled from a small value (soft case) to a high value. The two-jet correction is a good example to begin with, because it exclusively tests the quark-gluon splitting. To test also the gluon-gluon splitting one needs at least one more gluon; an example is shown in Fig. 1.5 which shows some selected situations in the calculation for seven-jet production.

In general, one can observe the expected (see Sec. 1.5.4.1) behavior for the unsubtracted matrix elements that follow $1/\lambda^2$. The subtraction terms correct this

behavior by correctly subtracting this part, leaving the remaining terms that follow a $1/\lambda$ behavior that is integrable. Depending on the randomly generated phase space point a different but constant number of dipoles are active in the soft limit. Far away from this limit the approximation of λ^x , $x \in \{1, 2\}$ no longer holds true and the matrix elements deviate from it.

1.5.7.2 Check of the Collinear Behavior

The collinear limit is generated as described in App. A.5.2.2. This generates two momenta p_i and p_j with indices i, j that can again be freely chosen. On the x-axis we determine the collinearity $k = \sqrt{2p_i \cdot p_j}$, so that the collinear limit is on the left side of small k .

Fig. 1.6 shows the possible situations that can happen during a two-jet calculation. Not shown is the situation when the quark becomes collinear to the antiquark which does not exhibit a collinear divergence. If other particles become collinear we see the $1/k^2$ -behavior that is canceled by the subtraction terms to an integrable $1/k$ -behavior. Note that the subtracted matrix elements suffer from numerical inaccuracies in the extreme collinear region. However this is due to the generation of the collinear momenta which involves numerical root finding and therefore is not present in an actual calculation. See also Sec. 1.2.1.1 for more discussion on numerical problems.

In both the collinear and the soft plots one can see that the $n + 1$ -matrix element stop at a certain point. This is where the $n + 1$ momenta no longer pass through the jet algorithm as a valid n -jet event and are therefore most likely a $n + 1$ event. The recombined (tilded) momenta in Eq. (1.106) however pass the n -jet tests and approximate the $n + 1$ -matrix element.

1.5.7.3 Numerical Precision

Fig. 1.4, 1.5, 1.6, 1.7 were obtained with extended precision (long double) codes so we could look at the behavior for a large region without numerical problems. The plots are similar for double-precision, although the numerical problems in the collinear regions start about one magnitude earlier.

1.5.8 Application to n-Jet Cross Sections

The ultimate test of our newly derived subtraction terms for random polarizations is the check whether the integrated results agree with those obtained by helicity eigenstates and Catani-Seymour subtraction terms. A comparison of both methods can be found in Tab. 1.6 which lists the results for the real correction to jet

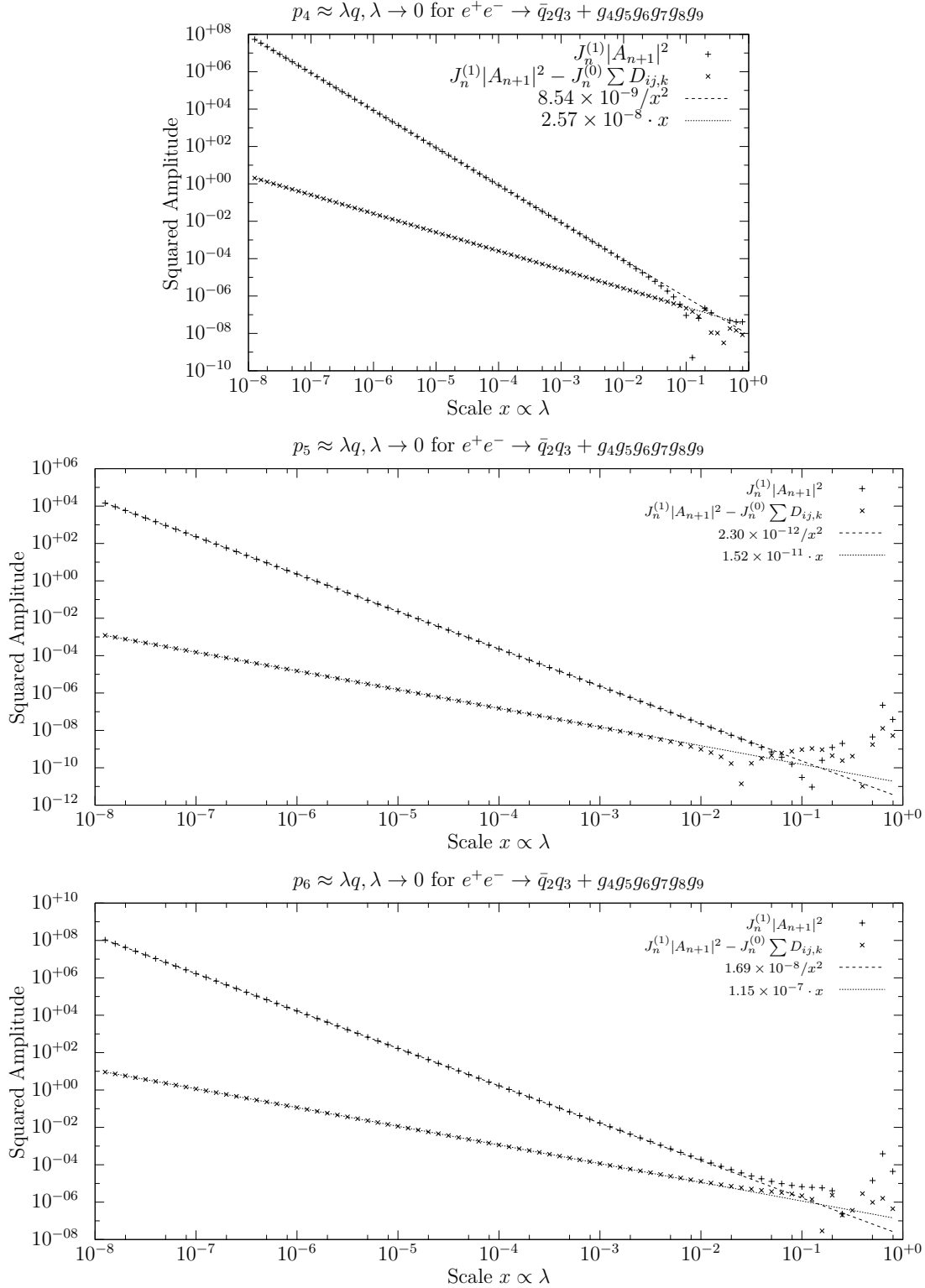


Figure 1.5: Soft behavior of the unsubtracted and subtracted matrix elements for the seven jet correction for a randomly generated phase space point (see App. A.5.2.1) when a single momentum $p_j = \lambda q$ is rescaled with λ .

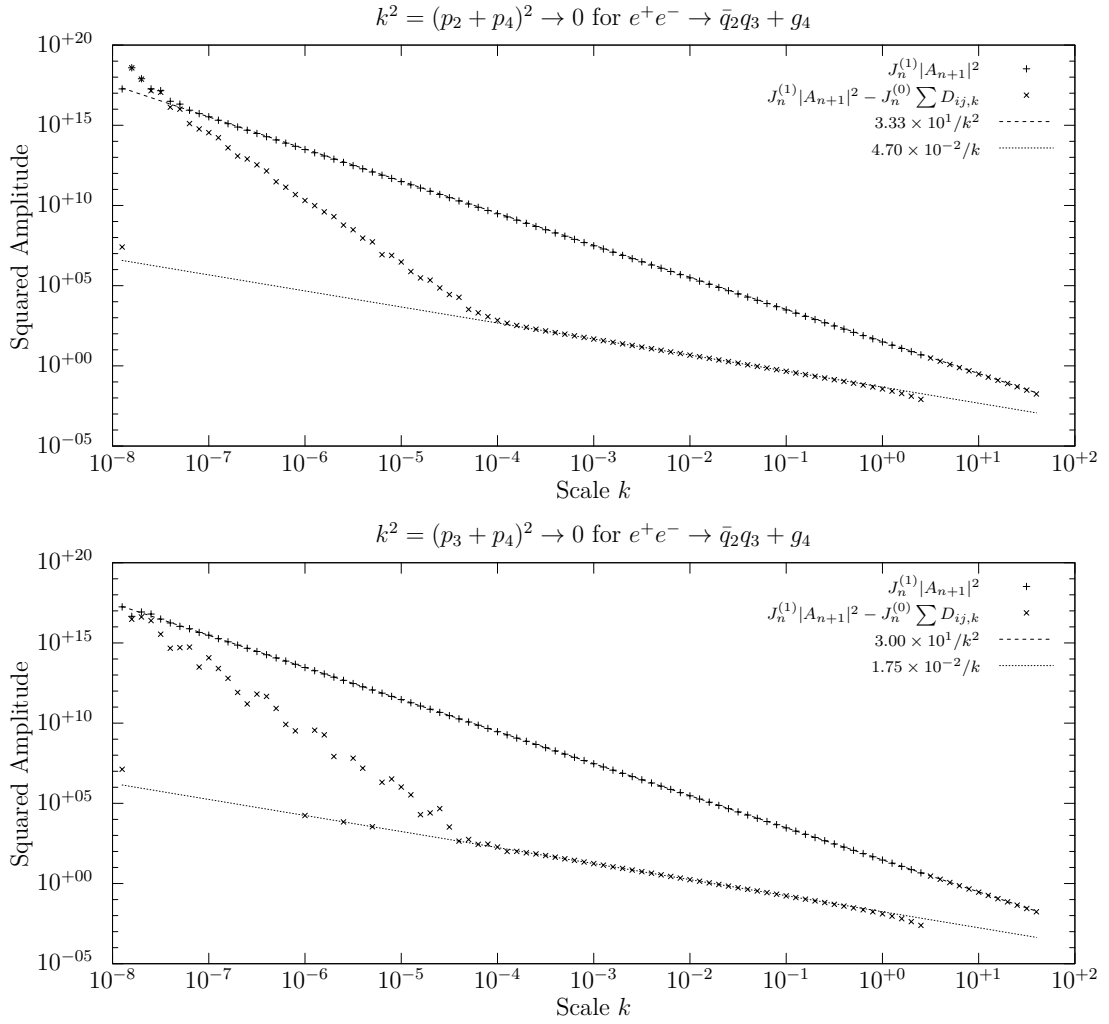


Figure 1.6: Collinear behavior of the unsubtracted and subtracted matrix elements.

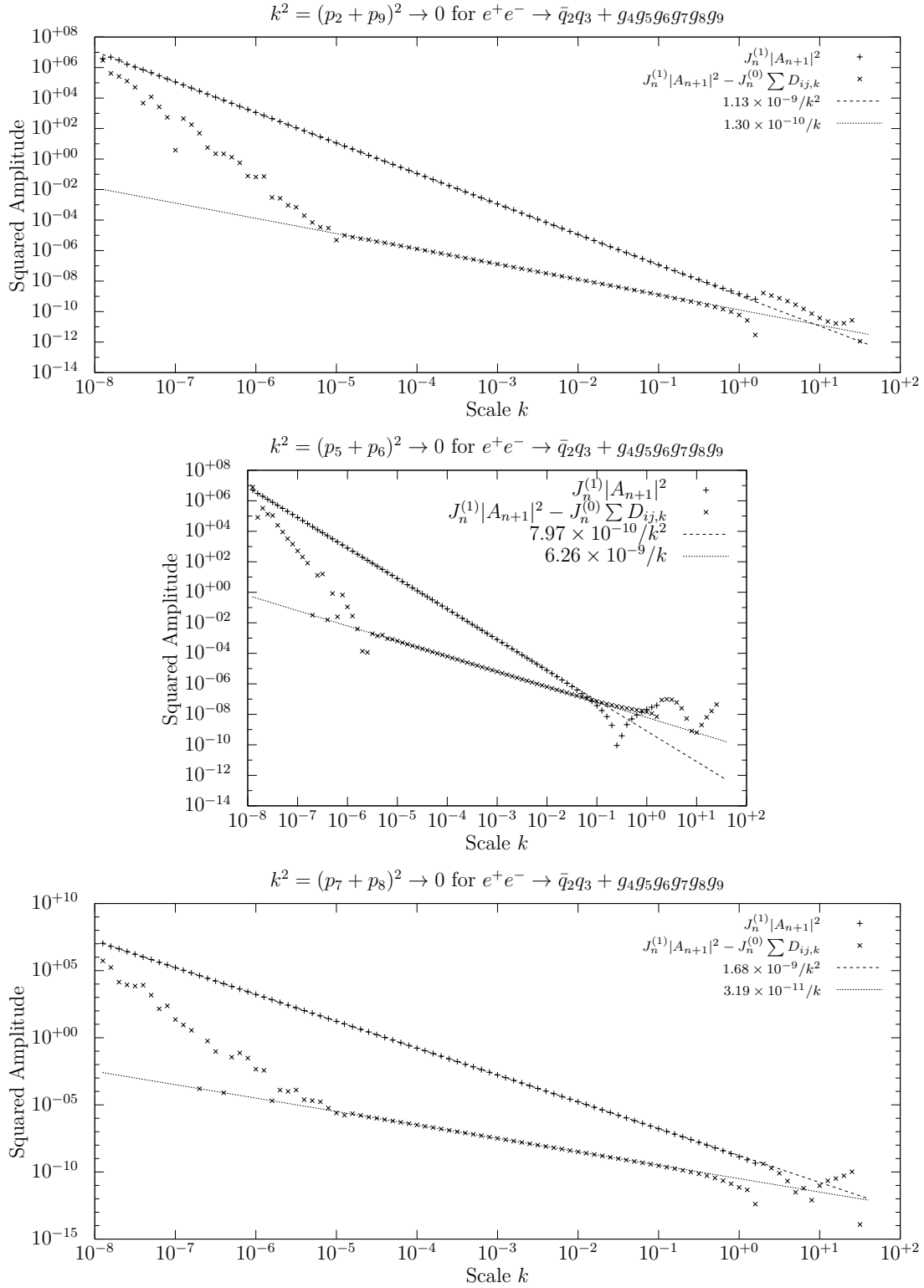


Figure 1.7: Collinear behavior of the unsubtracted and subtracted matrix elements.

Random Polarizations					
	N	E		χ^2/dof	t_{CPU}
		[pb]	[%]		[s]
$e^+e^- \rightarrow \bar{q}q$	10^6	-48.489 ± 0.041	0.08	0.81	1078
$e^+e^- \rightarrow \bar{q}qg$	10^7	-34.256 ± 0.074	0.22	1.13	14 960
$e^+e^- \rightarrow \bar{q}qgg$	10^8	-13.802 ± 0.047	0.34	1.75	195 753
$e^+e^- \rightarrow \bar{q}qggg$	10^9	-3.610 ± 0.018	0.50	1.42	3 061 408
Helicity Eigenstates					
	N	E		χ^2/dof	t_{CPU}
		[pb]	[%]		[s]
$e^+e^- \rightarrow \bar{q}q$	10^6	-48.476 ± 0.022	0.04	1.09	185
$e^+e^- \rightarrow \bar{q}qg$	10^7	-34.215 ± 0.029	0.09	0.51	9043
$e^+e^- \rightarrow \bar{q}qgg$	10^8	-13.826 ± 0.021	0.15	1.13	431 949
$e^+e^- \rightarrow \bar{q}qggg$	10^8	-3.637 ± 0.014	0.40	0.98	2 347 488

Table 1.6: Results for radiative corrections of the total cross section $\sigma(e^+e^- \rightarrow \gamma^* \rightarrow \bar{q}q + (n-2)g$ summed over all light quark-flavors $q = u, d, c, s, b$ at $\sqrt{s} = 90 \text{ GeV}$. The results were obtained using 10 VEGAS iterations, each with N calls and weighing the results as described in Sec. 2.3.3.

production in e^+e^- collisions at $\sqrt{s} = 90 \text{ GeV}$. Both methods yield results that agree well within the error-bound, the last result agrees in within three error bounds.

1.5.8.1 Efficiency of the Randomly Polarized Dipoles

We can now compare both methods again as we did in Sec. 1.3.2. The results are given in Tab. 1.7.

The results for the additional errors introduced by the random polarizations are similar to the LO case (see Tab. 1.5) for a low multiplicity, i.e. two or three times higher than with helicity eigenstates. For $n = 5$ the error is four times higher. Unfortunately our implementation is not as efficient as would be needed so the results for x are lower than one meaning that the traditional method with helicity summation is faster. This, however, does not necessarily mean that one should disfavor random polarizations for real corrections, because there is still room to improve the implementation. The main advantage of the Catani-Seymour dipole terms is that they can be expressed in invariants which are evaluated very fast,

	$\frac{\sigma_{\text{HS}}}{\sigma_{\text{RP}}}$	$f = \frac{t_{\text{HS}}}{t_{\text{RP}}}$	$x = f \left(\frac{\sigma_{\text{HS}}}{\sigma_{\text{RP}}} \right)^2$
$e^+e^- \rightarrow \bar{q}q$	0.537	0.17	0.05
$e^+e^- \rightarrow \bar{q}qg$	0.392	0.60	0.09
$e^+e^- \rightarrow \bar{q}qgg$	0.447	2.21	0.44
$e^+e^- \rightarrow \bar{q}qggg$	0.246	7.67	0.03

Table 1.7: Comparison of random polarizations vs. helicity eigenstates in the real correction. These results were computed with the numbers in Tab. 1.6.

see Eq. (1.129). The additional dipoles for the helicity dependent terms must be evaluated using the \mathcal{R} -operator which we implemented naïvely as its definition in Eq. (1.117) suggest. Future work can improve this by deriving expressions of Eq. (1.130) where the \mathcal{R} -operator is performed explicitly and thus avoids the subtraction of the helicity summed result.

CHAPTER 2

Monte Carlo Integration

Monte Carlo (MC) integration is the name for a class of algorithms that numerically approximate definite integrals. The name refers to its characterizing feature, i.e. its use of (pseudo-)random numbers to perform the integration. It is a popular choice because, in contrast to other methods, the error of the approximation converges independently of the integration dimension — an important criterion since we will need to perform high-dimensional (phase space-)integrals. In addition MC integration allows to make use of knowledge about the integrand, even if it is only approximate, to further improve the approximation.

This chapter is organized as follows: In Sec. 2.1 we will give a short introduction into the simplest form of MC integration, commonly called “plain” or “naïve” MC. In Sec. 2.2 we explain some advanced techniques; the VEGAS[36, 37] MC algorithm that makes use of these will be presented in Sec. 2.3. The discussion of VEGAS is supplemented with a close examination of its adaptive strategy. We include this for two reasons: The algorithm as found in common implementations differs (slightly) from the one defined in the original implementation and the other reason is an educational one; to build upon or even to improve it one needs to understand it first. This chapter ends with Sec. 2.4 which presents the author’s own implementation of the aforementioned algorithms and discusses some technical problems, e.g. parallelization in Sec. 2.4.1 for computer clusters.

2.1 Introduction

Let us define the general problem. We need an approximation of the d -dimensional integral I of the integrand f defined over the domain $U = [0, 1]^d$,

$$I = \int_{[0,1]^d} d^d x f(\vec{x}). \quad (2.1)$$

The domain of the integrand f is restricted to the unit hypercube $[0, 1]^d$ for convenience. We assume that domains different from U can be remapped by a

change of variables. We also require that f is finite everywhere and that I is finite as well.

2.1.1 Plain Monte Carlo Integration

By artificially rewriting the integral I we see that we can interpret it as an expectation value E of the function f

$$I = \int_{\mathbb{R}^d} d^d x p(\vec{x}) f(\vec{x}) = E[f] \quad (2.2)$$

with a d -dimensional *uniform probability distribution*

$$p(\vec{x}) = \begin{cases} \forall \vec{x} \in [0, 1]^d : 1 \\ \forall \vec{x} \notin [0, 1]^d : 0 \end{cases} . \quad (2.3)$$

Using N numbers $\{\vec{x}_i \in [0, 1]^d\}_{i=1}^N$ uniformly distributed (i.e. according to p) the expectation value can then be approximated by the *sample average* E_N ,

$$E_N = \frac{1}{N} \sum_{i=1}^N f(\vec{x}_i) \longrightarrow E[f] = I, \quad (2.4)$$

where the convergence is guaranteed by the law of large numbers.

2.1.2 Convergence Behavior

If, additionally, the function f is square-integrable, i.e. $\int_U d^d x f^2(\vec{x}) < \infty$, then the variance

$$V = \int_{[0,1]^d} d^d x (f(\vec{x}) - I)^2 = \int_{[0,1]^d} d^d x f^2(\vec{x}) - I^2 \quad (2.5)$$

exists and the central limit theorem applies and implies that the sample average converges in distribution to a normal one

$$E_N \xrightarrow{d} \mathcal{N}\left(I, \frac{V}{N}\right), \quad (2.6)$$

with variance

$$S^2 = \frac{V}{N} \quad (2.7)$$

that serves as an error and describes how well the approximation $E_N \approx I$ is. In practice the variance V is approximated (in the same sense as I is approximated with E_N) with the *sample variance*

$$V_N = \frac{1}{N-1} \sum_{i=1}^N (f(\vec{x}_i) - E_N)^2 \quad (2.8)$$

because the value of V is as inaccessible as that of the integral I . The factor $N/N-1$ that was applied to S^2 (the denominator with $N-1$ instead of N) is called Bessel's correction and accounts for the reduced number of freedoms since E_N is computed with the set $\{f_i = f(\vec{x}_i)\}_{i=1}^N$ as well. In practice this correction factor is neglected because N is usually very high.

2.1.3 The “Curse of Dimensionality”

As can be seen from Eq. (2.7) the convergence of the error S of the approximation follows $1/\sqrt{N}$, i.e. is not affected by the dimension d of the integrand. For $d = 1$ this is worse than e.g. the simplest quadrature rules, but in higher dimensions it outperforms these classical algorithms that scale exponentially in d . This is usually referred to as the “curse of dimensionality” which means that these methods fail to provide a reasonable answer for high-dimensional integrals. A qualitative explanation why MC converges faster is that by evaluating the function at random points in the limit of high sample size N is sampled equally well for any N ; the classical algorithms mentioned above all use a grid decomposing the unit hypercube U along its d dimensions each with r intervals into a total of d^r cells. To achieve a regular sampling the integrand has to be evaluated in each cell and therefore the exponential scaling arises.

A hybrid approach are the so-called quasi-random numbers[38] that generate numbers in a way that the unit hypercube is sampled more regularly than they would with a pseudo-random number generator.

However, converging independently of the integration dimension does not imply that the MC error is independent of the dimension of the function. In fact, as can be seen from Eq. (2.6) the dimension of the function indirectly enters via the variance V that is induced by f . In practice one observes e.g. that the jet cross sections require more statistics if the number of particle is higher (see Sec. 1). To compensate high variances we introduce some variance reduction techniques in the next section.

2.2 Variance Reduction Techniques

The error of a MC integration algorithm was derived as

$$S = \frac{\sigma}{\sqrt{N}}. \quad (2.9)$$

When we are interested in decreasing this error, there are several ways: If we have enough computational power we can simply increase the sample size N ; for example

by increasing the number of integrand evaluations N by four we can cut the error by 50%. Another way is to modify the function in a way that the value of the integral I is preserved but the standard deviation σ is decreased: This can e.g. be achieved with importance sampling (Sec. 2.2.1), control variates (Sec. 2.2.2) and antithetic variates [38, 39]. Finally there are hybrid approaches which subdivide the function into subfunctions and treat those according to their contribution to the total variance. This subdivision can be e.g. a spatial one, which is then called stratified sampling, or based on the peak-structure e.g. with multi-channel MC [40].

2.2.1 Importance Sampling

Importance sampling is based on the idea that one reduces the variance by distributing more integrand evaluations into “important” regions, i.e. where the function has a high value. Ideally, we thereby smooth out the function so that the resulting one fluctuates less and giving a smaller variance.

For this purpose let $p(\vec{x})$ be a probability distribution function that is properly normalized

$$\int_{[0,1]^d} d^d x p(\vec{x}) = 1, \quad (2.10)$$

and positive everywhere on the hypercube. Then we can rewrite Eq. (2.1) to

$$I = \int_{[0,1]^d} d^d x p(\vec{x}) \frac{f(\vec{x})}{p(\vec{x})} \quad (2.11)$$

and perform the substitution $d^d x p(\vec{x}) = d^d y$.

If the PDF is separable, i.e. if $p(\vec{x}) = \prod_{i=1}^n p_i(x_i)$, then the problem effectively reduces to d one-dimensional ones and the new variables

$$y_i = \int_0^{x_i} dx'_i p_i(x'_i) \quad (2.12)$$

are the cumulative probability functions (CDF) that fulfill $y_i(0) = 0$ and $y_i(1) = 1$ and therefore map the hypercube to itself. This enables us to rewrite Eq. (2.11) to

$$I = \int_{[0,1]^d} d^d y \frac{f(\vec{x}_{\vec{y}})}{p(\vec{x}_{\vec{y}})} \quad (2.13)$$

with $\vec{x}_{\vec{y}}$ being understood as the inverse of Eq. (2.12), called inverse CDF or quantile function. Then Eq. (2.13) is approximated with

$$I \approx \frac{1}{N} \sum_{i=1}^N \frac{f(\vec{x}_{\vec{y}_i})}{p(\vec{x}_{\vec{y}_i})} \quad (2.14)$$

where now \vec{y} is a vector of d random variables $y_i \in [0, 1]$ each uniformly distributed. This vector is mapped to \vec{x} by the inverse CDF which makes clear that in practice we have an additional and stricter requirement to $p(\vec{x})$ used for importance sampling: Its inverse CDF must be “easily” converted into code, e.g. where the PDF is piecewise-constant.

One question still remains: What is the optimal choice of the PDF $p(\vec{x})$? Obviously, we want to minimize the variance

$$V = \int d^d x p(\vec{x}) \left(\frac{f(\vec{x})}{p(\vec{x})} - I \right)^2 \quad (2.15)$$

under the constraint that p be a positive, normalized function. This gives (for proof see App. A.1.10):

$$p(\vec{x}) = \frac{|f(\vec{x})|}{\int d^d x' |f(\vec{x}')|}. \quad (2.16)$$

If we insert this back into the expression for the variation we have

$$V = \left(\int d^d x |f(\vec{x})| \right)^2 - \left(\int d^d x f(\vec{x}) \right)^2 \quad (2.17)$$

which vanishes if f has the same sign everywhere on the hypercube.

2.2.2 Control Variates

Another method to reduce the variance are control variates. Here we subtract a function g that is similar to the function f that we want to integrate:

$$I = \int_{[0,1]^d} d^d x (f(\vec{x}) - g(\vec{x})) + I' \quad (2.18)$$

and add it back via the integral

$$I' = \int_{[0,1]^d} d^d x g(\vec{x}) \quad (2.19)$$

which, by construction of g , can be performed analytically. Obviously, the best choice which would set the variance to zero is $g = f$. However, this result is useless in practice because it shifts the problem of finding I to I' .

2.3 The VEGAS Integration Algorithm

The VEGAS[36, 37] integration algorithm combines importance sampling with an adaptive strategy: Instead of calling the integrand-function N times, this number

is subdivided into m smaller ones,

$$N_1, N_2, \dots, N_m \quad \text{with} \quad \sum_{i=1}^m N_m = N. \quad (2.20)$$

The number m is the number of *iterations* VEGAS performs and typically the N_i are chosen equal. In each iteration i importance sampling is used with N_i integrand evaluations and a PDF $p_i(\vec{x})$ constructed from the previous iteration $i - 1$. For the first iteration the PDF used is

$$p_1(\vec{x}) = 1, \quad (2.21)$$

due to lack on information about the integrand, i.e. the first iteration is a plain MC integration.

2.3.1 Automatic PDF construction in VEGAS

How are the p_i constructed for subsequent iterations? The only information the algorithm can make use of are the function values $f_j = f(\vec{x}_j)$ at randomly chosen points \vec{x}_j . VEGAS uses these evaluations not only to give an approximation of the integral, but also to construct the p_i 's. To simplify matters, it uses a separable PDF

$$p_i(\vec{x}) = \prod_{k=1}^d p_{i,k}(x_k) \quad (2.22)$$

where the $p_{i,k}$ are piecewise-constant functions for iteration i and dimension k . Note that this changes the optimal PDFs to

$$p_i(x_i) = \frac{\bar{f}(x_i)}{\int dx'_i \bar{f}(x'_i)} \quad \bar{f}^2(x_i) = \int \left(\prod_{k \neq i}^d dx'_k \right) \frac{f^2(\vec{x}'|_{x'_i=x_i})}{\prod_{j \neq i}^d p_j(x'_j)} \quad (2.23)$$

instead of Eq. (2.16), see App. A.1.10.

The piecewise-constant functions are constructed with M constant pieces with boundaries (from now on we will drop the indices i and k)

$$0 = x_0 < x_1 < \dots < x_M = 1. \quad (2.24)$$

We will call the set of all boundaries *grid* and two neighboring boundaries $[x_{j-1}, x_j]$ the *bin* j . With this the PDFs are defined as

$$p(x) = \frac{1}{M \Delta x_j} \quad (2.25)$$

with bin-size

$$\Delta x_j = x_j - x_{j-1} \quad (2.26)$$

where j is determined by the bin x falls into: $\exists j : x \in [x_{j-1}, x_j]$. This choice is a bin-uniform one, meaning that the probability for a random-number x falling into bin j depends only on the number of bins:

$$\int_{x_{j-1}}^{x_j} dx p(x) = \frac{1}{M}. \quad (2.27)$$

An important advantage of this piecewise-constant PDF is that we can write down both the CDF

$$y = \int_0^x dx' p(x') = \frac{j-1}{M} + \frac{x-x_{j-1}}{M\Delta x_j}, \quad (2.28)$$

and its inverse

$$x = x_{j-1} + \left(y - \frac{j-1}{M}\right) M\Delta x_j \quad (2.29)$$

in a closed form which is used to transform the uniformly generated numbers $y \in [0, 1]$ into ones that are distributed according to Eq. (2.25). The index j is chosen to be $j = \lceil yM \rceil$.

2.3.2 The Rebinning Algorithm

It remains to discuss how the grid $\{x_j\}_{j=0}^M$ is computed. Initially it is set to

$$x_{j+1} = x_j + \frac{1}{M} \quad \text{with} \quad x_0 = 0 \quad (2.30)$$

giving an equally-spaced grid ($\Delta x_j = \frac{1}{M}$) that correctly reproduces Eq. (2.21). After each iteration (and for every dimension) the *rebinning* procedure

$$\{x_j\}_{j=0}^M \longrightarrow \{x'_{j'}\}_{j'=0}^M \quad (2.31)$$

updates the grid with the aid of the *binned importance values* c_j that tell us “how important” the integrand in bin j was; we will give a detailed definition later on. The new grid will be constructed such that its new bins will contain approximately the *average importance value*

$$\bar{c} = \frac{1}{M} \sum_{i=1}^M c_i. \quad (2.32)$$

Starting with $j = j' = 1$ and $c' = 0$ this is done by

1. summing the importance values c_j of the old bins $j, \dots, j+k$ to make their sum as close as possible the average \bar{c} , i.e. we determine the smallest k for which

$$c' = c' + \sum_{i=j}^{j+k} c_i > \bar{c}, \quad (2.33)$$

2. setting the new boundary

$$x'_{j'} = x_{j+k} - \Delta x_{j+k} \frac{c' - \bar{c}}{c_{j+k}}, \quad (2.34)$$

3. updating $j' = j' + 1$, $j = j + k$, $c' = c' - \bar{c}$ and

4. repeating this for all bins.

To understand the algorithm above we have to look at the two possible situations:

- If in step 1 it turns out that $k = 0$ then the importance value c_j in bin j is larger than the average \bar{c} so we have to shrink the bin. This is done in step 2 where the boundary is updated. If we assume the integrand contributes to the importance value c_j in a constant way over the whole bin size,

$$x \in [x_{j-1}, x_j] : \quad \frac{d}{dx} c_j(x) = 0, \quad (2.35)$$

then $\frac{\bar{c}}{c_j}$ is the fraction of the bin size that belongs to the new bin j' ; since c' contains the whole bin already we have to subtract the remaining part of the bin size, i.e the fraction $1 - \frac{\bar{c}}{c_j} = \frac{c_j - \bar{c}}{c_j}$.

- If $k > 0$ then we have to enlarge the bin. We do this by simply adding the next bin, as long as the importance values of the of bins are smaller in sum as the average. The last bin that we add makes the new importance value larger than the average so that we have to shrink it which was discussed before.

Note that the assumption shown in Eq. (2.35) is, in general, wrong but shifts the grid in the correct direction (“correct” as far as the algorithm can judge from data available in this iteration and as far as the separability of the integrand is true). Since we will perform more iterations the hope is that the grid will eventually converge to the optimal one. An alternative choice that the CUBA[41] implementation of VEGAS offers is to replace in Eq. (2.34)

$$c_{j+k} \quad \text{with} \quad \frac{c_{j+k} + c_{j+k-1}}{2} \quad (2.36)$$

which averages the importance of bin $j + k$ with its left-neighbor bin $j + k - 1$.

We now discuss the importance values c_j . They are computed using the binned integral values,

$$b_j = \sum_i (f_i \Delta x_j)^2 \quad (2.37)$$

where f_i is the function value $f(\vec{x}_i)$ with \vec{x}_i the vector whose k -th component x^k falls into the bin j , i.e. $x^k \in [x_{j-1}, x_j]$. Note that this choice of b_j was found in all

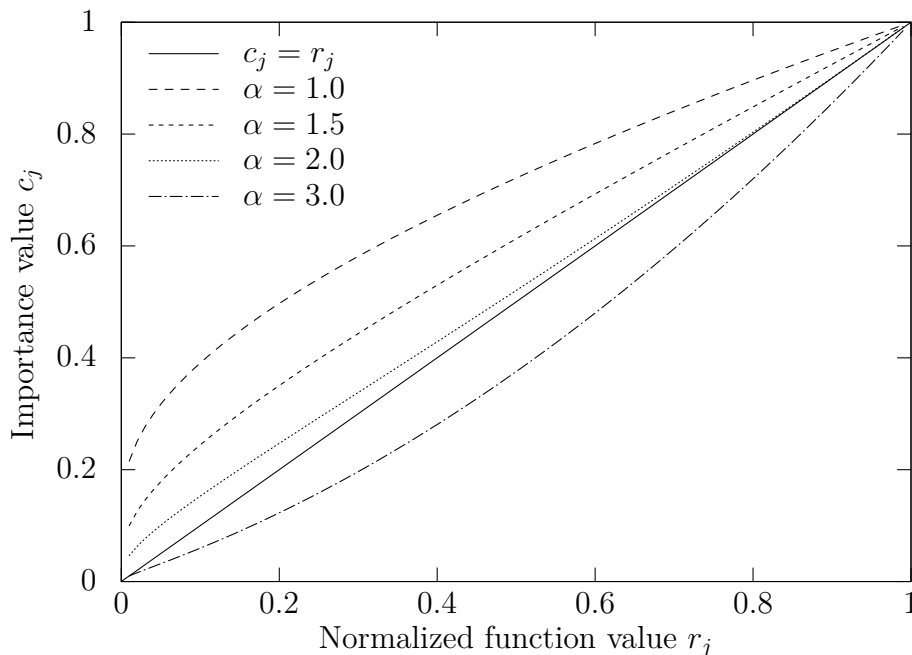


Figure 2.1: *The result of damping the importance function with Eq. (2.39) against a non-damped one*

VEGAS implementations[37, 41–43] the authors looked into and is in contrast to what is described in the original VEGAS publication[36] where it was suggested that $b_j = \sum_i |f_i| \Delta x_j$. The normalized bin values

$$r_j = b_j \left[\sum_{i=1}^M b_i \right]^{-1} \quad (2.38)$$

are then used to calculate the binned importance values

$$c_j = \left(\frac{r_j - 1}{\ln r_j} \right)^\alpha. \quad (2.39)$$

A simpler choice would be to set $c_j = r_j$, the choice above gives smaller r more importance and thereby reduces the difference between smaller and larger r . This dampens the grid adaption-process which can be understood by looking at the extreme case $\alpha = 0$. Here the importance is equal for every bin irrespective of its value ($c_j = 1$) and therefore every bin contains the average importance (2.32) which prevents that the boundaries are recomputed ($c' = \bar{c}$ in the algorithm above). For further discussion of the importance function see also Fig. 2.1.

2.3.3 Results and Chi-Square Test

For each iteration i VEGAS returns a tuple of approximation E_i and an error S_i of the integral I . These are combined into a weighted average

$$\bar{E} = \frac{1}{m} \sum_{i=1}^m w_i E_i \quad (2.40)$$

and a weighted error

$$\bar{S}^2 = \sum_{i=1}^m w_i^2 S_i^2 \quad (2.41)$$

with normalized weights[36]

$$w_i = \left[\sum_{i=1}^m \frac{1}{S_i^2} \right]^{-1} \frac{1}{S_i^2}. \quad (2.42)$$

Alternatively, one can include the estimations of the integral in into the weights

$$w'_i = \left[\sum_{i=1}^m \frac{E_i^2}{S_i^2} \right]^{-1} \frac{E_i^2}{S_i^2} \quad (2.43)$$

which are less prone to underestimations of the integral. This can happen in the earliest iterations when the integrand has peaks so sharp they are missed[36].

To verify if the individual results E_i agree with each other one performs a Chi-square test in which

$$\chi_{\text{dof}}^2 = \frac{1}{m} \sum_{i=1}^m \frac{(E_i - I)^2}{\sigma_i^2} \approx \frac{1}{m-1} \sum_{i=1}^m \frac{(E_i - \bar{E})^2}{S_i^2} \quad (2.44)$$

is the chi-square per degrees of freedom that should be around one.

2.3.4 Concluding Discussion

VEGAS' most severe limitation is its use of a separable grid function, i.e. its assumption that the integrand factorizes. If this is true, then VEGAS indeed constructs the optimal PDF. This assumption need not be true for the algorithm to work, however, it does affect the variance; in practice one notices that the variance can be greatly reduced if the integrand can be rewritten so it approximately factorizes in its maxima. If the integrand does not factorize, VEGAS can have the opposite effect where the grid destabilizes and the algorithm produces meaningless results that do not converge over the iterations.

Here we described a variant of VEGAS which does not use stratified sampling; this was found[37] to be ineffective for integrands with dimensions higher than $d = 4$.

2.4 Implementation and Related Details

Existing[37, 41–43] implementations e.g. of VEGAS have some shortcomings, among others they

- fix the value of α and the bin number M ,
- are not available in a parallel form,
- do not easily allow to specify the sample size N_i for each iteration,
- do not easily allow the access to the individual iteration results, e.g. to throw away the first iterations,
- make it impossible to specify custom random number generators, e.g. to use quasi-random numbers.

This motivated us to implement the algorithms ourselves which are freely available online[44] now. The library is written in C++11 which was chosen because it brings a new random number library (`#include <random>`) with many (pseudo-) random number generators such as the Mersenne-Twister[45] which is a good¹ choice for Monte Carlo integrations. An example how to use the integrator is shown in the following listing in line 28:

```

1 #include <hep/mc.hpp> // everything is in here, no library needs to be linked
2
3 #include <cstdint>
4 #include <iostream>
5 #include <vector>
6
7 // the function that will be integrated
8 double square(hep::mc_point<double> const& x)
9 {
10     return x.point[0] * x.point[0];
11 }
12
13 int main()
14 {
15     // this is what the approximation should give
16     double reference_result = 1.0 / 3.0;
17
18     // print reference result
19     std::cout << "computing integral of x^2 from 0 to 1\n";
20     std::cout << "reference result is " << reference_result << "\n\n";
21

```

¹The Mersenne Twister (MT) is good in the sense that it has 1) a period of $P = 2^{19937} - 1$ and 2) passes the k -distribution test[45] with $k \leq 623$ for 32-bit random numbers. The k -distribution test is considered a strong test for uniformity, i.e. in particular one tests if the distribution of P consecutively generated random numbers mapped to a k -dimensional hypercube is uniform.

```
22 // print results for each iteration
23 hep::vegas_callback<double>(hep::vegas_verbose_callback<double>);
24
25 // perform 5 iterations with 1000 calls each for a one-dimensional
26 // function with double precisions; parameters 4,5,6 are optional
27 // (comments show the default values)
28 auto results = hep::vegas<double>(
29     1,
30     std::vector<std::size_t>(5, 1000),
31     square /*,
32     128, // number of bins per dimension
33     1.5, // alpha value for rebinning
34     std::mt19937() // mersenne-twister random number generator */
35 );
36
37 // combine results from all iterations except the first in a single
38 // cumulative result
39 auto result = hep::cumulative_result0(results.begin() + 1, results.end());
40 double chi_square_dof = hep::chi_square_dof0(results.begin() + 1, results.end
    ());
41
42 std::cout << "cumulative result (without first iteration):\n";
43 std::cout << "N=" << result.calls << " I=" << result.value << " +- ";
44 std::cout << result.error << " chi^2/dof=" << chi_square_dof << "\n";
45
46 return 0;
47 }
```

2.4.1 Parallelization of VEGAS

In Sec. 2.2 we mentioned that it is possible to decrease the variance by simply increasing the number of integrand evaluation N . However, this also implies that the time needed to perform the evaluations increases with the same factor. This problem can be overcome by using (much) more computational power, i.e. more computers with more processors. To make use of them, however, one has to think of a way how to divide the problem into smaller sub-problems and how to distribute each of them onto different processors. This problem is called parallelization. The difficulty is that the sub-problems are usually not independent and therefore processors need to communicate with each other. Because the computations depend on the outcome of the communication the computation has to be halted and therefore effectively slows down. This can render certain algorithms completely unfeasible for parallelization. We will show that the VEGAS algorithm does not belong to this class and can be formulated in a way that gives a speedup near the number of parallel running processors.

In this section we will discuss a parallelization strategy that differs from the one proposed by Kreckel[46] and is similar to the method called “macro-parallelization (sync)” in this publication: For each of the p processors denoted by indices $0, 1, \dots, p - 1$ we

1. either
 - (a) seed the random number generator differently, or we
 - (b) seed it with the same number and discard random numbers such that we use different random numbers from the same generator,
2. start a single VEGAS iteration with $\frac{N}{p}$ integrand evaluations,
3. and send the cumulative variables (sum of the integrand, sum of the squares, grids) to each processor and calculate a single grid that will be used to perform the next iteration (step 2).

By seeding the random number generators differently we make sure that the random numbers are different (independent and identically distributed), otherwise we would obtain p times the same result. The processors work independently from each other and only at the end of each VEGAS iteration the grid refinement data is exchanged. This data is not too large (in the worst case $8 \cdot 128 \cdot (17 + 9) \approx 27$ kB for a LEP 7-Jet LO cross section²) and is exchanged using the the Message Passing Interface[47], in particular Open MPI[48], with a single `MPI_Allreduce` call.

In this approach there are two circumstances that lead to a “waste” of CPU time:

1. Each processor finishes its computation at a different time: This can and does happen because each phase space point is checked if it passes the jet-algorithm; if this is not the case, then the matrix element is not computed and the contribution to the phase-space integral is zero. These points are of course significantly faster than the ones for which the matrix element has to be computed. How many are actually computed depends on the random numbers and of course will be different for each processor. However, because each processor will, by construction of the algorithm above, use the same grid at each iteration, this difference should remain small if the sample size $\frac{N}{p}$ of each processor is high enough, and does not grow if more iterations are performed. Another reason that we experienced on the MOGON-cluster is that for large number of processes (> 128) the processes are started at different times.

The time difference to the slowest process is “wasted” because every processor then waits for the `MPI_Allreduce` to finish, which

²A double-precision (each double 8Byte large) grid with 128 bins for each dimension which is 16 for the phase-space and an additional 9 dimensions for the random polarization angles.

2. must exchange the grid adjustment data from every processor to every other processor. This is a rather costly operation, especially if many processors are involved, but can be implemented to scale logarithmically, see e.g. Ref. [49].

In practice this parallelization strategy works up to a certain number of processors p , as Fig. 2.2 suggests. Ideally the CPU-time should stay constant, but as $p = 128$ it doubles. Although the error is quite large (the band indicates the $1\text{-}\sigma$ deviation), this suggests that the processors spend too much waiting for the other processors to finish or respond to their message. When we increase the problem size by a magnitude, we see that now the percentage of “wasted” CPU-time is lowered which is almost constant.

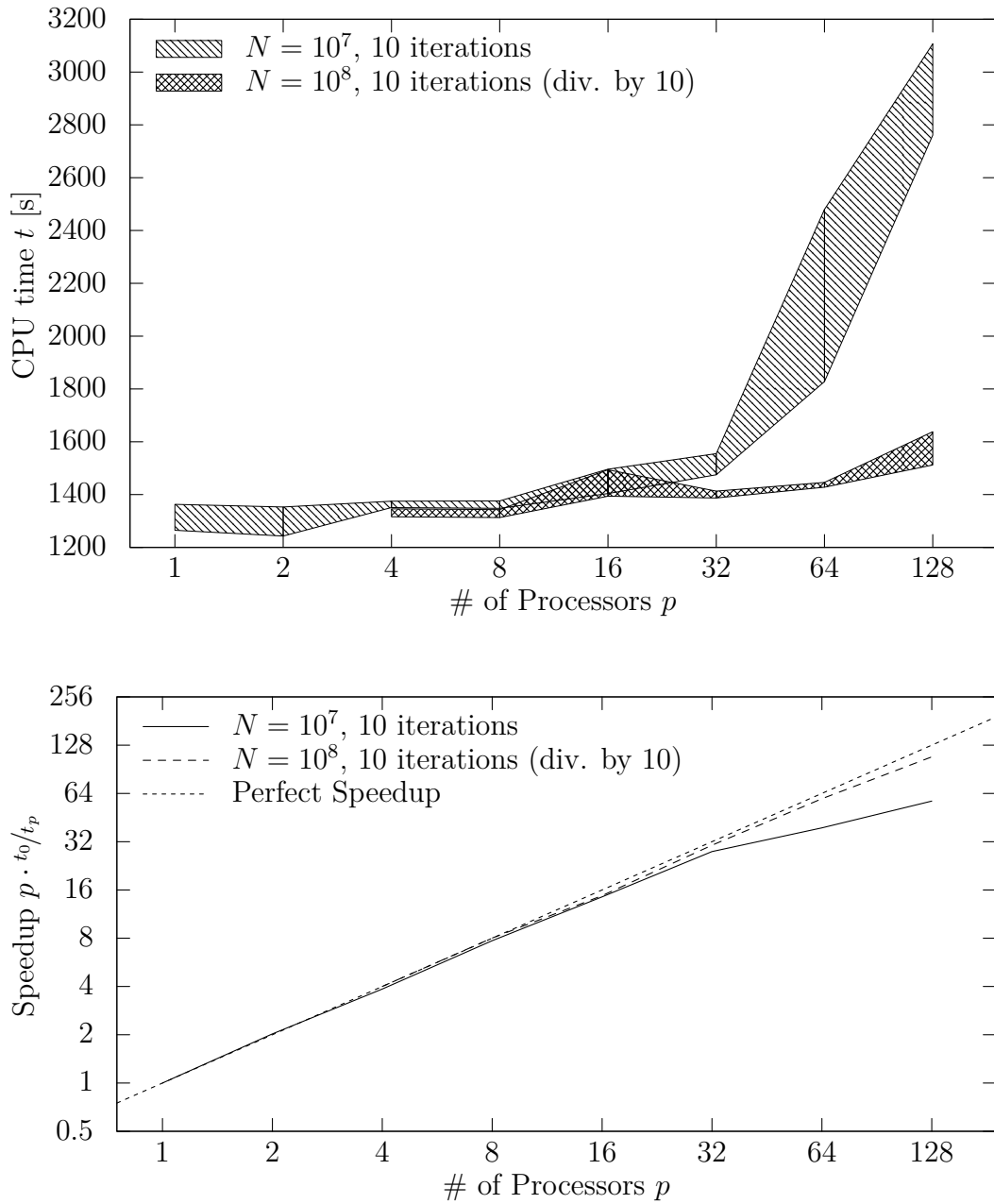


Figure 2.2: CPU-time spent for a total of 10 iterations distributed among p processors using a sample size of $N = 10^7$ and $N = 10^8$ per iteration. The bands are the 1- σ interval. The integral is a 5 jet LEP LC LO cross section. For $N = 10^7$ and $p > 32$ the CPU time strongly depends on the number of processors which is because the sample size N was too small; the real-time computation was less than 2 min for $p = 64$. Increasing N by a factor of 10 shows that the dependence gets smaller, as expected. The computations were performed on the MOGON cluster that has a node size of 64 processors.

CHAPTER 3

Conclusion and Outlook

In this thesis we have reported several techniques to speed up perturbative QCD calculations. The first idea we discussed was the leading-color approximation for which we showed that the error it introduces is less than 10 % for jet cross sections in e^+e^- scatterings at LO. The leading color approximation enables us to compute a squared amplitude that scales as n^4 which allows for the computation of observables with high multiplicities. The limiting factor is the phase-space integration.

An important idea are the random polarizations that enable us to integrate over newly defined helicity-angles instead of summing over the 2^n helicity configuration. As shown in Sec. 1.3.2 this allows us to gain an effective speedup x that ranges from one for a process with a low multiplicity to 32 for the LC LO 6-jet cross section. A second advantage that we showed in Sec. 1.4 is that random polarizations allow us to use real-valued operations instead complex ones, that additionally give a speedup of up to a factor of 14 (see Fig. 1.1). We did not include this number in the previous and following results since we showed in Sec. 1.4.4 that one can also perform spin-summation with the real-valued Feynman rules.

Another factor that can speed up our calculations is the use of single-precision numbers, although the advantage decreases with the number of particles, as explained in Sec. 1.2.1.1.

Finally we presented subtraction terms in Sec. 1.5.4 for the use in real corrections with random polarizations. We found that random polarizations increased the error similar to the LO case for lower multiplicities, although for a higher number they increase the error by a factor of four. Because our implementation of the new subtraction terms is not as efficient as possible, this method did not yield a speedup. In Sec. 1.5.8.1 we argued how to improve this.

Since Monte Carlo integration is an important ingredient in every perturbative calculation we described the popular VEGAS algorithm and presented an implementation that parallelizes the integration on a computer cluster. We showed that,

given a reasonably large problem, the speedup gets very close to the optimal one.

APPENDIX A

Appendix

A.1 Proofs

This section gathers all proofs that were too long to be included in the main text.

A.1.1 Proof of the Collinear Phase Space Jacobian

Proof of Eq. (1.75). We use the Sudakov parametrization and choose

$$\vec{p} = |\vec{p}| \hat{e}_3 \tag{A.1a}$$

$$k^0 = 0 \tag{A.1b}$$

$$\vec{k} = |\vec{k}| (\cos \phi \hat{e}_1 + \sin \phi \hat{e}_2) \tag{A.1c}$$

$$\vec{n} = \frac{\vec{p} \times \vec{k}}{|\vec{p}| |\vec{k}|} = -\sin \phi \hat{e}_1 + \cos \phi \hat{e}_2 \tag{A.1d}$$

so that $(\vec{p}, \vec{k}, \vec{n})$ form a orthogonal coordinate system where we have chosen the collinear axis to lie along \hat{e}_3 . In the following we abbreviate $k = |\vec{k}|$. It follows that

$$\vec{n} = \frac{1}{k} \frac{\partial \vec{k}}{\partial \phi} \tag{A.2a}$$

$$\frac{\partial \vec{n}}{\partial \phi} = \frac{\vec{k}}{k} \tag{A.2b}$$

$$n^0 = 1 \tag{A.2c}$$

$$p \cdot n = p^0 n^0 - \vec{p} \cdot \vec{n} = |\vec{p}| \tag{A.2d}$$

$$\frac{n^\mu}{p \cdot n} = \frac{1}{|\vec{p}|} (1, \vec{n})^T. \tag{A.2e}$$

The energy of the particle j is

$$E_j = (1 - z)p^0 + \frac{k^2}{1 - z} \frac{n^0}{2p \cdot n} = (1 - z)|\vec{p}| + \frac{k^2}{1 - z} \frac{1}{2|\vec{p}|}. \tag{A.3}$$

To compute the Jacobian we first compute the partial derivatives

$$\frac{\partial \vec{p}_j}{\partial z} = -|\vec{p}| \hat{e}_3 + \frac{1}{2|\vec{p}|(1-z)^2} (-\sin \phi \hat{e}_1 + \cos \phi \hat{e}_2) \quad (\text{A.4a})$$

$$\frac{\partial \vec{p}_j}{\partial \phi} = k \left(-\left(\sin \phi + \frac{1}{1-z} \frac{\cos \phi}{2|\vec{p}|} \right) \hat{e}_1 + \left(\cos \phi - \frac{1}{1-z} \frac{\sin \phi}{2|\vec{p}|} \right) \hat{e}_2 \right) \quad (\text{A.4b})$$

$$\frac{\partial \vec{p}_j}{\partial k} = -\left(\frac{2k}{1-z} \frac{\sin \phi}{2|\vec{p}|} + \cos \phi \right) \hat{e}_1 + \left(\frac{2k}{1-z} \frac{\cos \phi}{2|\vec{p}|} - \sin \phi \right) \hat{e}_2 \quad (\text{A.4c})$$

so that

$$\begin{aligned} \det \left(\frac{\partial \vec{p}_j}{\partial z}, \frac{\partial \vec{p}_j}{\partial \phi}, \frac{\partial \vec{p}_j}{\partial k} \right) &= \frac{\partial (\vec{p}_j)_3}{\partial z} \left(\frac{\partial (\vec{p}_j)_1}{\partial \phi} \frac{\partial (\vec{p}_j)_2}{\partial k} - \frac{\partial (\vec{p}_j)_2}{\partial \phi} \frac{\partial (\vec{p}_j)_1}{\partial k} \right) \\ &= -|\vec{p}| k \left(1 - \frac{2k}{(1-z)^2} \frac{1}{4|\vec{p}|^2} \right). \end{aligned} \quad (\text{A.5})$$

Together we have

$$\frac{1}{2E_j} \left| \det \left(\frac{\partial \vec{p}_j}{\partial z}, \frac{\partial \vec{p}_j}{\partial \phi}, \frac{\partial \vec{p}_j}{\partial k} \right) \right| = \frac{k}{1-z} \frac{|\vec{p}| - \frac{k^2}{(1-z)^2} \frac{1}{2|\vec{p}|}}{|\vec{p}| + \frac{k^2}{(1-z)^2} \frac{1}{2|\vec{p}|}} = \frac{k}{1-z} + \mathcal{O}(k^2). \quad (\text{A.6})$$

A.1.2 Proof for the Soft Case for Quarks

Proof of Eq. (1.88). Let us first consider the case where the polarization $\bar{u}(p_i)$ is a helicity-/spin eigenstate, i.e. $u(p_i) = u^{\lambda_i}(p_i)$ with $\lambda = \pm$. Then Eq. (1.88) reads

$$\sum_{\lambda=\pm} \left[\bar{u}_\alpha^{\lambda_i}(p_i) \gamma_{\alpha\beta}^\nu u_\beta^\lambda(p_i) \right] \varepsilon_\nu^*(p_j) \mathbf{T}_i \left(\bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma \right). \quad (\text{A.7})$$

The content of the square-brackets is $2p_i^\nu \delta_{\lambda\lambda_i}$ so Eq. (1.88) has been proven.

In the case of random polarizations, $\bar{u}(p_i) = \bar{u}(p_i, \phi)$ we use the definition to write in terms of helicity-/spin-eigenstates:

$$\sum_{\lambda=\pm} \left(\left[e^{-i\phi} \bar{u}_\alpha^+(p_i) + e^{+i\phi} \bar{u}_\alpha^-(p_i) \right] \gamma_{\alpha\beta}^\nu u_\beta^\lambda(p_i) \right) \varepsilon_\nu^*(p_j) \mathbf{T}_i \left(\bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma \right). \quad (\text{A.8})$$

We use the same identity as in the first case which selects one term in the inner brace and becomes

$$\begin{aligned} &\sum_{\lambda=\pm} \left(\left[e^{-\lambda\phi} \bar{u}_\alpha^+(p_i) \right] \gamma_{\alpha\beta}^\nu u_\beta^\lambda(p_i) \right) \varepsilon_\nu^*(p_j) \mathbf{T}_i \left(\bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma \right) \\ &= \sum_{\lambda=\pm} \left(e^{-\lambda\phi} 2p_i^\nu \right) \varepsilon_\nu^*(p_j) \mathbf{T}_i \left(\bar{u}_\gamma^\lambda(p_i) \mathcal{A}'_\gamma \right) \\ &= 2p_i \cdot \varepsilon^*(p_j) \mathbf{T}_i \left[\sum_{\lambda=\pm} e^{-\lambda\phi} \bar{u}_\gamma^\lambda(p_i) \right] \mathcal{A}'_\gamma. \end{aligned} \quad (\text{A.9})$$

Since the term in the square brackets is the same random polarization $\bar{u}(p_i, \phi)$ Eq. (1.88) hold true. \blacksquare

A.1.3 Proof of the Soft Function

Proof of Eq. (1.95). We start with the expression of the original soft function given in Eq. (1.93) that is obtained by simply squaring the eikonal current:

$$\sum_{i=1}^n \sum_{k=1}^n \mathbf{T}_i \cdot \mathbf{T}_k \frac{p_i \cdot \varepsilon_j p_k \cdot \varepsilon_j^*}{p_i \cdot p_j p_k \cdot p_j} \quad (\text{A.10})$$

We also make use of color conservation

$$\sum_{i=1}^n \sum_{k=1}^n \mathbf{T}_i \cdot \mathbf{T}_k = 0 \quad \Leftrightarrow \quad \mathbf{T}_i^2 = - \sum_{i=1}^n \sum_{k \neq i}^n \mathbf{T}_i \cdot \mathbf{T}_k \quad (\text{A.11})$$

to rewrite the sum into

$$\begin{aligned} & \sum_{i=1}^n \left(\mathbf{T}_i^2 \frac{(p_i \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)^2} + \sum_{k \neq i}^n \mathbf{T}_i \cdot \mathbf{T}_k \frac{p_i \cdot \varepsilon_j p_k \cdot \varepsilon_j^*}{p_i \cdot p_j p_k \cdot p_j} \right) \\ &= - \sum_{i=1}^n \sum_{k \neq i}^n \mathbf{T}_i \cdot \mathbf{T}_k \left(\frac{(p_i \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)^2} - \frac{p_i \cdot \varepsilon_j p_k \cdot \varepsilon_j^*}{p_i \cdot p_j p_k \cdot p_j} \right). \end{aligned} \quad (\text{A.12})$$

We now split the sum $\sum_{k \neq i}$ into two sums $\sum_{k < i} + \sum_{k > i}$ and make use of the following identity that disentangles the poles

$$\frac{1}{(p_i \cdot p_j)(p_k \cdot p_j)} = \frac{1}{(p_i \cdot p_j)(p_i \cdot p_j + p_k \cdot p_j)} + \frac{1}{(p_k \cdot p_j)(p_k \cdot p_j + p_i \cdot p_j)}, \quad (\text{A.13})$$

leading to

$$\begin{aligned} & \sum_{i=1}^n \sum_{i < k}^n \mathbf{T}_i \cdot \mathbf{T}_k \frac{(p_i \cdot \varepsilon_j)(p_k \cdot \varepsilon_j^*)}{(p_i \cdot p_j)(p_i \cdot p_j + p_k \cdot p_j)} + \frac{(p_i \cdot \varepsilon_j)(p_k \cdot \varepsilon_j^*)}{(p_k \cdot p_j)(p_k \cdot p_j + p_i \cdot p_j)} + \\ & \sum_{k=1}^n \sum_{k > i}^n \mathbf{T}_k \cdot \mathbf{T}_i \frac{(p_k \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_k \cdot p_j)(p_k \cdot p_j + p_i \cdot p_j)} + \frac{(p_k \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)(p_i \cdot p_j + p_k \cdot p_j)} \end{aligned} \quad (\text{A.14})$$

where the labels k and i were interchanged in the second line. Because the summations in both lines are equal (both over all pair with $k > i$) we can interchange the second terms from both lines and rename the indices in the second line again and finally write it as one sum:

$$\sum_{i=1}^n \sum_{i \neq k}^n \mathbf{T}_i \cdot \mathbf{T}_k \frac{(p_i \cdot \varepsilon_j)(p_k \cdot \varepsilon_j^*) + (p_k \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)(p_i \cdot p_j + p_k \cdot p_j)}. \quad (\text{A.15})$$

Inserted back into Eq. (A.12) we finally obtain Eq. (1.95)

$$- \sum_{i=1}^n \sum_{k \neq i}^n \mathbf{T}_i \cdot \mathbf{T}_k \left(\frac{(p_i \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)^2} - \frac{(p_i \cdot \varepsilon_j)(p_k \cdot \varepsilon_j^*) + (p_k \cdot \varepsilon_j)(p_i \cdot \varepsilon_j^*)}{(p_i \cdot p_j)(p_i \cdot p_j + p_k \cdot p_j)} \right). \quad (\text{A.16}) \quad \blacksquare$$

A.1.4 The Soft Function for Helicity Eigenstates

Here we will give the calculation for the soft function S summed over the helicity eigenstates of the soft particle j . In this case the soft function is

$$S = - \sum_{\lambda_j = \pm} \frac{(p_i \cdot \varepsilon_j^*)(p_k \cdot \varepsilon_j) + (p_i \cdot \varepsilon_j)(p_k \cdot \varepsilon_j^*)}{(p_i \cdot p_j)(p_i \cdot p_j + p_j \cdot p_k)}. \quad (\text{A.17})$$

Performing the polarization sum and going over to invariants we obtain

$$S = \frac{4s_{ik}}{s_{ij}(s_{ij} + s_{jk})} - \frac{4}{s_{ij} + s_{jk}} p_k \cdot Q_j - \frac{4s_{jk}}{s_{ij}(s_{ij} + s_{jk})} p_i \cdot Q_j \quad (\text{A.18})$$

Using the Catani-Seymour variables y , z , and P we obtain

$$S = \frac{1}{P^2} \left(\frac{4}{y} \frac{z(1-y)}{1-z(1-y)} - \frac{4}{1-z(1-y)} (p_k - p_i) \cdot Q_j - \frac{4}{y} p_i \cdot Q_j \right) \quad (\text{A.19})$$

A.1.5 The Splitting Functions for the Collinear Limit

Eq. (1.100a) and Eq. (1.100c) are simply the three-particle Feynman diagrams with external momenta p_{ij}, p_i, p_j and Feynman rules from App. A.4.2. Eq. (1.100b) uses the three-gluon-vertex with $p_1 = -p_{ij} = -(p_i + p_j)$, $p_2 = p_j$, and $p_3 = p_i$ and setting particles i and j incoming (therefore also the minus-sign for the momenta), particle ij outgoing. This gives the following expression for the diagram A :

$$\begin{aligned} A &= \varepsilon_{ij}^* \cdot \varepsilon_j (p_{ij} + p_j) \cdot \varepsilon_i + \varepsilon_j \cdot \varepsilon_i (-p_j + p_i) \cdot \varepsilon_{ij}^* + \varepsilon_i \cdot \varepsilon_{ij}^* (-p_i - p_{ij}) \cdot \varepsilon_j \\ &= \varepsilon_{ij}^* \cdot \varepsilon_j 2p_j \cdot \varepsilon_i + \varepsilon_j \cdot \varepsilon_i 2p_i \cdot \varepsilon_{ij}^* + \varepsilon_i \cdot \varepsilon_{ij}^* (-2p_i) \cdot \varepsilon_j, \end{aligned} \quad (\text{A.20})$$

where we used $p_{ij} = p_i + p_j$ and the transversality property of the polarization vectors: $\varepsilon(p) \cdot p = 0$.

A.1.6 Catani Seymour Momenta in the Soft and Collinear Limits

The case of two collinear particles i and j and/or a soft particle j can be summarized with the limit of the invariant $s_{ij} \rightarrow 0$. In this limit the factors containing y behave as

$$\frac{1}{1-y} = \frac{s_{ij} + s_{ik} + s_{jk}}{s_{ik} + s_{jk}} = 1 + \frac{s_{ij}}{s_{ik} + s_{jk}} \xrightarrow{s_{ij} \rightarrow 0} 1, \quad (\text{A.21})$$

$$\frac{y}{1-y} = \frac{s_{ij}}{s_{ij} + s_{ik} + s_{jk}} \frac{s_{ij} + s_{ik} + s_{jk}}{s_{ik} + s_{jk}} = \frac{s_{ij}}{s_{ik} + s_{jk}} \xrightarrow{s_{ij} \rightarrow 0} 0. \quad (\text{A.22})$$

Proof of Eq. (1.107a) and Eq. (1.108a). From Eq. (A.21) immediately follows that

$$\tilde{p}_k = \frac{1}{1-y} p_k \xrightarrow{s_{ij} \rightarrow 0} p_k. \quad (\text{A.23})$$

Proof of Eq (1.107b) and Eq. (1.108b). Using Eq. (A.22) we can show that

$$\tilde{p}_{ij} = p_i + p_j + \frac{y}{1-y} p_k \xrightarrow{s_{ij} \rightarrow 0} p_i + p_j \xrightarrow{p_j \rightarrow 0} p_i. \quad (\text{A.24})$$

A.1.7 The Soft Reparametrization

Proof of Eq. (1.111a). Let us first assume that $h_i = \lambda_i = \pm$, i.e. the spinor is a helicity eigenstate. In that case it is obvious that Eq. (1.111a) is proportional to

$$S_{\lambda\lambda'} \propto \delta_{\lambda\lambda_i} \delta_{\lambda_i\lambda'}. \quad (\text{A.25})$$

The non-vanishing part is

$$\frac{1}{2p_i \cdot q_{ij}} \frac{1}{2p_{ij} \cdot q_{ij}} \frac{1}{2} \left(\text{tr} \left(\not{p}_i \not{q}_{ij} \not{p}_{ij} \not{q}_{ij} \right) + \lambda \text{tr} \left(\gamma_5 \not{p}_i \not{q}_{ij} \not{p}_{ij} \not{q}_{ij} \right) \right) = 1 \quad (\text{A.26})$$

because the second trace is zero since $\epsilon_{\mu\nu\dots} q_{ij}^\mu q_{ij}^\nu = 0$ and the first trace $8(p_i \cdot q_{ij})(p_{ij} \cdot q_{ij})$ cancels the denominators. Therefore Eq. (A.25) is exact and we have shown that Eq. (1.110) holds true.

We now discuss the remaining case where $h_i = \phi_i$, i.e. the spinor is randomly polarized. In that case we expand them in helicity eigenstates,

$$\begin{aligned} u_{\phi_i} \bar{u}_{\phi_i} &= \sum_{\lambda_i = \pm} u_{\lambda_i} \bar{u}_{\lambda_i} + e^{2i\phi_i} u_+ \bar{u}_- + e^{-2i\phi_i} u_- \bar{u}_+ \\ &= \not{p}_i + \sum_{\lambda_i = \pm} e^{2i\lambda_i \phi_i} u_{\lambda_i}(p_i) \bar{u}_{-\lambda_i}(p_i). \end{aligned} \quad (\text{A.27})$$

The first term is the part that was already calculated, i.e. $S_{++} = S_{--} = 1$ and does not contribute to S_{+-} or S_{-+} , the second part only contributes for S_{+-} and S_{-+} :

$$S_{+-} = \frac{1}{2p_i \cdot q_{ij}} \frac{1}{2p_{ij} \cdot q_{ij}} \bar{u}_+(p_{ij}) \not{q}_{ij} u_+(p_i) \bar{u}_-(p_i) \not{q}_{ij} u_-(p_{ij}) \quad (\text{A.28})$$

The first part is just the case discussed above, with $\lambda_i = +$ covered by $\lambda = \lambda' = +$ and $\lambda_i = -$ covered by $\lambda = \lambda' = -$. The remaining cases $\lambda \neq \lambda'$ are summarized by

$$S_{\lambda\lambda'} = e^{2\lambda i \phi_i} \frac{1}{2p_i \cdot q_{ij}} \frac{1}{2p_{ij} \cdot q_{ij}} \text{tr} \left(\not{q}_{ij} \frac{\lambda}{\sqrt{2}} P_\lambda \not{p}_i \not{p}_i^{\lambda'} \not{q}_{ij} \frac{\lambda'}{\sqrt{2}} P_{\lambda'} \not{p}_{ij} \not{p}_{ij}^{\lambda} \right). \quad (\text{A.29})$$

where we used Eq. (A.49). The trace evaluates to

$$4(p_i \cdot q_{ij})(p_{ij} \cdot q_{ij}) \left(-\varepsilon^{\lambda'}(p_i) \cdot \varepsilon^\lambda(p_{ij}) \right) + (p_{ij} \cdot q_{ij}) \lambda \text{tr} \left(\gamma_5 \not{p}_i \not{p}_i^{\lambda'} \not{q}_{ij} \not{p}_{ij}^{\lambda} \right). \quad (\text{A.30})$$

The first term contains the expression $\varepsilon^{\lambda'}(p_i) \cdot \varepsilon^{\lambda}(p_{ij})$ which is Eq. (A.57) in the soft limit. The second trace is proportional to

$$\epsilon_{\mu\nu\rho\sigma} p_i^\mu \varepsilon_i^\nu q^\rho \varepsilon_{ij}^\sigma \quad (\text{A.31})$$

and finite in the soft limit. ■

Proof of Eq. (1.111b). If particle i is a helicity eigenstate, then we have in the soft limit

$$S_{\lambda\lambda'} = \varepsilon_\lambda(p_{ij}) \cdot \varepsilon_{\lambda_i}^*(p_i) \varepsilon_{\lambda_i}(p_i) \cdot \varepsilon_{\lambda'}^*(p_{ij}) \xrightarrow{p_{ij} \rightarrow p_i} \delta_{\lambda\lambda_i} \delta_{\lambda_i\lambda'} \quad (\text{A.32})$$

where we made use of Eq. (A.57). If particle i is randomly polarized with angle ϕ_i we can decompose it into helicity eigenstates:

$$S_{\lambda\lambda'} = \varepsilon_\lambda(p_{ij}) \cdot \left(e^{i\phi_i} \varepsilon_+(p_i) + e^{-i\phi_i} \varepsilon_-(p_i) \right)^* \left(e^{i\phi_i} \varepsilon_+(p_i) + e^{-i\phi_i} \varepsilon_-(p_i) \right) \cdot \varepsilon_{\lambda'}^*(p_{ij}) \xrightarrow{p_{ij} \rightarrow p_i} e^{(-\lambda+\lambda')i\phi_i} \quad (\text{A.33})$$

which is 1 in case $\lambda = \lambda'$ and gives the helicity dependent terms if $\lambda \neq \lambda'$. ■

A.1.8 Proof of the Spin-Correlation Tensor Transversality

Proof of Eq. (1.132). If $S^{\mu\nu} = g^{\mu\nu}$ then Eq. (1.132) reads $\tilde{p}_{ij}^2 = 0$ and is shown because \tilde{p}_{ij} is lightlike. If $S^{\mu\nu} = v^\mu v^\nu$ with $v^\mu = z_i p_i^\mu - z_j p_j^\mu$ then we can, using the identities in Eq. (1.121), also show that

$$\begin{aligned} 2\tilde{p}_{ij} \cdot v &= 2z\tilde{p}_{ij} \cdot p_i - 2(1-z)\tilde{p}_{ij} \cdot p_j \\ &= z(1-z)yP^2 - (1-z)(1-(1-z))yP^2 = 0. \end{aligned} \quad (\text{A.34})$$

Because the spin-correlation tensor is, in general, a linear combination of those two vectors the property still holds. ■

A.1.9 Proof of the Integrability of the Additional Term

Proof that $\frac{2z+2(z-1)p_i \cdot Q_j}{y} \xrightarrow{k \rightarrow 0} \mathcal{O}\left(\frac{1}{k}\right)$. We start with Eq. (1.71), the Sudakov parametrization, to show that

$$\begin{aligned} p_i \cdot Q_j &= \frac{\tilde{z} p \cdot q_j + k \cdot q_j}{(1-\tilde{z})p \cdot q_j - k \cdot q_j} + \mathcal{O}(k^2) \\ &= \frac{\tilde{z}}{1-\tilde{z}} \left(1 + \frac{1}{\tilde{z}(1-\tilde{z})} \frac{k \cdot q_j}{p \cdot q_j} \right) + \mathcal{O}(k^2) \end{aligned} \quad (\text{A.35})$$

with \tilde{z} the parameter from the Sudakov parametrization and z the variable that we use in the context of Catani-Seymour dipoles, see Eq. (1.119). Both are related in the collinear limit according to

$$z = \frac{2\tilde{z}p \cdot p_k + 2k \cdot p_k + \mathcal{O}(k^2)}{2p \cdot p_k + \mathcal{O}(k^2)} = \tilde{z} + \frac{k \cdot p_k}{p \cdot p_k} + \mathcal{O}(k^2), \quad (\text{A.36})$$

Finally we can show that

$$\begin{aligned} \frac{2z + 2(z-1)p_i \cdot Q_j}{y} &= \frac{2z + 2(z-1)\frac{\tilde{z}}{1-\tilde{z}} + \mathcal{O}(k)}{y} \\ &= \frac{2z + 2(z-1)\frac{z}{1-z} + \mathcal{O}(k)}{y} = \frac{\mathcal{O}(k)}{y} \xrightarrow{k \rightarrow 0} \mathcal{O}\left(\frac{1}{k}\right) \end{aligned} \quad (\text{A.37})$$

■

A.1.10 Extremal PDF for Importance Sampling

Proof of Eq. (2.16). To derive the optimal PDF $p(\vec{x})$ for importance sampling we write down the functional

$$J[p] = \lambda \left[-1 + \int d^d x' p(\vec{x}') \right] + \int d^d x' p(\vec{x}') \left(\frac{f(\vec{x}')}{p(\vec{x}')} - I \right)^2 \quad (\text{A.38})$$

that uses a Lagrange multiplier taking care of the constraint that p be normalized. Applying the Euler-Lagrange equation $\frac{\partial}{\partial p} J[p] = 0$ (there is no dependence on a derivative of p) yields

$$\begin{aligned} \lambda + \left(\frac{f(\vec{x})}{p(\vec{x})} - I \right)^2 + 2p(\vec{x}) \left(\frac{f(\vec{x})}{p(\vec{x})} - I \right) \left(-\frac{f(\vec{x})}{p^2(\vec{x})} \right) &= 0 \quad \Leftrightarrow \\ \lambda + I^2 - \left(\frac{f(\vec{x})}{p(\vec{x})} \right)^2 &= 0 \quad \Rightarrow \quad p(\vec{x}) = \frac{|f(\vec{x})|}{\sqrt{I^2 + \lambda}}. \end{aligned} \quad (\text{A.39})$$

because p has to be positive in order to be a PDF. Inserting this back into the normalizing constraint $\int d^d x' p(\vec{x}') = 1$ yields

$$\lambda = \left(\int d^d x' |f(\vec{x}')| \right)^2 - I^2 \quad (\text{A.40})$$

and gives with Eq. (A.39) the final result for an extremal variance:

$$p(\vec{x}) = \frac{|f(\vec{x})|}{\int d^d x' |f(\vec{x}')|}. \quad (\text{A.41})$$

■

Proof of Eq. (2.23). If the PDF is separable, the optimal PDF instead is derived as follows:

$$J = \sum_{j=1}^d \lambda_j \left[-1 + \int dx'_j p_j(x'_j) \right] + \int \left(\prod_{k=1}^d dx'_k \right) \frac{f^2(\vec{x}')}{\prod_{j=1}^d p_j(x'_j)} \quad (\text{A.42})$$

$$\frac{\partial}{\partial p_i} J[p_1, p_2, \dots, p_d] = 0$$

$$\lambda_i - \frac{1}{p_i^2(x_i)} \int \left(\prod_{k \neq i}^d dx'_k \right) \frac{f^2(\vec{x}'|_{x'_i=x_i})}{\prod_{j \neq i}^d p_j(x'_j)} = 0 \quad \Rightarrow$$

$$p_i(x_i) = \frac{1}{\sqrt{\lambda}} \left[\int \left(\prod_{k \neq i}^d dx'_k \right) \frac{f^2(\vec{x}'|_{x'_i=x_i})}{\prod_{j \neq i}^d p_j(x'_j)} \right]^{\frac{1}{2}} \quad (\text{A.43})$$

so that

$$p_i(x_i) = \frac{\bar{f}(x_i)}{\int dx'_i \bar{f}(x'_i)} \quad \bar{f}^2(x_i) = \int \left(\prod_{k \neq i}^d dx'_k \right) \frac{f^2(\vec{x}'|_{x'_i=x_i})}{\prod_{j \neq i}^d p_j(x'_j)} \quad (\text{A.44})$$

A.2 Identities

A.2.1 Spinor Identities

The *Chisholm-identity*[50] reads

$$\bar{u}_\lambda(q) \gamma^\mu u_{\lambda'}(p) \gamma_\mu = 2u_{\lambda'}(p) \bar{u}_\lambda(q) + 2u_{-\lambda}(q) \bar{u}_{-\lambda'}(p), \quad (\text{A.45})$$

which is in the case of $p = q$ similar to the following identity

$$\bar{u}_\lambda \gamma^\mu u_{\lambda'} = 2p^\mu \delta_{\lambda\lambda'}. \quad (\text{A.46})$$

The “line-reversal trick”[50] is the identity

$$\bar{u}_\lambda(p) \Gamma u_{\lambda'}(q) = \lambda \lambda' \bar{u}_{-\lambda'}(q) \Gamma^R u_{-\lambda}(p) \quad (\text{A.47})$$

where Γ is an arbitrary string of gamma matrices and Γ^R the string where the order of the matrices are reversed.

Furthermore we have the identity

$$\bar{u}_\lambda(p) u_\lambda(p) = P_\lambda \not{p} \quad (\text{A.48})$$

with projection operator $P_\lambda = \frac{1}{2}(1 + \lambda \gamma_5)$. When calculating matrix elements with random polarizations expressions with mixed helicity states arise, for which we have the identity

$$\bar{u}_\lambda(p) u_{\lambda'}(p) = \frac{\lambda}{\sqrt{2}} P_\lambda \not{p} \not{\lambda'}(p, q) \quad (\text{A.49})$$

if $\lambda \neq \lambda'$.

Proof. We first note that a slashed polarization vector can be rewritten using the parametrization (A.53) and the Chisholm-identity (A.45) to

$$\not{\epsilon}_+ = \frac{\gamma^\mu \langle q- | \gamma_\mu | p- \rangle}{\sqrt{2} \langle q- | p+ \rangle} = \sqrt{2} \frac{|p-\rangle \langle q- | + |q+\rangle \langle p+ |}{\langle q- | p+ \rangle}, \quad (\text{A.50a})$$

$$\not{\epsilon}_- = \frac{\gamma^\mu \langle q+ | \gamma_\mu | p+ \rangle}{\sqrt{2} \langle p+ | q- \rangle} = \sqrt{2} \frac{|p+\rangle \langle q+ | + |q-\rangle \langle p- |}{\langle p+ | q- \rangle}. \quad (\text{A.50b})$$

Using the right-hand side of Eq. (A.49) for the special case $\lambda = +$, $\lambda' = -$ we obtain

$$\frac{1}{\sqrt{2}} P_+ \not{\epsilon}_- (p, q) = \frac{1}{\sqrt{2}} |p+\rangle \langle p+ | \sqrt{2} \frac{|p+\rangle \langle q+ | + |q-\rangle \langle p- |}{\langle p+ | q- \rangle} = |p+\rangle \langle p-|. \quad (\text{A.51})$$

In the case $\lambda = -$, $\lambda' = +$ we have

$$\frac{-1}{\sqrt{2}} P_- \not{\epsilon}_+ (p, q) = - \frac{\langle p- | q+ \rangle}{\langle q- | p+ \rangle} |p-\rangle \langle p+| \quad (\text{A.52})$$

where we additionally need the line reversal trick (A.47) to cancel the sign. \blacksquare

A.2.2 Polarization Vector Parametrization and Identities

We parametrize the polarization vector as

$$\varepsilon_+^\mu(p, q) = \frac{\langle q- | \gamma^\mu | p- \rangle}{\sqrt{2} \langle q- | p+ \rangle} \quad \text{and} \quad \varepsilon_-^\mu(p, q) = \frac{\langle q+ | \gamma^\mu | p+ \rangle}{\sqrt{2} \langle p+ | q- \rangle} \quad (\text{A.53})$$

with *reference momentum* $q \neq p$. They satisfy the polarization sum identity

$$\sum_{\lambda=\pm} \varepsilon_\lambda^{\mu*} \varepsilon_\lambda^\nu = -g^{\mu\nu} + \frac{p^\mu q^\nu + q^\mu p^\nu}{p \cdot q} \quad (\text{A.54})$$

and are transverse both to momentum as well as to their reference momentum,

$$p \cdot \varepsilon_\lambda(p, q) = 0 \quad q \cdot \varepsilon_\lambda(p, q) = 0. \quad (\text{A.55})$$

Furthermore, in this parametrization with Eq. (A.47) one can show that

$$[\varepsilon_-^\mu(p, q)]^* = \frac{\langle p+ | \gamma^\mu | q+ \rangle}{\sqrt{2} \langle q- | p+ \rangle} = \frac{\langle q- | \gamma^\mu | p- \rangle}{\sqrt{2} \langle q- | p+ \rangle} = \varepsilon_+^\mu(p, q) \quad (\text{A.56})$$

as required in Sec. 1.3.1. A useful identity is

$$\varepsilon_\lambda(p, q) \cdot \varepsilon_{\lambda'}(p, q) = (\delta_{\lambda\lambda'} - 1). \quad (\text{A.57})$$

Proof. We prove the case $\lambda = +$, $\lambda' = -$:

$$\varepsilon_+(p, q) \cdot \varepsilon_-(p, q) = \frac{\langle q- | \gamma_\mu \langle q+ | \gamma^\mu | p+ \rangle | p- \rangle}{2 \langle q- | p+ \rangle \langle p+ | q- \rangle} = \frac{\langle q- | p+ \rangle \langle q+ | p- \rangle}{\langle q- | p+ \rangle \langle p+ | q- \rangle} = -1. \quad (\text{A.58})$$

If $\lambda = \lambda' = -$ we obtain a similar calculation that gives zero because of $\langle q+ | q- \rangle = \langle p- | p+ \rangle = 0$. The remaining case $\lambda = \lambda' = +$ follows because of Eq. (A.56). \blacksquare

A.3 Majorana- and Weyl-Representations

We start from a Weyl-representation of the gamma matrices that can be written in a compact form as

$$\gamma^\mu = \begin{pmatrix} 0 & \sigma^\mu \\ \bar{\sigma}^\mu & 0 \end{pmatrix} \quad \gamma_5 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (\text{A.59})$$

with $\sigma^\mu = (-1, \vec{\sigma})$ and $\bar{\sigma}^\mu = (-1, -\vec{\sigma})$ where $\vec{\sigma}$ denotes the vector of Pauli matrices:

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (\text{A.60})$$

A similarity transformation U maps the gamma matrices γ_{W}^μ into Majorana-gamma matrices γ_{M}^μ . For this we use the matrix $U = \check{U}$ from Eq. (6.18) in [51]:

$$U = \frac{1}{2} \begin{pmatrix} 1 + \sigma^2 & -i + i\sigma^2 \\ i - i\sigma^2 & 1 + \sigma^2 \end{pmatrix}. \quad (\text{A.61})$$

This matrix is self-inverse: $U^{-1} = U$. Applying the transformation $\gamma_{\text{M}}^\mu = U\gamma_{\text{W}}^\mu U^{-1}$ we obtain our gamma matrices as

$$\begin{aligned} \gamma^0 &= \begin{pmatrix} 0 & -\sigma_2 \\ -\sigma_2 & 0 \end{pmatrix} & \gamma^1 &= \begin{pmatrix} i\sigma_1 & 0 \\ 0 & i\sigma_1 \end{pmatrix} \\ \gamma^2 &= \begin{pmatrix} 0 & \sigma_2 \\ -\sigma_2 & 0 \end{pmatrix} & \gamma^3 &= \begin{pmatrix} i\sigma_3 & 0 \\ 0 & i\sigma_3 \end{pmatrix}, \end{aligned} \quad (\text{A.62})$$

and

$$\gamma_5 = \begin{pmatrix} \sigma^2 & 0 \\ 0 & -\sigma^2 \end{pmatrix}. \quad (\text{A.63})$$

A.3.1 Spinors in Weyl Representation

In the massless case Weyl representations are a convenient choice because chirality coincides with helicity. Therefore the helicity eigenstates have to satisfy $\gamma_5 u^\pm = \pm u^\pm$. Together with the form of the chirality operator γ_5 in Eq. (A.59) this requires that the spinors must decompose into an ‘‘upper’’ and ‘‘lower’’ spinor:

$$u_+(\vec{p}) = \begin{pmatrix} |p^+\rangle \\ 0 \end{pmatrix} \quad u_-(\vec{p}) = \begin{pmatrix} 0 \\ |p^-\rangle \end{pmatrix}. \quad (\text{A.64})$$

Up to a factor two-component spinors $|p\pm\rangle$ are easily determined using the Dirac equation:

$$\bar{\sigma}^\mu p_\mu |p+\rangle = 0 \quad \Leftrightarrow \quad |p+\rangle \propto \begin{pmatrix} -p_{\perp*} \\ p_+ \end{pmatrix} \quad (\text{A.65a})$$

$$\sigma^\mu p_\mu |p-\rangle = 0 \quad \Leftrightarrow \quad |p-\rangle \propto \begin{pmatrix} p_+ \\ p_{\perp} \end{pmatrix} \quad (\text{A.65b})$$

with

$$p_+ = p_0 + p_3 \quad p_{\perp} = p_1 + ip_2 \quad p_{\perp*} = p_1 - ip_2. \quad (\text{A.66})$$

The proportionality factor is determined using Eq. (A.46) for $\mu = 0$, $\bar{u}^\lambda \gamma_0 u^{\lambda'} = 2p_0 \delta^{\lambda\lambda'}$, which gives

$$|p+\rangle = \frac{1}{\sqrt{p_+}} \begin{pmatrix} -p_{\perp*} \\ p_+ \end{pmatrix} \quad |p-\rangle = \frac{1}{\sqrt{p_+}} \begin{pmatrix} p_+ \\ p_{\perp} \end{pmatrix}. \quad (\text{A.67})$$

If p_+ is negative, the solutions above are not uniquely determined and one has to decide for a specific phase.

A.3.2 Spinors in Majorana Representation

Using the similarity transformation matrix U from Eq. (A.61) and applying it to the Weyl spinors we obtain the same spinors in Majorana representation:

$$U \begin{pmatrix} |p+\rangle \\ 0 \end{pmatrix} = \frac{1}{2\sqrt{p_+}} \begin{pmatrix} -ip_+ - p_{\perp*} \\ p_+ - ip_{\perp*} \\ -p_+ - ip_{\perp*} \\ ip_+ - p_{\perp*} \end{pmatrix} \quad U \begin{pmatrix} 0 \\ |p-\rangle \end{pmatrix} = \frac{1}{2\sqrt{p_+}} \begin{pmatrix} -ip_+ + p_{\perp} \\ -p_+ - ip_{\perp} \\ p_+ - ip_{\perp} \\ ip_+ + p_{\perp} \end{pmatrix} \quad (\text{A.68})$$

To correct for their behavior under complex conjugation (see Eq. (1.34) in Sec. 1.3.1) we multiply with an additional phase-factor $-i$, take the absolute of p_+ in the denominator and define the result to be our final Majorana spinors:

$$u_{\pm} = \frac{1}{2} \frac{1}{\sqrt{|p_+|}} \begin{pmatrix} (p_2 - p_+) \pm ip_1 \\ -p_1 \pm i(p_2 - p_+) \\ -p_1 \pm i(p_2 + p_+) \\ (p_2 + p_+) \pm ip_1 \end{pmatrix}. \quad (\text{A.69})$$

Note that for $p_+ < 0$ Eq. (A.69) assumes a different phase convention, because by taking the absolute we effectively dropped a factor $\sqrt{-1} = \pm i$. However, this is not a problem because the case $p_+ < 0$ characterizes a momentum of an incoming

particle: Because of $p^2 = 0$, we have $|p_0| \geq |p_3|$ and therefore an outgoing particle has $p_0 > 0$ implying $p_+ \geq 0$. An incoming particle has $p_0 < 0$ and therefore $p_+ < 0$. Since every particle is always either incoming or outgoing and does not change in the phase space integration the spinors do not switch between the phase conventions. If necessary one can adjust the phase by simply multiplying the final amplitude with the correct phases.

Another case one has to consider is $p_+ = 0$ which e.g. happens if the 3-axis is chosen to be the beam axis. In this case, the denominator gets singular which is a problem in numerical programs. However, we can consider this case using a parity transformation, i.e. use

$$u_{\pm}(\vec{p}) = \eta \gamma^0 u_{\mp}(-\vec{p}) \quad (\text{A.70})$$

with a phase factor η so that for the special case $p_+ = 0$ we arrive with

$$u^{\pm} = \text{sign}(p_0) \sqrt{\frac{|p_0|}{2}} \begin{pmatrix} -1 \\ \mp i \\ \pm i \\ 1 \end{pmatrix}. \quad (\text{A.71})$$

A.3.3 Randomly Polarized Spinors

We have constructed the Majorana spinors such that $(u_{\pm})^* = u_{\mp}$ which enables us to define our randomly polarized spinors as:

$$u(p, \phi) = 2\Re \left\{ e^{+i\phi} u_+(p) \right\} = \frac{1}{\sqrt{|p_+|}} \begin{pmatrix} (p_2 - p_+) \cos \phi - p_1 \sin \phi \\ -p_1 \cos \phi - (p_2 - p_+) \sin \phi \\ -p_1 \cos \phi - (p_2 + p_+) \sin \phi \\ (p_2 + p_+) \cos \phi - p_1 \sin \phi \end{pmatrix} \quad (\text{A.72})$$

A.4 Feynman Rules

This section lists the Feynman rules used in this thesis for QCD derived from $i\mathcal{L}$. All momenta and particles are supposed to be outgoing; incoming momenta therefore get an additional minus sign and fermions are converted to antifermions and vice-versa (crossing-symmetry).

The rules contain the Gell-Mann matrices T^a that are the generators of SU(3) and satisfy the commutation relations

$$[T^a, T^b] = if^{abc} T^c \quad (\text{A.73})$$

with adjoint representation indices $a, b, c \in \{1, 2, \dots, 8\}$ and the fully antisymmetric symbols f^{abc} for SU(3).

A.4.1 QCD Feynman Rules

The Feynman rules are derived from

$$\mathcal{L} = -\frac{1}{2}\partial^\mu A_\nu^a (\partial_\mu A^{a,\nu} - \partial^\nu A_\mu^a) - \frac{1}{2\xi} (\partial^\mu A_\mu^a) (\partial^\nu A_\nu^a) \quad (\text{A.74a})$$

$$+ \bar{\psi}_{i,\beta} \delta_{ij} (i\gamma_{\alpha\beta}^\mu \partial_\mu - m_i \delta_{\alpha\beta}) \psi_{j,\beta} \quad (\text{A.74b})$$

$$- \frac{g^2}{4} (f^{eab} A_\mu^a A_\nu^b) (f^{eab} A^{c,\mu} A^{d,\nu}) \quad (\text{A.74c})$$

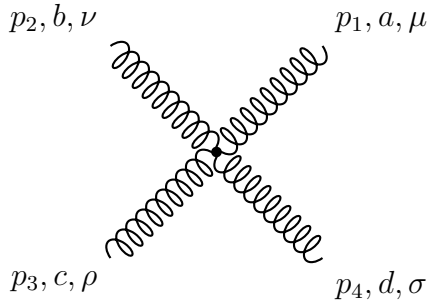
$$- g f^{abc} A_\mu^a A_\nu^b \partial_\mu A^{c,\nu} \quad (\text{A.74d})$$

$$+ g \bar{\psi}_{i,\alpha} \gamma_{\alpha\beta}^\mu A_\mu^a T_{ij}^a \psi_{j,\beta}. \quad (\text{A.74e})$$

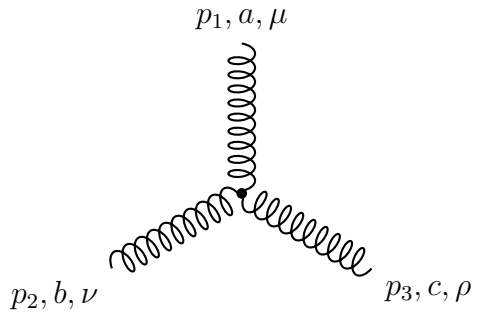
The Lagrangian and the rules derived from it agree with Peskin and Schroeder[52], Böhm, Denner, and Joos[53] and agrees with Collins[33] if one replaces $g \rightarrow -g$. The parameter ξ in the gauge-fixing term is set to $\xi = 1$, the *Feynman gauge* which is justified because in observables the dependence on ξ cancels.

$$a, \mu \text{ (wavy line)} b, \nu = -i \left[g^{\mu\nu} + (1 - \xi) \frac{p^\mu p^\nu}{p^2} \right] \frac{\delta^{ab}}{p^2 + i\epsilon} \quad (\text{A.75a})$$

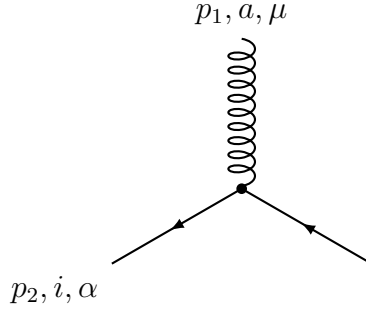
$$j, \beta \text{ (arrow)} i, \alpha = \frac{i (\not{p} + m)_{\alpha\beta} \delta_{ij}}{p^2 - m^2 + i\epsilon} \quad (\text{A.75b})$$



$$\begin{aligned}
 & -i g^2 [f^{bae} f^{cde} (g^{\nu\rho} g^{\mu\sigma} - g^{\nu\sigma} g^{\mu\rho}) \\
 & + f^{bce} f^{ade} (g^{\mu\nu} g^{\rho\sigma} - g^{\nu\sigma} g^{\mu\rho}) \\
 & + f^{bde} f^{ace} (g^{\mu\nu} g^{\rho\sigma} - g^{\nu\rho} g^{\mu\sigma})] \quad (\text{A.75c})
 \end{aligned}$$



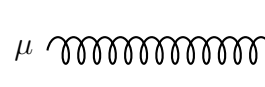
$$\begin{aligned}
 & -g f^{abc} [g^{\mu\nu} (p_1 - p_2)^\rho \\
 & + g^{\nu\rho} (p_2 - p_3)^\mu \\
 & + g^{\rho\mu} (p_3 - p_1)^\nu] \quad (\text{A.75d})
 \end{aligned}$$



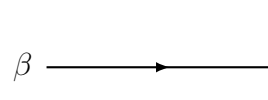
$$= ig\gamma_{\alpha\beta}^{\mu} T_{ij}^a \quad (\text{A.75e})$$

A.4.2 Color-stripped QCD Feynman Rules

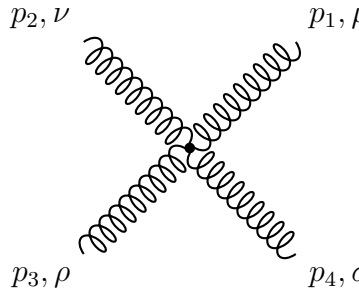
With the exception of the four-gluon vertex the color-stripped Feynman rules are very similar to the full Feynman rules:



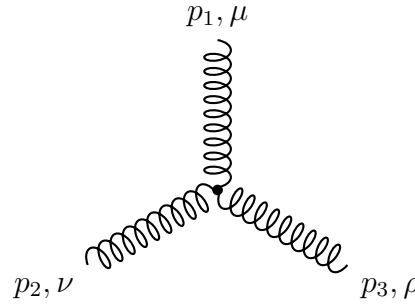
$$= -i \left[g^{\mu\nu} + (1 - \xi) \frac{p^{\mu} p^{\nu}}{p^2} \right] \frac{1}{p^2 + i\epsilon} \quad (\text{A.76a})$$



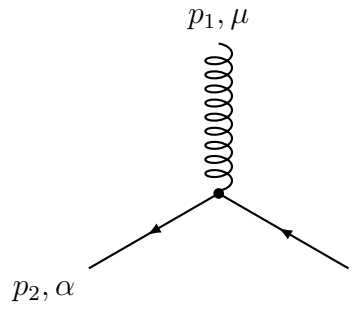
$$= \frac{i (\not{p} + m)_{\alpha\beta}}{p^2 - m^2 + i\epsilon} \quad (\text{A.76b})$$



$$= ig^2 [2g_{\mu\nu}g_{\rho\sigma} - g_{\mu\rho}g_{\nu\sigma} - g_{\mu\sigma}g_{\nu\rho}] \quad (\text{A.76c})$$



$$= -ig [g^{\mu\nu}(p_1 - p_2)^{\rho} + g^{\nu\rho}(p_2 - p_3)^{\mu} + g^{\rho\mu}(p_3 - p_1)^{\nu}] \quad (\text{A.76d})$$



$$= ig\gamma_{\alpha\beta}^{\mu} \quad (\text{A.76e})$$

A.5 Phase Space Generation

Given m random numbers collected in a vector $\vec{x} \in [0, 1]^m$, a phase space generator (PSG) is a function mapping

$$\vec{x} \longmapsto \{\vec{p}_i\}_{i=1}^n, \quad p_i^2 = m_i^2, \quad (\text{A.77})$$

i.e. it maps numbers from the m -dimensional unit-hypercube onto on-shell momenta \vec{p}_i (with masses m_i often set to zero) that additionally fulfill momentum conservation

$$\sum_i p_i = \sum_j q_j \quad (\text{A.78})$$

where q_j are the incoming momenta that are fixed by the experiment and frame the momenta are measured.

A.5.1 Introduction

From the view of a MC integrator a phase space generator converts the random numbers \vec{x} into momenta p_i that are used as the input for the matrix elements $\mathcal{A}(p_1, p_2, \dots, p_n)$. The phase space integral that the PSG thereby performs is

$$\prod_{i=1}^n \int \frac{d^3 \vec{p}_i}{E(\vec{p}_i)} \delta^{(4)} \left(\sum_{j=1}^m q_j - \sum_{i=1}^n p_i \right). \quad (\text{A.79})$$

Counting the number of free variables we obtain the relation $m \geq 3n - 4$, i.e. one needs at least $3n - 4$ random variables for phase space generation.

In this thesis we will employ several PSG, which all have their advantageous and shortcomings. This also allowed us to cross-check results. We will use

- RAMBO[54], which is very easy to implement and supports the generation of massive momenta. We need this for the generation of collinear momenta, as described in Sec. A.5.2. RAMBO needs $4n$ random variables;
- the QCD antenna generator, as described in Refs. [38, 55] for the most QCD phase space integrations. This generator is designed such that the invariants $s_{i(i+1)} = 2p_i \cdot p_{i+1}$ lie on the integration axis which leads to an approximate factorization of the matrix element (times its phase space weight). This is the reason why it works very well with the VEGAS algorithm (see Sec. 2.3). Furthermore it will be used to generate soft momenta (see Sec. A.5.2).

A.5.2 Soft- and Collinear-Momentum Generation

To ascertain if the subtraction terms are correct one can check their local behavior. To do this, however, one needs functions that generate soft and collinear momenta, i.e. produce the phase space points in regions where the subtraction terms must subtract the singular behavior.

A.5.2.1 Soft Momentum Generation

An event with a soft momentum can be generated with the QCD antenna generator (see App. A.5.1) where the first two random numbers x and y of the last three are set equal and then sent to zero. As Fig. A.1 illustrates, this gives a linear relation between the energy λ of the soft momentum

$$p = (\lambda, \vec{p}), \quad |\vec{p}| = \lambda \quad (\text{A.80})$$

and the scale (the random number) x :

$$\lambda = a \cdot x. \quad (\text{A.81})$$

The number a depends on the remaining random numbers, i.e. changes for each phase space point, which is not problematic since we are only interested in generating events with soft momenta over a linear scale.

A.5.2.2 Collinear Momentum Generation

An event with a pair of collinear momenta p_i and p_j with

$$(p_i + p_j)^2 = -\frac{k_\perp^2}{z(1-z)} = k^2 \quad (\text{A.82})$$

can be generated by

1. generating momenta with one momentum less than needed, but one massive particle with mass m . We did this by using the RAMBO algorithm for massive momenta.
2. The massive particle with momentum $k = (\sqrt{m^2 + \vec{k}^2}, \vec{k})$ is then subsequently decayed into two massless momenta p'_i and p'_j that, in the rest frame of k , are simply back-to-back. Both momenta are then boosted with transformation L into the frame of k , i.e. if $L^{-1}k = (m, 0, 0, 0)$ then $Lp'_i = p_i$ and $Lp'_j = p_j$.

The result can be seen in Fig. A.2 which shows that, by the construction above, $k = a \cdot x = x$ is equal to the collinear scale. The deviations from linearity of the left regions are caused by the finite precision of the calculations — if performed with single precision arithmetic these deviations show up earlier.

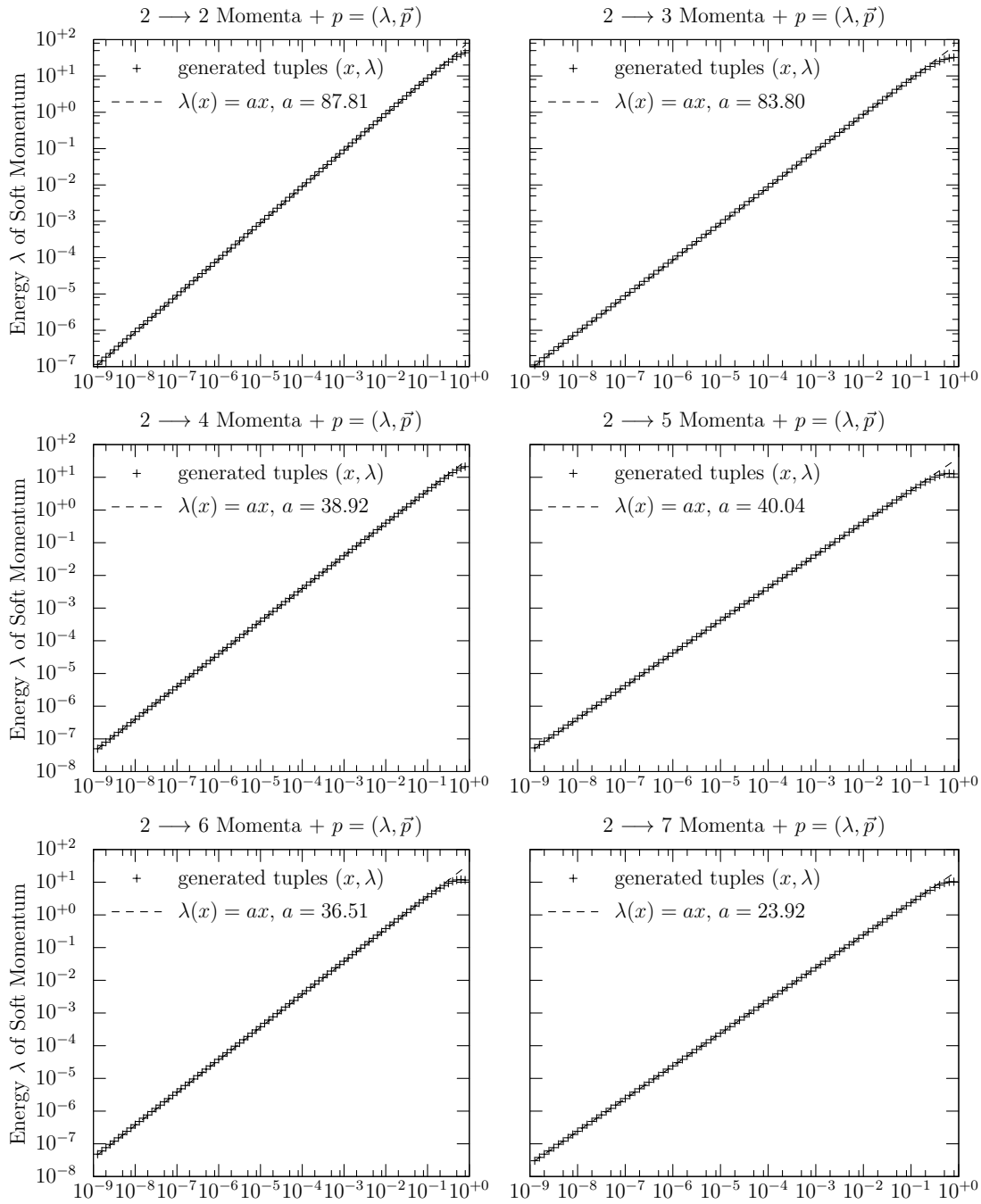


Figure A.1: *Soft momentum generation.* These plots show the generation of three to eight outgoing momenta of which one, labeled p , is soft. The plots show that there is an approximately linear behavior between the softness scale x (which corresponds to the last two random numbers that are fed into phase space generator) and the actual energy λ .

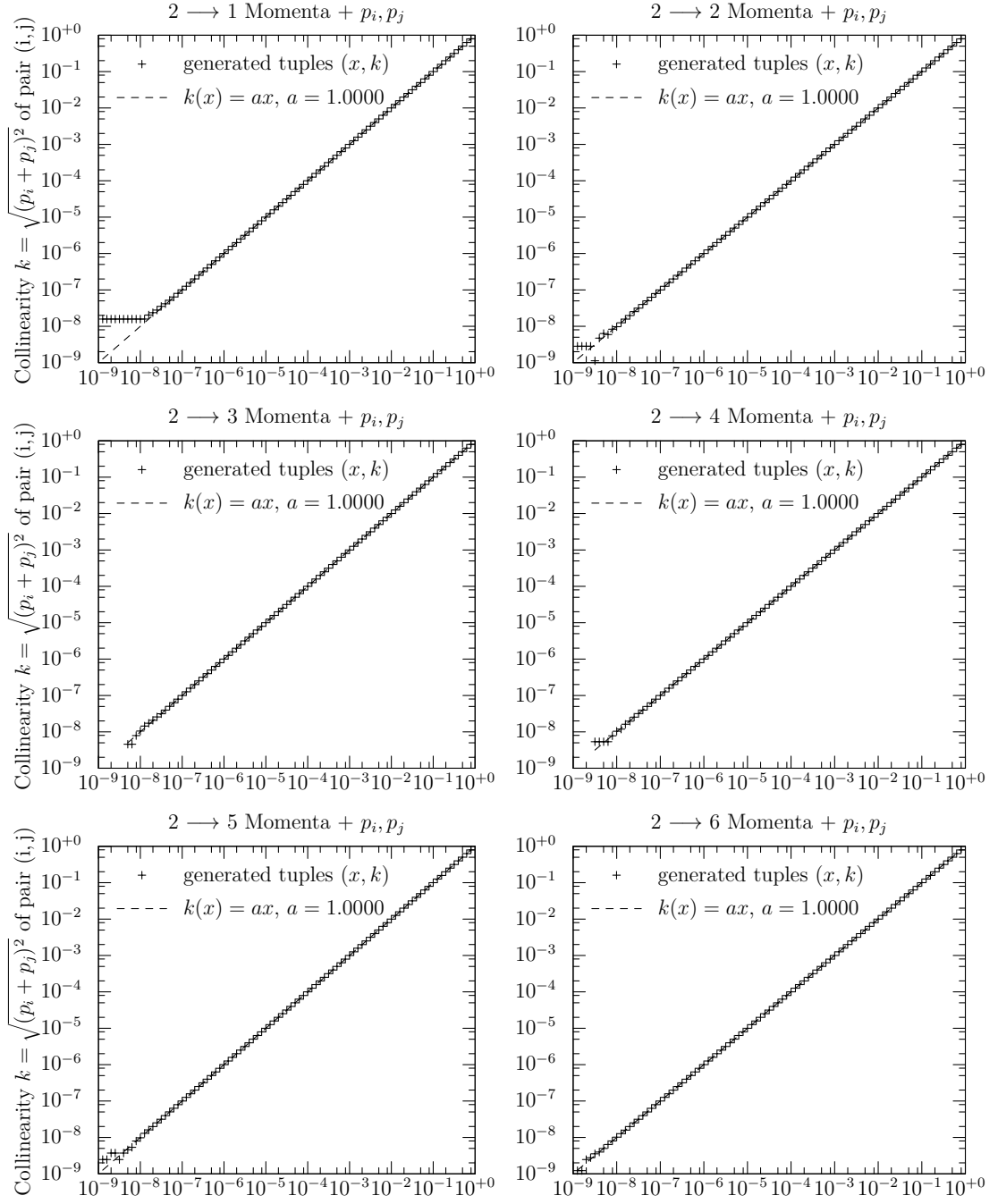


Figure A.2: Collinear momentum generation. These plots show the generation of three to eight outgoing momenta of which the last pair of particles, labeled p_i and p_j , are collinear. The plots show the scale x is identical to k . The deviations in the left regions are due to numerical problems, see text.

A.6 Technical Details

A.6.1 Instruction Counting with perf

The program `perf`[56] allows one to measure a variety of events of entire programs, such as executed number of instructions, instructions per cycle, number of branches, branch-misses, page-faults, etc. Its advantage over different profiling programs, e.g. `valgrind`[57, 58], is that `perf` uses event counters which are directly implemented on modern CPUs and thus do not interfere with execution, e.g. slow down the execution as is the case with `valgrind`. However, this also means that the executed code is only *sampled*, i.e. the results are approximate. The `perf` event framework was introduced with Linux 2.6.31 in 2009 and is available in all kernels since.

Unfortunately there is almost no documentation on the internet on how to measure the instruction count for a critical section of a program instead of the whole program. This section will show how to accomplish this.

First, one has to define the `perf_event_open` function, which is actually a system call:

```

1 #include <cstdlib>
2 #include <stdio>
3 #include <cstring>
4 #include <unistd.h>
5 #include <sys/ioctl.h>
6 #include <linux/perf_event.h>
7 #include <asm/unistd.h>
8
9 long perf_event_open(
10     perf_event_attr* hw_event,
11     pid_t pid,
12     int cpu,
13     int group_fd,
14     unsigned long flags
15 ) {
16     int ret = syscall(__NR_perf_event_open, hw_event, pid, cpu, group_fd, flags);
17     return ret;
18 }

```

Next, we have to select the events we are interested in. In this example we count hardware instructions (line 7):

```

1 perf_event_attr attr;
2
3 // select what we want to count
4 std::memset(&attr, 0, sizeof(perf_event_attr));
5 attr.size = sizeof(perf_event_attr);
6 attr.type = PERF_TYPE_HARDWARE;
7 attr.config = PERF_COUNT_HW_INSTRUCTIONS;
8 attr.disabled = 1;
9 attr.exclude_kernel = 1; // do not count the instruction the kernel executes
10 attr.exclude_hv = 1;
11

```

```
12 // open a file descriptor
13 int fd = perf_event_open(&attr, 0, -1, -1, 0);
14
15 if (fd == -1)
16 {
17     // handle error
18 }
```

In Sec. 1.4.3 we also counted the number of floating point instructions, in particular the SSE floating point double precision μ ops. This was done by replacing lines 6 and 7 with

```
1     attr.type = PERF_TYPE_RAW;
2     attr.config = 0x8010;
```

The config variable can be set to different values, which are documented in Ref. [59]. Note that these variables are highly processor dependent and this value was found for an Intel i7 “Nehalem” (Chapter 19.5). The value is composed by the “Umask Value” (in this case 0x80) and then the “Event Number” (in this case 0x10).

We can finally enable the counter in the critical section (line 5) and read it into the variable count (line 10):

```
1 // reset and enable the counter
2 ioctl(fd, PERF_EVENT_IOC_RESET, 0);
3 ioctl(fd, PERF_EVENT_IOC_ENABLE, 0);
4
5 // perform computation that should be measured
6
7 // disable and read out the counter
8 ioctl(fd, PERF_EVENT_IOC_DISABLE, 0);
9 long long count;
10 read(fd, &count, sizeof(long long));
11 // count now has the (approximated) result
12
13 // close the file descriptor
14 close(fd);
```

A.6.2 Efficiently Memorizing Subcurrents

Let us shortly illustrate the problem that we solve here: To compute an amplitude A we use Eq. (1.8) which instructs us to compute a polarization vector (see App. A.2.2 for explicit parametrization) and an on-shell current that is defined recursively, e.g. as defined in Eq. (1.6) for an all gluon process. We can compute these currents easily by simply implementing them using recursive functions in C++ but this way we compute many subcurrents over and over again, e.g. the polarization vectors. Instead, we can, before actually computing the value of the sub-current, find out if we already computed it and then, if this is the case, look it up instead of recomputing it, or compute it for the first time and store it in a cache for later use.

The advantage is that one can use the computer functions one implemented before in the “naïve” approach and add code for memorization.

In our implementation the computation of the currents were already quite fast so we could see an impact of the speed of the memorization. We subsequently found a technique that allows for a fast lookup and storage of results that we describe in the following.

We divide the problem into two parts: Key generation and storage/lookup. If we assume we can generate a “key”, i.e. an integer, for every sub-current, then we can use e.g. C++11’s `std::unordered_map` to store and lookup a sub-current using its corresponding key. The container `std::unordered_map` is a hash table that offers storage and lookup in constant time on average. Like many hash table implementations it has a adjustable parameter `max_load_factor` that can greatly effect the actual lookup/storage times. In our case we found that 0.5 is a good choice. Once the first amplitude is computed, the hash table has its final form and since the next phase-space point has the same subcurrents we can simply overwrite the old values.

The other part, the key generation, makes use of the fact that a single partial amplitude A has a fixed permutation of the momenta, e.g. if we compute $A(p_1, p_2, p_3, p_4)$ for an all-gluon process we will need the following currents,

$$J^\mu(p_2), J^\mu(p_3), J^\mu(p_4), J^\mu(p_2, p_3), J^\mu(p_3, p_4), J^\mu(p_2, p_3, p_4), \quad (\text{A.83})$$

where the relative ordering of the arguments is preserved. We then compute a “global key” g ,

$$(x_0, x_1, \dots, x_{n-1}) \mapsto g = \sum_{i=0}^{n-1} x_i \cdot 2^{4i} \quad (\text{A.84})$$

once for every amplitude where the x_i are the indices of the momenta (in our example above, $x_0 = 1, x_1 = 2, x_2 = 3, x_3 = 4$). The keys of the subcurrents are then computed by the same map, but with a restricted tuple of length $l < n$ starting at k by

$$\begin{aligned} (x_k, x_{k+1}, \dots, x_{k+l}) \mapsto g - \sum_{i=0}^{k-1} x_i \cdot 2^{4i} - \sum_{i=k+l+1}^{n-1} x_i \cdot 2^{4i} \\ = \left((g \bmod 2^{4(k+l+1)}) \operatorname{div} 2^{4 \cdot (k-1)} \right) \cdot 2^{4k} \quad (\text{A.85}) \end{aligned}$$

with `div` denoting integer division, and `mod` the remainder of the integer division. This can be computed with only two bit-shift operations, two integer multiplications and one subtraction and one summation and uses no time-consuming loop.

This operation assumes that $x_i \in \{0, 2^4 - 1\}$, i.e. we have sixteen particle at maximum.

Bibliography

- [1] Laurent Canetti, Marco Drewes, and Mikhail Shaposhnikov. “Matter and Antimatter in the Universe”. In: *New Journal of Physics* 14 (2012), p. 095012. DOI: 10.1088/1367-2630/14/9/095012. arXiv: 1204.4186 [hep-ph].
- [2] J. Beringer et al. “Review of Particle Physics”. In: *Physical Review D* 86 (1 2012), p. 010001. DOI: 10.1103/PhysRevD.86.010001.
- [3] Michael Dinsdale, Marko Ternick, and Stefan Weinzierl. “A comparison of efficient methods for the computation of Born gluon amplitudes”. In: *Journal of High Energy Physics* 2006.03 (2006), p. 056. arXiv: hep-ph/0602204.
- [4] Christopher Schwan. “Numerical computation of matrix elements at leading order for QCD with photons”. Diploma thesis. Institut für Physik (THEP), Universität Mainz, 2011. URL: <http://wwwthep.physik.uni-mainz.de/Publications/theses/dip-schwan.pdf> (visited on 05/19/2014).
- [5] F. A. Berends and W. T. Giele. “Recursive calculations for processes with n gluons”. In: *Nuclear Physics B* 306.4 (1988), pp. 759–808. ISSN: 0550-3213. DOI: 10.1016/0550-3213(88)90442-7.
- [6] Ronald Kleiss and Hans Kuijf. “Multigluon cross sections and 5-jet production at hadron colliders”. In: *Nuclear Physics B* 312.3 (1989), pp. 616–644. ISSN: 0550-3213. DOI: 10.1016/0550-3213(89)90574-9.
- [7] Stefan Weinzierl. “Automated computation of spin- and colour-correlated Born matrix elements”. In: *The European Physical Journal C* 45 (3 2006), pp. 745–757. ISSN: 1434-6044. DOI: 10.1140/epjc/s2005-02467-6. arXiv: hep-ph/0510157.
- [8] F. A. Berends and W. Giele. “The six-gluon process as an example of Weyl-van der Waerden spinor calculus”. In: *Nuclear Physics B* 294.0 (1987), pp. 700–732. ISSN: 0550-3213. DOI: 10.1016/0550-3213(87)90604-3.
- [9] Michelangelo Mangano, Stephen Parke, and Zhan Xu. “Duality and multi-gluon scattering”. In: *Nuclear Physics B* 298.4 (1988), pp. 653–672. ISSN: 0550-3213. DOI: 10.1016/0550-3213(88)90001-6.

- [10] Michaelangelo Mangano and Stephen J. Parke. “Quark-gluon amplitudes in the dual expansion”. In: *Nuclear Physics B* 299.4 (1988), pp. 673–692. ISSN: 0550-3213. DOI: 10.1016/0550-3213(88)90368-9.
- [11] Michelangelo Mangano. “The color structure of gluon emission”. In: *Nuclear Physics B* 309.3 (1988), pp. 461–475. ISSN: 0550-3213. DOI: 10.1016/0550-3213(88)90453-1.
- [12] Vittorio Del Duca, Lance J. Dixon, and Fabio Maltoni. “New color decompositions for gauge amplitudes at tree and loop level”. In: *Nuclear Physics B* 571 (2000), pp. 51–70. DOI: 10.1016/S0550-3213(99)00809-3. arXiv: hep-ph/9910563.
- [13] Zvi Bern and David A. Kosower. “Color decomposition of one-loop amplitudes in gauge theories”. In: *Nuclear Physics B* 362.1–2 (1991), pp. 389–448. ISSN: 0550-3213. DOI: 10.1016/0550-3213(91)90567-H.
- [14] Christian Reuschle and Stefan Weinzierl. “Decomposition of one-loop QCD amplitudes into primitive amplitudes based on shuffle relations”. In: *Physical Review D* 88.10 (2013), p. 105020. DOI: 10.1103/PhysRevD.88.105020. arXiv: 1310.0413 [hep-ph].
- [15] F. Maltoni et al. “Color-flow decomposition of QCD amplitudes”. In: *Physical Review D* 67.1 (Jan. 2003), p. 014026. DOI: 10.1103/PhysRevD.67.014026. arXiv: hep-ph/0209271.
- [16] *BLAS (Basic Linear Algebra Subprograms)*. URL: <http://www.netlib.org/blas/> (visited on 09/01/2014).
- [17] Daniel Goetz. “Efficient automated computation of tree-level amplitudes in QCD and QED”. Diploma thesis. Institut für Physik (THEP), Universität Mainz, 2011. URL: <http://wwwthep.physik.uni-mainz.de/Publications/theses/dip-dagoetz.pdf> (visited on 05/19/2014).
- [18] Claude Duhr, Stefan Höche, and Fabio Maltoni. “Color-dressed recursive relations for multi-parton amplitudes”. In: *Journal of High Energy Physics* 2006.08 (2006), p. 062. DOI: 10.1088/1126-6708/2006/08/062. arXiv: hep-ph/0607057.
- [19] Walter Giele, Gerben Stavenga, and Jan-Christopher Winter. “Thread-scalable evaluation of multi-jet observables”. In: *Eur. Phys. J. C* 71 (2011). DOI: 10.1140/epjc/s10052-011-1703-5. arXiv: 1002.3446 [hep-ph].
- [20] S. Catani et al. “New clustering algorithm for multijet cross sections in $e^+e^- \rightarrow \gamma^* \rightarrow q\bar{q}$ annihilation”. In: *Physics Letters B* 269.3–4 (1991), pp. 432–438. ISSN: 0370-2693. DOI: 10.1016/0370-2693(91)90196-W.

-
- [21] G. Dissertori. “QCD studies with e^+e^- annihilation data at 189 GeV”. In: CERN-OPEN-99-291, ALEPH-99-023 (1999).
- [22] Sebastian Becker et al. “NLO results for five, six and seven jets in electron-positron annihilation”. In: *Physical Review Letters* 108 (2012), p. 032005. DOI: 10.1103/PhysRevLett.108.032005. arXiv: 1111.1733 [hep-ph].
- [23] William Kahan. “Pracniques: Further Remarks on Reducing Truncation Errors”. In: *Commun. ACM* 8.1 (1965), pp. 40–. ISSN: 0001-0782. DOI: 10.1145/363707.363723.
- [24] Petros Draggiotis, Ronald H. P. Kleiss, and Costas G. Papadopoulos. “On the computation of multigluon amplitudes”. In: *Physics Letters B* 439.1-2 (1998), pp. 157–164. ISSN: 0370-2693. DOI: 10.1016/S0370-2693(98)01015-6. arXiv: hep-ph/9807207.
- [25] David Goldberg. “What Every Computer Scientist Should Know About Floating-point Arithmetic”. In: *ACM Comput. Surv.* 23.1 (1991), pp. 5–48. ISSN: 0360-0300. DOI: 10.1145/103162.103163.
- [26] F. Bloch and A. Nordsieck. “Note on the Radiation Field of the Electron”. In: *Physical Review* 52 (2 1937), pp. 54–59. DOI: 10.1103/PhysRev.52.54.
- [27] Toichiro Kinoshita. “Mass Singularities of Feynman Amplitudes”. In: *Journal of Mathematical Physics* 3 (4 1962), pp. 650–677. DOI: 10.1063/1.1724268.
- [28] T. D. Lee and M. Nauenberg. “Degenerate Systems and Mass Singularities”. In: *Phys. Rev.* 133 (6B 1964), B1549–B1562. DOI: 10.1103/PhysRev.133.B1549.
- [29] S. Catani and M. H. Seymour. “A general algorithm for calculating jet cross sections in NLO QCD”. In: *Nuclear Physics B* 485 (1997), pp. 291–419. DOI: 10.1016/S0550-3213(96)00589-5. arXiv: hep-ph/9605323.
- [30] S. Frixione, Z. Kunszt, and A. Signer. “Three-jet cross sections to next-to-leading order”. In: *Nuclear Physics B* 467.3 (1996), pp. 399–442. ISSN: 0550-3213. DOI: 10.1016/0550-3213(96)00110-1. arXiv: hep-ph/9512328.
- [31] Stefano Frixione. “A general approach to jet cross sections in QCD”. In: *Nuclear Physics B* 507.1–2 (1997), pp. 295–314. ISSN: 0550-3213. DOI: 10.1016/S0550-3213(97)00574-9. arXiv: hep-ph/9706545.
- [32] David A. Kosower. “Antenna factorization of gauge-theory amplitudes”. In: *Physical Review D* 57 (9 1998), pp. 5410–5416. DOI: 10.1103/PhysRevD.57.5410. arXiv: hep-ph/9710213.

- [33] John Collins. *Foundations of Perturbative QCD*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology 32. Cambridge University Press, 2013. ISBN: 978-1-107-64525-7.
- [34] Guido Altarelli and G. Parisi. “Asymptotic Freedom in Parton Language”. In: *Nuclear Physics B* 126 (1977), p. 298. DOI: 10.1016/0550-3213(77)90384-4.
- [35] Daniel Goetz, Christopher Schwan, and Stefan Weinzierl. “Random polarisations of the dipoles”. In: *Physical Review D* 85 (2012), p. 116011. DOI: 10.1103/PhysRevD.85.116011. arXiv: 1205.4109 [hep-ph].
- [36] G. Peter Lepage. “A new algorithm for adaptive multidimensional integration”. In: *Journal of Computational Physics* 27.2 (1978), pp. 192–203. ISSN: 0021-9991. DOI: 10.1016/0021-9991(78)90004-9.
- [37] G. Peter Lepage. *VEGAS. An Adaptive Multidimensional Integration Program*. CLNS-80/447. 1980. URL: <http://www.physics.buffalo.edu/phy411-506/topic6/VEGAS.pdf> (visited on 03/07/2014).
- [38] Stefan Weinzierl. *Introduction to Monte Carlo methods*. 2000. arXiv: hep-ph/0006269.
- [39] F. James. “Monte Carlo Theory and Practice”. In: *Reports on Progress in Physics* 43 (1980), pp. 1145–1189. DOI: 10.1088/0034-4885/43/9/002.
- [40] Ronald Kleiss and Roberto Pittau. “Weight optimization in multichannel Monte Carlo”. In: *Computer Physics Communications* 83 (1994), pp. 141–146. DOI: 10.1016/0010-4655(94)90043-4. arXiv: hep-ph/9405257.
- [41] Thomas Hahn. “CUBA: A Library for multidimensional numerical integration”. In: *Computer Physics Communications* 168 (2005), pp. 78–95. DOI: 10.1016/j.cpc.2005.01.010. arXiv: hep-ph/0404043.
- [42] William H. Press et al. *Numerical Recipes. The Art of Scientific Computing*. Cambridge University Press. ISBN: 978-0521880688. URL: <http://www.nr.com> (visited on 05/27/2014).
- [43] *GNU Scientific Library*. URL: <http://www.gnu.org/software/gsl/>.
- [44] Christopher Schwan. *A C++11 Template Library for Monte Carlo Integration*. URL: <https://github.com/cschwan/hep-mc>.
- [45] Makoto Matsumoto and Takuji Nishimura. “Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator”. In: *ACM Trans. Model. Comput. Simul.* 8.1 (1998), pp. 3–30. ISSN: 1049-3301. DOI: 10.1145/272991.272995.

-
- [46] Richard Kreckel. “Parallelization of adaptive MC integrators”. In: *Computer Physics Communications* 106.3 (1997), pp. 258–266. ISSN: 0010-4655. DOI: 10.1016/S0010-4655(97)00099-4. arXiv: physics/9710028.
- [47] *Message Passing Interface Forum*. URL: <http://www.mpi-forum.org/> (visited on 07/03/2014).
- [48] Edgar Gabriel et al. “Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation”. In: *Proceedings, 11th European PVM/MPI Users’ Group Meeting*. Budapest, Hungary, 2004, pp. 97–104.
- [49] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. “Optimization of Collective Communication Operations in MPICH”. In: *International Journal of High Performance Computing Applications* 19.1 (2005), pp. 49–66. DOI: 10.1177/1094342005051521.
- [50] R. Kleiss and W. J. Stirling. “Spinor techniques for calculating $p\bar{p} \rightarrow W^\pm/Z^0 + \text{JETS}$ ”. In: *Nuclear Physics B* 262.2 (1985), pp. 235–262. ISSN: 0550-3213. DOI: 10.1016/0550-3213(85)90285-8.
- [51] Palesh B. Pal. *Dirac, Majorana and Weyl fermions*. arXiv: 1006.1718 [hep-ph].
- [52] Michael E. Peskin and Daniel V. Schroeder. *An Introduction to Quantum Field Theory*. Westview Press, 1995. ISBN: 0-201-50397-2.
- [53] Manfred Böhm, Ansgar Denner, and Hans Joos. *Gauge Theories of the Strong and Electroweak Interaction*. B. G. Teubner, 2001. ISBN: 3-519-23045-3.
- [54] R. Kleiss, W. J. Stirling, and S. D. Ellis. “A new Monte Carlo treatment of multiparticle phase space at high energies”. In: *Computer Physics Communications* 40 (1986), pp. 359–373. ISSN: 0010-4655. DOI: 10.1016/0010-4655(86)90119-0.
- [55] Stefan Weinzierl and David A. Kosower. “QCD corrections to four jet production and three jet structure in e^+e^- annihilation”. In: *Physical Review D* 60 (1999), p. 054028. DOI: 10.1103/PhysRevD.60.054028. arXiv: hep-ph/9901277.
- [56] *perf: Linux profiling with performance counters*. URL: https://perf.wiki.kernel.org/index.php/Main_Page (visited on 08/06/2014).
- [57] *Valgrind: A GPL’d system for debugging and profiling Linux programs*. URL: <http://valgrind.org/> (visited on 08/06/2014).
- [58] Nicholas Nethercote and Julian Seward. “Valgrind: A Framework for Heavy-weight Dynamic Binary Instrumentation”. In: *SIGPLAN Not.* 42.6 (2007), pp. 89–100. ISSN: 0362-1340. DOI: 10.1145/1273442.1250746.

- [59] *Intel®64 and IA-32 Architectures Software Developer's Manual. Volume 3B: System Programming Guide, Part 2.* Chap. 19. URL: <http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html> (visited on 08/06/2014).