Numerical computation of matrix elements at leading order for QCD with photons

Diplomarbeit

am

Institut für Physik (WA THEP)

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

vorgelegt von

Christopher Schwan

Mai 2010

Contents

Chapter 1

Introduction

The standard model of particle physics incorporates every currently known elementary particle and describes their interactions.

The standard model, as a theory, can be seen as the sum of three important developments. This is the extension of quantum electrodynamics to the *electroweak model*[1, 2] which describes the leptons and their mediating bosons: The photon and the weak bosons. The Higgs-mechanism[3–5] is used to give the weak bosons mass. On the other hand this is the formulation of Quantum Chromodynamics[6, 7] (QCD) describing quarks and gluons. Finally, the quark-mixing matrix[8, 9] (CKM-matrix) describes how electroweak bosons couple to quarks and thus connects both theories to the standard model we know it nowadays.

Despite its success, the standard model leaves fundamental questions unanswered. For example, one can ask why the masses take the value one can measure or why there are three generations of fermions. Furthermore, the Higgs particle which is predicted by the standard model remains to be observed.

To this end, there are particle collider experiments testing the standard model and to look for yet unknown particles, e.g. the Higgs H. Since these particles must have masses higher than energies already probed in past experiments, a high center-of-mass energy is crucial. This implies the possibility of events with a *large number of particles* which theoretical predictions must take into account.

Purpose and Outline of this Thesis

In this thesis we deal with a part of the theoretical prediction. Our goal is the numerical computation of QCD matrix elements, which are the dominant ones, with

additional photons for an arbitrary number of particles. More formal, we want to compute matrix elements of a quantum field theory with gauge group SU(N)×U(1). The extension of the gauge group with U(1) was chosen because it is the simplest extension of QCD and prepatory work to the computation of full standard model matrix elements. We restrict ourselfs to matrix elements, because the additional phase space integration needed for observables is a challenge for its own.

In detail, our goal is the automated computation of matrix elements with following properties:

Numerical Our computations will be numerical, since for a large number of particles analytical computations are no longer praticable. We will discuss this point further in section 2.1.

Efficient Processes with a large number of particles require much more computation time than processes with less. Since we are indeed interested in such processes, computations must be efficient. In section 3.7 we will present techniques for speeding up computations.

Tree-level The computations will be carried out in perturbative QFT, thus results will only approximate the exact matrix element. We restrict ourselfs to tree-level matrix elements. The tree matrix elements will then be the basis for computations to the next higher order[10].

Color-ordered The computation will be based on a color-ordered[11] algorithm[12].

Conventions used in this thesis

Throughout this thesis the summation convention is used, i.e. summation is implied over indices appearing twice in a product. We use latin lowercase letters $a, b, \ldots$ for color indices, dotted and undotted latin uppercase letters $A, \dot{A}, B, \ldots$ for spinor indices and greek letters $\alpha, \beta, \ldots$ for lorentz indices. Sometimes objects carry a lot of indices and arguments; in this case, a simple letter or number printed in bold is used to abbreviate these. For example, the polarization vector of a gluon with particle index 1, momenta p_1, reference momenta q_1, helicity h_1 and lorentz-index μ_1 is abbreviated as follows:

$$\varepsilon_{\mu_1}\left(p_1, q_1, h_1\right) \equiv \varepsilon_{\mathbf{1}} \tag{1.1}$$

Chapter 2

Foundations: pQFT

Perturbative quantum field theory (pQFT) is a high energy physics framework enabling the physicist to compute *observables*, i.e. functions which can be measured in experiments. For example, the *total cross section* σ for the *process* $e^+e^- \to \mu^+\mu^-$ is the rate of production of Muon-pairs by colliding electrons with positrons.

The computation of an observable $\mathcal{O}$ requires knowledge of the corresponding *squared matrix element* $|\mathscr{M}|^2$ which depends on the participating particles, their momenta, helicites/spin direction and maybe other quantum numbers. In perturbative QFT one approximates $\mathscr{M}$ up to a definite order of the *coupling constants* g_i (assumed to be small). For a theory with one coupling constant g, e.g. QCD, this reads

$$\mathscr{M} = g^{n-2}\Big(\mathscr{M}^{(0)} + g^2\mathscr{M}^{(1)} + g^4\mathscr{M}^{(2)} + \ldots\Big). \tag{2.1}$$

Dots denote higher orders which are neglected, n denotes the number of particles. The matrix elements $\mathscr{M}^{(i)}$ are typically calculated with the method of *Feynman diagrams*.[13]

This thesis deals with the so-called *tree-level* (sometimes also called Born-level or leading-order) matrix elements $\mathscr{M}^{(0)}$, which from now on will be written as $\mathscr{M}$ without superscript and contains the coupling constants g.

2.1 The Feynman Diagram Method

The Feynman diagram method is a formalism which is usually employed for calculating matrix elements analytically. The general procedure for calculating tree-level matrix elements $\mathscr{M}$ is as follows:

n	Diagrams	Square	Color-ordered
4	4	16	3
5	25	625	10
6	220	48400	36
7	2485	6175225	133
8	34300	1176490000	501

Table 2.1: *Number of Feynman diagrams for a processes with n gluons (numbers taken from [11]) vs. the number of terms yielded from squaring the matrix element*

1. Write down the process to be calculated.
2. Using the *Feynman rules*, which are determined by the gauge theory one uses, construct every Feynman diagram contributing to the process.
3. Translate the diagrams using Feynman rules into an analytic expression, imposing momentum conservation at each vertex.
4. If the process contains more than one fermion pair, figure out the overall signs of Feynman diagrams. Sum up every diagram. The result is the matrix element $\mathscr{M}$.
5. If unpolarized oberservables are calculated, sum the squared matrix element over helicities/spin directions and average accordingly. Treat other quantum numbers such as color similarly.

The construction of Feynman diagrams in item 2 is described in every QFT-textbook, e.g. Peskin and Schroeder [13]. An algorithm for the contruction of tree-level digrams is described e.g. in Stelzer and Long [14].

By calculating matrix elements for processes with five and more particles one immediately recognizes that the number of Feynman diagrams quickly grows and analytical calculations become more and more cumbersome. The “worst case” in QCD are processes with gluons only, see Table 2.1. The number of diagrams asymptotically grows[11] as $(2n)!$.

Another source of complexity arises when one calculates the squared matrix element. Because the Feynman diagrams $\mathscr{M}_i$ depend on momenta and helicities, they

can not be evaluated at this point and thus the number of terms are also squared:

$$\mathscr{M} = \sum_{i=1}^{k} \mathscr{M}_i \quad \Rightarrow \quad |\mathscr{M}|^2 = \sum_{i=1}^{k} \sum_{j=1}^{k} \mathscr{M}_i^* \mathscr{M}_j. \tag{2.2}$$

After squaring the matrix element one usually applies spin- and helicity sums in order to obtain the unpolarized result. This requires additional summation over the helicities[1]

$$\sum_{\lambda_1=\pm} \cdots \sum_{\lambda_n=\pm} \tag{2.3}$$

In analytical calculations we can replace these summations with[15, 16]

$$\sum_{\lambda=\pm} \varepsilon_\lambda^\mu(p) \left(\varepsilon_\lambda^\nu(p)\right)^* = -g^{\mu\nu} + \frac{p^\mu q^\nu + q^\mu p^\nu}{p \cdot q}. \tag{2.4}$$

The momenta q are arbitrary lighlike momenta. In QED calculations one may leave out the fraction, but in this case Equation 2.4 is no longer an identity. This equation also implies that explicit expressions for the polarization states are not neccessarily needed.

In summary, we can conclude that that these analytic "textbook techniques" are not well suited for the calculation of matrix elements for processes with a large number of external particles.

A solution for the problem, which we will follow, is to abstain from calculating an analytical expression and instead do *numerical computation* on a computer: Once particle information as momenta, helicity and so on is known, one can evaluate the contributing Feynman diagrams $\mathscr{M}_i$ to a *number* and add up their results — this is a simple addition of complex numbers. After summing, squaring the resulting complex number yields the desired result for $|\mathscr{M}|^2$.

2.1.1 Crossing symmetry

An important relation in quantum field theories is *crossing symmetry*[13, 17]. It relates amplitudes of "similar" processes and thus helps to simplify certain calculations. For example, let us look at a process with two incoming (P_1, P_2) and two outgoing (P_3, P_4) particles with momenta p_1, p_2, electrical charge Q_1, Q_2 and p_3, p_4, Q_3, Q_4

[1] Again, we assume massless spin-1 particles. Summation for massive particles/particles with different spin look similar.

respectively. The momentum and charge conservation for the amplitude $\mathscr{M}\left(P_1P_2 \to P_3P_4\right)$ reads

$$p_1 + p_2 = p_3 + p_4, \tag{2.5}$$

$$Q_1 + Q_2 = Q_3 + Q_4. \tag{2.6}$$

The quantities on the left-hand side denote incoming particles, the right-hand side the quantities of outgoing particles. It is trivial to rewrite both equations to

$$0 = \left(-p_1\right) + \left(-p_2\right) + p_3 + p_4, \tag{2.7}$$

$$0 = \left(-Q_1\right) + \left(-Q_2\right) + Q_3 + Q_4, \tag{2.8}$$

where all four quantities appear as outgoing. Now, crossing symmetry states that one can indeed identify these quantities with a new matrix element

$$\left|\mathscr{M}\left(P_1P_2 \to P_3P_4\right)\right|^2 = \left|\mathscr{M}\left(0 \to \overline{P}_1\overline{P}_2P_3P_4\right)\right|^2 \tag{2.9}$$

which is equal to the old one.

As an example, consider compton scattering which is related to pair annihilation by crossing symmetry

$$\mathrm{e}^-\gamma \to \mathrm{e}^-\gamma \Longleftrightarrow \mathrm{e}^-\mathrm{e}^+ \to \gamma\gamma. \tag{2.10}$$

The outgoing electron in compton scattering is taken to be incoming, thus the electrical charge is reversed — the electron now has become a positron. The same is done with the incoming photon which remains a photon in the outgoing channel. The resulting process is pair-annihilation. However, that forces us to cope with negative energies. This requires a proper analytic continuation of the polarization states.

One of the advantages of crossing symmetry is that incoming particles can be transformed into the outgoing channel. In particular, this is done by reversing the momenta and flipping helicities, as demontrated above. Having every particle outgoing, one does not need the dirac spinors for incoming particles:

$$u^s(p), \overline{v}^s(p) \tag{2.11}$$

Furthermore, crossing symmetry lets us conclude that there are e.g. only four distinct processes with four particles in an arbitrary gauge theory:

- Four bosons, for example gluon scattering in QCD: $\mathrm{gg} \to \mathrm{gg}$
- Two bosons, one fermion/antifermion pair

- Two fermion/antifermion pairs of the same flavor
- Two fermion/antifermion pairs of different flavors

This is be important when we want to check our program — every possible process can be checked by testing distinct processes and to verify crossing symmetry by permuting the particles and their momementa.

2.2 Color decomposition

Numerical calculation of amplitudes is made difficult by the fact that Feynman rules involve a lot of different mathematical objects. These are "space-time objects" such as dirac spinors, gamma matrices and lorentz vectors, but also "inner-symmetry objects" like color matrices T^a and their structure constants f^{abc} for non-abelian theories such as QCD. In this section we will show how to eliminate the structure constants appearing in Feynman rules of $SU(N)$-gauge theories in favor of traces of T^a. Together with matrix entries $T^a_{i\bar{j}}$ of the fermion vertices one arrives with a simpler structure, which will be further simplified in subsection 2.2.5.

We use the following normalization of the $SU(N)$-generators T^a, $a \in \{1,\dots,8\}$:

$$\mathrm{Tr}\left(T^a T^b\right) = \frac{1}{2}\delta^{ab}. \tag{2.12}$$

The generators satisfy the following identities:

$$\left[T^a, T^b\right] = i f^{abc} T^c. \tag{2.13}$$

By multiplying Equation 2.13 with a generator and taking the trace one shows that the structure constant f^{abc} can be represented by the trace of three generators

$$i f^{abc} = 2\,\mathrm{Tr}\left(\left[T^a, T^b\right] T^c\right) = 2\left\{\mathrm{Tr}\left(T^a T^b T^c\right) - \mathrm{Tr}\left(T^b T^a T^c\right)\right\} \tag{2.14}$$

which allows us to rewrite the three- and four-boson feynman rules containing the structure constants.

We will also make use of a Fierz identitiy for the generators of SU(N)

$$T^a_{ij} T^a_{kl} = \frac{1}{2}\left(\delta_{il}\delta_{jk} - \frac{1}{N}\delta_{ij}\delta_{kl}\right). \tag{2.15}$$

2.2.1 Three-boson-vertex

The Feynman rule for the three-boson-vertex reads (see also Appendix C):

$$V_{\text{ggg}}^{abc,\alpha\beta\gamma}\left(p_1,p_2,p_3\right) = g f^{abc}\Big\{g^{\alpha\beta}\left(p_1-p_2\right)^{\gamma} + g^{\beta\gamma}\left(p_2-p_3\right)^{\alpha} + g^{\gamma\alpha}\left(p_3-p_1\right)^{\beta}\Big\}. \tag{2.16}$$

We define the color-less part of the vertex as

$$V_{\text{ggg}}^{\alpha\beta\gamma}\left(p_1,p_2,p_3\right) = -i\left\{g^{\alpha\beta}\left(p_1-p_2\right)^{\gamma} + g^{\beta\gamma}\left(p_2-p_3\right)^{\alpha} + g^{\gamma\alpha}\left(p_3-p_1\right)^{\beta}\right\}. \tag{2.17}$$

In this definition the vertex is antisymmetric by interchanging the arguments $\left(\alpha,p_1\right) \leftrightarrow \left(\beta,p_2\right)$

$$V_{\text{ggg}}^{\alpha\beta\gamma}\left(p_1,p_2,p_3\right) = -V_{\text{ggg}}^{\beta\alpha\gamma}\left(p_2,p_1,p_3\right), \tag{2.18}$$

which leads together with the structure constant in Equation 2.14 to the following identity for the three-boson-vertex

$$V_{\text{ggg}}^{abc,\alpha\beta\gamma}\left(p_1,p_2,p_3\right) = 2g\,\text{Tr}\left(T^a T^b T^c\right) V_{\text{ggg}}^{\alpha\beta\gamma}\left(p_1,p_2,p_3\right) + 2g\,\text{Tr}\left(T^b T^a T^c\right) V_{\text{ggg}}^{\beta\alpha\gamma}\left(p_2,p_1,p_3\right). \tag{2.19}$$

By letting $\mathbf{1},\mathbf{2},\mathbf{3}$ denote the indices and vectors (a,α,p_1), (b,β,p_2), and (c,γ,p_3) respectively, one can write this identity in a more concise way

$$V_{\text{ggg}}^{abc,\alpha\beta\gamma} = 2g\sum_{P(\mathbf{1},\mathbf{2})}\text{Tr}\left(T^{\mathbf{1}}T^{\mathbf{2}}T^{\mathbf{3}}\right)V_{\text{ggg}}^{\mathbf{123}} \tag{2.20}$$

which shows that the summation is actually a summation over a non-cyclical permutation.

2.2.2 Four-boson-vertex

The four-boson-vertex

$$V_{\text{gggg}}^{abcd,\alpha\beta\gamma\delta} = -ig^2\Big\{f^{abe}f^{cde}\left(g^{\alpha\gamma}g^{\beta\delta}-g^{\alpha\delta}g^{\beta\gamma}\right) + f^{ace}f^{bde}\left(g^{\alpha\beta}g^{\gamma\delta}-g^{\alpha\delta}g^{\beta\gamma}\right) + f^{ade}f^{bce}\left(g^{\alpha\beta}g^{\gamma\delta}-g^{\alpha\gamma}g^{\beta\delta}\right)\Big\}, \tag{2.21}$$

is rewritten similarly, though it needs much more work. The structure constants translate into two traces of generators with repeated indices which are summed over.

The Fierz-identity in Equation 2.15 is used to perform the summation. The result is a sum of traces of four generators and terms proportional to $1/N$ of a product of traces of two generators. The terms poportional to $1/N$ cancel each other. The remaining terms must be multiplied with the metric tensors in the equation above. After sorting the terms the result is

$$V_{\text{gggg}}^{abcd,\alpha\beta\gamma\delta} = 2g^2 \sum_{P(\mathbf{1},\mathbf{2},\mathbf{3})} \text{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}} T^{\mathbf{4}}\right) V_{\text{gggg}}^{\mathbf{1234}} \tag{2.22}$$

which looks quite similar to Equation 2.20. The color-less part of the vertex reads

$$V_{\text{gggg}}^{\mathbf{1234}} = i\left\{2g^{\mathbf{13}}g^{\mathbf{24}} - g^{\mathbf{14}}g^{\mathbf{23}} - g^{\mathbf{12}}g^{\mathbf{34}}\right\}. \tag{2.23}$$

2.2.3 Matrix elements

Having the color ordered Feynman rules at hand, one can now begin to calculate processes involving bosons only. The most simple process is a gluon-decay: g $\rightarrow$ gg. Although this process is kinematically forbidden, we will calculate its matrix element because it is simple and already takes the general form of n-gluon scattering. The only Feynman diagram contributing to the processis is a three-gluon-vertex with polarization vectors attached

$$\mathscr{M} = 2g \sum_{P(\mathbf{1},\mathbf{2})} \text{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}}\right) V_{\text{ggg}}^{\mathbf{123}} \varepsilon_{\mathbf{1}}\varepsilon_{\mathbf{2}}\varepsilon_{\mathbf{3}}. \tag{2.24}$$

One can rewrite the last equation to

$$\mathscr{M} = 2g \sum_{P(\mathbf{1},\mathbf{2})} \text{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}}\right) A(\mathbf{1},\mathbf{2},\mathbf{3}), \tag{2.25}$$

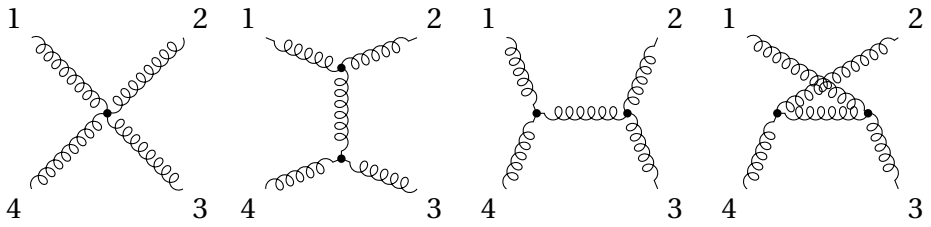
where we implitly defined the *partial amplitude*

$$A(\mathbf{1},\mathbf{2},\mathbf{3}) = \varepsilon_{\mathbf{1}}\varepsilon_{\mathbf{2}}\varepsilon_{\mathbf{3}} V_{\text{ggg}}^{\mathbf{123}}, \tag{2.26}$$

which can be represented by diagrams similar to those of Feynman diagrams (in this case the diagram is equal to the Feynman diagram). The traces in front of the partial amplitudes carrying the color information are sometimes called Chan-Paton factors[18], we will also call them *color-factors* in a more general context.

The first non-trivial process has four gluons. The contributing Feynman diagrams are shown in Figure 2.1. The part of the four-gluon-vertex can be immediately written down and reads

$$\mathscr{M}_4 = 2g^2 \sum_{P(\mathbf{1},\mathbf{2},\mathbf{3})} \text{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}} T^{\mathbf{4}}\right) V_{\text{gggg}}^{\mathbf{1234}} \varepsilon_{\mathbf{1}}\varepsilon_{\mathbf{2}}\varepsilon_{\mathbf{3}}\varepsilon_{\mathbf{4}}. \tag{2.27}$$



(a) four-vertex (b) s-channel (c) u-channel (d) t-channel

Figure 2.1: *Feynman diagrams for the four-gluon-scattering process: $gg \to gg$. Numbers denote in- and outgoing gluons. The channel type is determined with the usual definition of the mandelstam variables[19]: $s = (p_1 + p_2)^2$, $t = (p_1 - p_3)^2$, and $u = (p_1 - p_4)^2$.*

The remaining three Feynman diagrams with two three-gluon-vertices and a gluon-propagator are more complicated, but by clever arragement[20] of terms one can write down a short expression for all diagrams

$$\mathcal{M}_{1+2+3} = \sum_{C(\mathbf{1},\mathbf{2},\mathbf{3})} \underbrace{\left(2g \sum_{P(\mathbf{1},\mathbf{2})} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\boldsymbol{a}}\right) V_{\mathrm{ggg}}^{\mathbf{12}\boldsymbol{a}} \varepsilon_{\mathbf{1}} \varepsilon_{\mathbf{2}}\right)}_{\text{three-gluon-vertex}} \underbrace{\left(P(\mathbf{1}+\mathbf{2})\, \delta^{\boldsymbol{ab}}\right)}_{\text{gluon-propagator}} \underbrace{\left(2g \sum_{P(\mathbf{3},\mathbf{4})} \mathrm{Tr}\left(T^{\boldsymbol{b}} T^{\mathbf{3}} T^{\mathbf{4}}\right) V_{\mathrm{ggg}}^{\boldsymbol{b}\mathbf{34}} \varepsilon_{\mathbf{3}} \varepsilon_{\mathbf{4}}\right)}_{\text{three gluon vertex}}. \quad (2.28)$$

These three diagrams are constructed by cyclical permutation (denoted by $C(\mathbf{1},\mathbf{2},\mathbf{3})$ in the equation above) of the first three particles, see Figure 2.2 for a graphical proof.

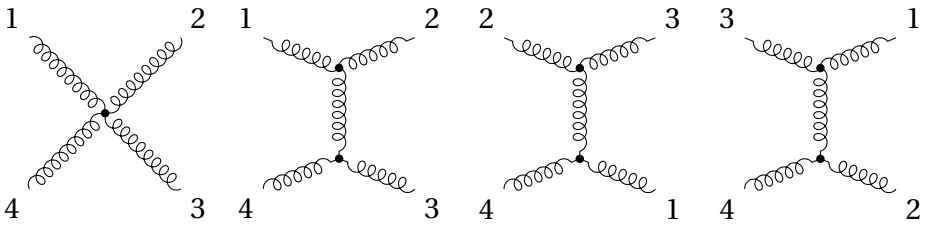
The traces occuring in Equation 2.28 are again evaluated with Fierz-identities, because $\boldsymbol{a}$ and $\boldsymbol{b}$, the types of internal gluons, are implicity summed over:

$$\mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\boldsymbol{a}}\right) \mathrm{Tr}\left(T^{\boldsymbol{a}} T^{\mathbf{3}} T^{\mathbf{4}}\right) = \frac{1}{2} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}} T^{\mathbf{4}}\right) - \frac{1}{2N} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}}\right) \mathrm{Tr}\left(T^{\mathbf{3}} T^{\mathbf{4}}\right). \quad (2.29)$$

Terms proportional to $1/N$ again cancel each other by making use of the antisymmetry of the three-gluon-vertices and the trace property, for example:

$$\frac{1}{N} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}}\right) \mathrm{Tr}\left(T^{\mathbf{3}} T^{\mathbf{4}}\right) V_{\mathrm{ggg}}^{\mathbf{12}\boldsymbol{a}} V_{\mathrm{ggg}}^{\boldsymbol{a}\mathbf{34}} + \frac{1}{N} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}}\right) \underbrace{\mathrm{Tr}\left(T^{\mathbf{4}} T^{\mathbf{3}}\right)}_{\mathrm{Tr}\left(T^{\mathbf{3}} T^{\mathbf{4}}\right)} V_{\mathrm{ggg}}^{\mathbf{12}\boldsymbol{a}} \underbrace{V_{\mathrm{ggg}}^{\boldsymbol{a}\mathbf{43}}}_{-V_{\mathrm{ggg}}^{\boldsymbol{a}\mathbf{34}}} = 0 \quad (2.30)$$

After writing down the terms not proportional to $1/N$ and sorting them one arrives



(a) four-vertex (b) s-channel (c) u-channel (d) t-channel

Figure 2.2: *Rearranged Feynman diagrams for the four gluon process in Figure 2.1. Reading the numbers in clockwise order and omitting the last particle yields the cyclic permutations of three numbers:* $(1,2,3)$ *(2.2(b)),* $(2,3,1)$ *(2.2(c)), and* $(3,1,2)$ *(2.2(d)).*

with

$$\mathscr{M}_{1+2+3} = 2g^2 \sum_{P(\mathbf{1},\mathbf{2},\mathbf{3})} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}} T^{\mathbf{4}}\right) \varepsilon_{\mathbf{1}}\varepsilon_{\mathbf{2}}\varepsilon_{\mathbf{3}}\varepsilon_{\mathbf{4}} \left(V_{\mathrm{ggg}}^{\mathbf{12}\boldsymbol{a}} P\left(\mathbf{1}+\mathbf{2}\right) V_{\mathrm{ggg}}^{\boldsymbol{a}\mathbf{34}} + V_{\mathrm{ggg}}^{\mathbf{23}\boldsymbol{a}} P\left(\mathbf{2}+\mathbf{3}\right) V_{\mathrm{ggg}}^{\boldsymbol{a}\mathbf{41}}\right) \quad (2.31)$$

Obviously, the traces are the same as those appearing in the four-gluon-vertex. All four diagrams yield the following expression

$$\mathscr{M}_{1+2+3+4} = 2g^2 \sum_{P(\mathbf{1},\mathbf{2},\mathbf{3})} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} T^{\mathbf{3}} T^{\mathbf{4}}\right) A\left(\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}\right) \quad (2.32)$$

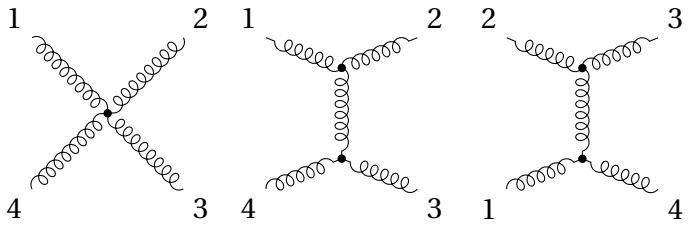
with the partial amplitude

$$A\left(\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}\right) = \varepsilon_{\mathbf{1}}\varepsilon_{\mathbf{2}}\varepsilon_{\mathbf{3}}\varepsilon_{\mathbf{4}} \left(V_{\mathrm{ggg}}^{\mathbf{12}\boldsymbol{A}} P\left(\mathbf{1}+\mathbf{2}\right) V_{\mathrm{ggg}}^{\boldsymbol{A}\mathbf{34}} + V_{\mathrm{ggg}}^{\mathbf{23}\boldsymbol{A}} P\left(\mathbf{2}+\mathbf{3}\right) V_{\mathrm{ggg}}^{\boldsymbol{A}\mathbf{41}} + V_{\mathrm{gggg}}^{\mathbf{1234}}\right) \quad (2.33)$$

One can draw (color-ordered) diagrams representing this partial amplitude, see Figure 2.3. As opposed to the case of three gluons, these diagrams are not the same as the Feynman diagrams; there is no t-channel-type (with crossed legs) contribution. This is special for the color-ordered diagrams. In general, there are significantly fewer color-ordered diagrams than Feynman diagrams (see Table 2.1). However, one needs to sum these diagrams over non-cyclical permutations.

From the structure of the results for three and four gluon matrix elements one conjectures the general case:

$$\mathscr{M} = 2g^{n-2} \sum_{P(\mathbf{1},\ldots,\boldsymbol{n}-\mathbf{1})} \mathrm{Tr}\left(T^{\mathbf{1}} T^{\mathbf{2}} \cdots T^{\boldsymbol{n}}\right) A\left(\mathbf{1},\mathbf{2},\ldots,\boldsymbol{n}\right) \quad (2.34)$$



(a) four-vertex (b) s-channel (c) u-channel

Figure 2.3: *Color-ordered diagrams for four gluon scattering. Note there is no t-channel contribution as opposed to the Feynman diagrams.*

which has been proven by Berends and Giele [20]. The partial amplitudes A are contructed using the *on-shell currents* J^μ

$$A(\mathbf{1},\ldots,\boldsymbol{n}) = J(\mathbf{1},\ldots,\boldsymbol{n-1})\cdot\varepsilon_{\boldsymbol{n}}. \tag{2.35}$$

The on-shell currents in turn are contstructed using the Berends-Giele recursion relations[20] for *off-shell currents* J^μ:

$$\begin{aligned} J^\mu(\mathbf{1},\ldots,\boldsymbol{n}) = P^\mu_\nu(\mathbf{1}+\ldots+\boldsymbol{n}) \Bigg\{ & \sum_{\boldsymbol{j}=\mathbf{2}}^{\boldsymbol{n-1}} V^{\nu\rho\sigma}_{\mathrm{ggg}} J_\rho\left(\mathbf{1},\ldots,\boldsymbol{j}\right) J_\sigma\left(\boldsymbol{j+1},\ldots,\boldsymbol{n}\right) \\ & + \sum_{\boldsymbol{j}=\mathbf{2}}^{\boldsymbol{n-1}} \sum_{\boldsymbol{k}=\boldsymbol{j+1}}^{\boldsymbol{n-2}} V^{\nu\rho\sigma\tau}_{\mathrm{gggg}} J_\rho(\mathbf{1},\ldots,\boldsymbol{j}) J_\sigma(\boldsymbol{j+1},\ldots,\boldsymbol{k}) J_\tau(\boldsymbol{k+1},\ldots,\boldsymbol{n}) \Bigg\} \end{aligned} \tag{2.36}$$

with end condition

$$J^\mu(k) = \varepsilon^\mu(k). \tag{2.37}$$

On- and off-shell currents differ by the definition of the propagator

$$P^{\mu\nu}(p) = \begin{cases} g^{\mu\nu} \\ \frac{ig^{\mu\nu}}{p^2+i\epsilon} \end{cases}, \tag{2.38}$$

which is effectively left out in the case of on-shell currents. A graphical representation of this recursion is drawn in Figure 2.4.

With these recursion relations Berends and Giele [20] were able to prove exact results for matrix elements for certain helicity configurations,which were conjectured before by Parke and Taylor [21].

Up to now, we did not discuss fermions, because in general the resulting formulas are not as easy as for gluons. However, for a single quark-antiquark pair and $n-2$

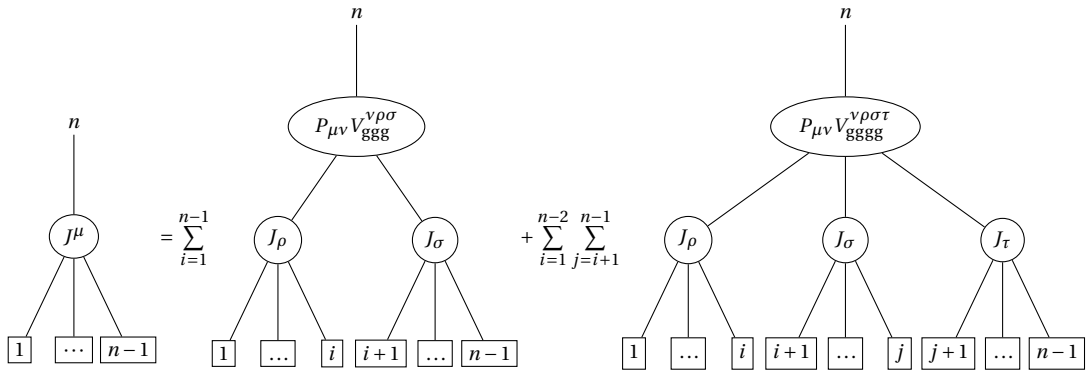


Figure 2.4: *Off-/on-shell recurrence relations for gluon currents. The gluon current* $J^\mu(1,\ldots,n-1)$ *branches into two and three subcurrents, which are attached with a Propagator* $P^{\mu\nu}$ *and the three-gluon-vertex* $V^{\nu\rho\sigma}_{ggg}$ *and four-gluon-vertex* $V^{\nu\rho\sigma\tau}_{gggg}$.

gluon the formula[16] is still simple:

$$\mathcal{M} = g^{n-2} \sum_{P(\mathbf{1},\ldots,\boldsymbol{n-2})} \left(T^{\mathbf{1}}\cdots T^{\boldsymbol{n-2}}\right)_{i\bar{j}} A\left(\overline{q}, q, \mathbf{1}, \ldots, \boldsymbol{n-2}\right). \tag{2.39}$$

This formula is derived in a similar way as we did for the gluons. Because the quark-antiquark-pair carries the same color-structure (color and anticolor) as the gluons, **the pair** is treated as such in the sum over permutations and there are no permutations of quarks with gluons. Furthermore, the presence of quarks is responsible for a *string of color matrices* instead of a trace.

To compute the partial amplitudes, one needs to add quark- and antiquark-currents which couple to gluons via the quark-antiquark-gluon vertex.

2.2.4 The emergence of U(N) and U(1) gluons

Beginning with two quark pairs one additional complicating structure arises: Terms proportional to $1/N$ from the Fierz-identities (Equation 2.15) do not always cancel. To illustrate this, let us look at quark-scattering with different flavors, e.g. $\mathrm{d}\overline{\mathrm{d}} \to \mathrm{u}\overline{\mathrm{u}}$ (see Figure 2.5). The color related part of the only Feynman diagram reads

$$T^a_{i_1\bar{j}_2}\delta^{ab}T^b_{i_3\bar{j}_4}. \tag{2.40}$$

The indices a, b denote the types mediating gluons, $i_1, \bar{j}_2$ are the colors of incoming quark and antiquark, $i_3, \bar{j}_4$ the color of the outgoing particles. With the implicit

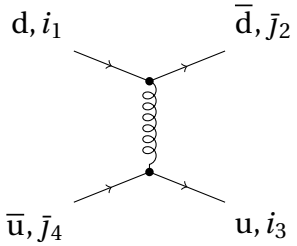


Figure 2.5: *Feynman diagram for quark-scattering: $d\overline{d} \to u\overline{u}$. Indices denote colors of quarks and antiquarks.*

summation over a and b this is the Fierz-identity

$$\frac{1}{2}\left(\delta_{i_1\bar{\jmath}_4}\delta_{\bar{\jmath}_2 i_3} - \frac{1}{N}\delta_{i_1\bar{\jmath}_2}\delta_{i_3\bar{\jmath}_4}\right) \tag{2.41}$$

showing that quark-pairs lead to terms proportional to $1/N$. Note that in the term proportional to $1/N$ the colors of the incoming quarks are connected, as are the colors of the outgoing quarks which is illustrated by the pictorial representation of this equation in Figure 2.5. Thus, we will call the first part *U(N)-gluon* and the second part of the color structure *U(1)-gluon* which does not carry color. These gluons are not physical, of course.

The application of the Fierz-identity which we have done in the previous examples will also lead us to a new color decomposition: The *color-flow decomposition*.

2.2.5 Color-flow Decomposition

The color-flow decomposition was already used by Hooft [22] but introduced by that name in Maltoni et al. [11]. It was shown that a different representation of the color-factors allows to unify the structures appearing in color-ordered matrix elements. Instead of having strings and traces of generators, one deals with strings of Kronecker delta symbols **only**. This allows for a more systematic construction of diagrams containing U(1)-gluons.

The idea of color-flow decomposition is simple: Since matrix elements are usually summed over color, one may insert unities in color space. The important structure of a color-summed matrix element squared for gluons reads:

$$|\mathcal{M}|^2 = \quad \ldots \quad \underbrace{2\,\mathrm{Tr}\left(T^{a_1}\cdots T^{a_n}\right)}_{\text{from } \mathcal{M}^*} \underbrace{\left(\delta^{a_1 b_1}\cdots\delta^{a_n b_n}\right)}_{\text{color summation}} \underbrace{2\,\mathrm{Tr}\left(T^{b_1}\cdots T^{b_n}\right)}_{\text{from } \mathcal{M}} \quad \ldots \tag{2.42}$$

Inserting unities for each color index a_k, b_k

$$\delta^{a'_k a_k} = 2\,\mathrm{Tr}\left(T^{a'_k} T^{a_k}\right) = \sqrt{2}T^{a'_k}_{i_k \bar{j}_k} \sqrt{2}T^{a_k}_{\bar{j}_k i_k} \tag{2.43}$$

$$\delta^{b'_k b_k} = 2\,\mathrm{Tr}\left(T^{b'_k} T^{b_k}\right) = \sqrt{2}T^{b'_k}_{\bar{i}_k j_k} \sqrt{2}T^{b_k}_{\bar{i}_k j_k} \tag{2.44}$$

in between the traces and the color summations yields for the traces

$$2\,\mathrm{Tr}\left(T^{a_1} \cdots T^{a_n}\right)\left(\sqrt{2}T^{a_1}_{i_1 \bar{j}_1}\right) \cdots \left(\sqrt{2}T^{a_n}_{i_n \bar{j}_n}\right) \tag{2.45}$$

and the summations

$$\left(\sqrt{2}T^{a_1}_{\bar{j}_1 i_1} \delta^{a_1 b_1} \sqrt{2}T^{b_1}_{\bar{i}_1 j_1}\right) \cdot \ldots \cdot \left(\sqrt{2}T^{a_n}_{\bar{j}_n i_n} \delta^{a_n b_n} \sqrt{2}T^{b_n}_{\bar{i}_n j_n}\right). \tag{2.46}$$

The traces are thus rewritten to

$$2\sqrt[n]{2}\frac{1}{2^n}\delta_{i_1 \bar{j}_2}\delta_{i_2 \bar{j}_3} \cdots \delta_{i_{n-1} \bar{j}_n}\delta_{i_n \bar{j}_1} + O\left(\frac{1}{N}\right) \tag{2.47}$$

and the summations (color factor) to

$$\left(\delta_{\bar{i}_1 \bar{j}_1}\delta_{i_1 j_1} - \frac{1}{N}\delta_{i_1 \bar{j}_1}\delta_{\bar{i}_1 j_1}\right) \cdot \ldots \cdot \left(\delta_{\bar{i}_n \bar{j}_n}\delta_{i_n j_n} - \frac{1}{N}\delta_{i_n \bar{j}_n}\delta_{\bar{i}_n j_n}\right). \tag{2.48}$$

The color factor contains terms suppressed by powers of $1/N$ which are not written down. These terms vanish if less than two quark-antiquark-pairs are present, which can be shown be applying the procedure above to the color ordered Feynman rules in subsection 2.2.1 and subsection 2.2.2. Terms proportional to powers over $1/N$ cancel each other because of the same symmetries used there. An alternative way is to make use of the Kleiss-Kuijf-relations[23], which enables one to show the same at the level of amplitudes.

One arrives with following expression for gluon amplitudes:

$$\mathscr{M} = \left(\frac{g}{\sqrt{2}}\right)^{n-2} \sum_{P(1,\ldots,n-1)} \delta_{i_1 \bar{j}_2}\delta_{i_2 \bar{j}_3} \cdot \ldots \cdot \delta_{i_n \bar{j}_1} A(1,2,\ldots,n). \tag{2.49}$$

Starting from Equation 2.39 one derives the following color factor for processes with a single antiquark/quark-pair and $(n-2)$ gluons:

$$\delta_{i_2 \bar{j}_3}\delta_{i_3 \bar{j}_4} \cdot \ldots \cdot \delta_{i_{n-1} \bar{j}_n}\delta_{i_n \bar{j}_1}. \tag{2.50}$$

Note that a non-barred index (i_2) appear for quarks and a barred index ($\bar{j}_1$) for anti-quarks; gluons carry both types of indices. This leads to the interpretation that gluons carry a charge and an anti-charge.

2.2.6 Arbitrary number of quarks

If there are more than a single antiquark/quark-pair is present, we have to respect the following complications:

- Quarks couple together with their antiquarks to gluons, therefore those pairs are permuted with gluons. We call those pairs and the gluons *pseudo particles*. The permutations we will then sum over are the *pseudo particle permutations*. Since we are free to choose an order of the particles of the process (if we give particles numbers, this does not make a difference in physics, see also crossing symmetry in subsection 2.1.1), we will order particles in a way such that quarks immediately follow antiquarks. Those pairs may be interspersed with gluons. This order will be called *cyclic order*[12].

- Additional permutations are quark-permutations: By permuting different quarks of the same flavor we obtain additional diagrams. Those diagrams come with a fermion permutation sign (see also section 2.1)

- U(1)-gluons appear in diagrams. These couple only to antiquark/quark-pairs and have a different color factor. The maximum number of U(1)-gluons is

$$n_q - 1. \tag{2.51}$$

  Since U(1)-gluons do not carry color, they separate a given diagram into *color-disconnected* pieces. We call those pieces *color clusters*. For n_q quarks there are

$$n_{\text{cluster}}^{\max} = \max\left(n_q, 1\right) \tag{2.52}$$

  clusters at maximum (there is always at least one cluster). We thus have to generate every diagram containing $\left[0, n_{\text{cluster}}^{\max}\right)$ number of U(1)-gluons.

These considerations lead to the general form of QCD-amplitudes

$$\mathscr{M} = \left(\frac{g}{\sqrt{2}}\right)^{n-2} \sum_{P_i} C\left(P_i\right) A\left(P_i\right), \tag{2.53}$$

with $C\left(P_i\right)$ the color factor, consisting of a string of Kronecker-deltas, and the partial amplitudes $A\left(P_i\right)$ which are computed with a Berends-Giele-type recursion relation. Both objects depend on the permutations of particles and the cluster configuration. The next chapter describes the algorithmic generation of both of these objects.

2.2.7 Limitations of the Color-Decompositions

Up to now, we presented two different color decompositions for SU(N)-theories: One containing the Chan-Patton factors which are suitable for gluon processes and the color-flow decomposition. In section 2.5 we will argue that the procedure also applies to QED. One may now ask if this is possible for every gauge theory and if the formulas for the computation of tree-level amplitudes always take the straightforward form of Equation 2.53. Unfortunately, the answer is no; an example is the electroweak theory. This can be seen by looking at the form of the vertices, e.g. the γ-γ-W^+-W^--vertex, which has the same forma as Equation 2.23:

$$V_{\gamma\gamma W^+W^-} = ie^2\left\{2g^{\mu\gamma}g^{\beta\delta} - g^{\mu\delta}g^{\beta\gamma} - g^{\alpha\beta}g^{\gamma\delta}\right\} \tag{2.54}$$

Since the electroweak theory is spontaneously broken, the vertices do not contain color information (which would be pauli matrices in this case) and thus takes the form of the color-less four-vertex. However, there is no summation over permutations which is responsible of the form of color-ordered amplitudes.

2.3 Weyl-van der Werden Formalism

To treat space-time objects, i.e. Lorentz-vectors and Dirac spinors on the same footing, it is covenient to introduce the Weyl–van der Werden formalism[24]. This formalism is especially helpful for carrying out massless calculations. In our case, we are interested in the spinor formalism because it simplifies the implementation of the objects appearing in Feynman-rules.

We follow the convention used by Borodulin, Rogalev, and Slabospitsky [25] and define the four-dimensional Pauli matrices

$$\sigma^{\mu}_{A\dot{B}} = (1, -\vec{\sigma}) \quad \overline{\sigma}^{\mu\dot{A}B} = (1, \vec{\sigma}) \tag{2.55}$$

using the three-dimensional Pauli matrices

$$\sigma^1 = \begin{pmatrix} 0 & +1 \\ 1 & 0 \end{pmatrix}, \tag{2.56}$$

$$\sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{2.57}$$

$$\sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.58}$$

These obey the relations

$$\sigma^{\mu}_{A\dot{B}}\overline{\sigma}^{\nu\dot{B}A} = 2g^{\mu\nu} \tag{2.59a}$$

$$\sigma^{\mu}_{A\dot{B}}\overline{\sigma}^{\dot{C}D}_{\mu} = 2\delta_{A}{}^{D}\delta_{\dot{B}}{}^{\dot{D}} \tag{2.59b}$$

$$\sigma^{\mu}_{A\dot{B}}\sigma_{\mu C\dot{D}} = 2\epsilon_{AC}\epsilon_{\dot{B}\dot{D}} \tag{2.59c}$$

$$\overline{\sigma}^{\mu\dot{A}B}\overline{\sigma}^{\dot{C}D}_{\mu} = 2\epsilon^{\dot{A}\dot{C}}\epsilon^{BD} \tag{2.59d}$$

where we have used the spinor metric

$$\epsilon_{AB} = \epsilon^{AB} = \epsilon_{\dot{A}\dot{B}} = \epsilon^{\dot{A}\dot{B}} = i\sigma^2 = \begin{pmatrix} 0 & +1 \\ -1 & 0 \end{pmatrix} \tag{2.60}$$

which is used to raise and lower spin indices, very much like $g^{\mu\nu}$ for Lorentz indices.

These definitions allows us to rewrite the Feynman rules. The basic building blocks of those are the metric tensor $g^{\mu\nu}$ (two Lorentz-indices), momenta p^{μ} (one Lorentz index) and the gamma matrices γ^{μ} (one Lorentz-index, two dirac spinor indices or four weyl spinor indices). To make use of the relations above, we have to use the *Weyl-* or *chiral representation* of the gamma matrices:

$$\gamma^{\mu} = \begin{pmatrix} 0 & \sigma^{\mu}_{A\dot{B}} \\ \overline{\sigma}^{\mu\dot{A}B} & 0 \end{pmatrix} \tag{2.61}$$

We will now rewrite every object carrying Lorentz indices into objects with spinor indices, similar to the color-flow decomposition were we have replaced gluon indices a with quark-antiquark indices $i\bar{j}$. Since co- and contravariant indices make a difference, we define the variance of Lorentz indices to be contravariant for edge-like Feynman rules E (i.e. polarization vectors, propagrators) and covariant for vertex-like Feynman rules V. Every vertex is connected with an edge, so we may write:

$$V_{\mu}E^{\mu} = V_{\mu}g^{\mu\nu}E_{\nu} = V_{\mu}\left(\frac{1}{\sqrt{2}}\sigma^{\mu}_{A\dot{B}}\right)\left(\frac{1}{\sqrt{2}}\overline{\sigma}^{\nu\dot{B}A}\right)E_{\nu}. \tag{2.62}$$

In summary, we replace a covariant index μ by spinor indices $A\dot{B}$ by applying the pauli matrix $\frac{1}{\sqrt{2}}\sigma^{\mu}_{A\dot{B}}$, a contravariant index by applying $\frac{1}{\sqrt{2}}\overline{\sigma}^{\mu}_{\dot{A}B}$.

The objects mentioned above rewrite to:

$$\frac{1}{\sqrt{2}}\sigma^{\mu}_{A\dot{B}}\,\gamma_{\mu} = \sqrt{2}\begin{pmatrix} 0 & \epsilon_{AC}\epsilon_{\dot{B}\dot{D}} \\ \delta_{A}{}^{D}\delta_{\dot{B}}{}^{\dot{C}} & 0 \end{pmatrix} \tag{2.63}$$

$$\frac{1}{\sqrt{2}}\sigma^{\mu}_{A\dot{B}}\,p_{\mu} = \frac{1}{\sqrt{2}}p_{A\dot{B}} \tag{2.64}$$

$$\frac{1}{\sqrt{2}}\sigma^{\mu}_{A\dot{B}}\frac{1}{\sqrt{2}}\sigma^{\nu}_{C\dot{D}}\,g_{\mu\nu} = \epsilon_{AC}\epsilon_{\dot{B}\dot{D}} \tag{2.65}$$

$$\frac{1}{\sqrt{2}}\overline{\sigma}^{\dot{B}A}_{\mu}\frac{1}{\sqrt{2}}\overline{\sigma}^{\dot{D}C}_{\nu}\,g^{\mu\nu} = \epsilon^{AC}\epsilon^{\dot{B}\dot{D}} \tag{2.66}$$

$$p^{\mu}\gamma_{\mu} = \begin{pmatrix} 0 & p_{A\dot{B}} \\ p^{\dot{A}B} & 0 \end{pmatrix} \tag{2.67}$$

With these intermediate results, one rewrites the Feynman rules. The color-less gluon-three-vertex (Equation 2.17) reads in spinor notation:

$$V_{\mathrm{ggg}} = \frac{i}{\sqrt{2}}\Big(\epsilon_{AC}\epsilon_{\dot{B}\dot{D}}\left(p_2-p_1\right)_{E\dot{F}} + \epsilon_{CE}\epsilon_{\dot{D}\dot{F}}\left(p_3-p_2\right)_{A\dot{B}} + \epsilon_{EA}\epsilon_{\dot{F}\dot{B}}\left(p_1-p_3\right)_{E\dot{F}}\Big). \tag{2.68}$$

The vertex now contains spinor metrics ϵ and bispinors $p_{A\dot{B}}$

$$p_{A\dot{B}} = \begin{pmatrix} p^0+p^3 & p^1-ip^2 \\ p^1+ip^2 & p^0-p^3 \end{pmatrix}. \tag{2.69}$$

Rewriting the gluon-four-vertex and the gluon-propagator is now straightforward:

$$V_{\mathrm{gggg}} = i\Big(2\epsilon_{AE}\epsilon_{\dot{B}\dot{F}}\epsilon_{CG}\epsilon_{\dot{D}\dot{H}} - \epsilon_{AG}\epsilon_{\dot{B}\dot{H}}\epsilon_{CE}\epsilon_{\dot{D}\dot{F}} - \epsilon_{AC}\epsilon_{\dot{B}\dot{D}}\epsilon_{EG}\epsilon_{\dot{F}\dot{H}}\Big) \tag{2.70}$$

$$P_{\mathrm{g}} = -\frac{i}{k^2}\epsilon^{\dot{B}\dot{D}}\epsilon^{AC} \tag{2.71}$$

The antiquark-quark-gluon vertex and the quark/antiquark-propagator a more complicated, because they involve spinors of different types:

$$V_{\overline{q}q\mathrm{g}} = i\sqrt{2}\begin{pmatrix} 0 & \epsilon_{AC}\epsilon_{\dot{B}\dot{D}} \\ \delta_{A}{}^{D}\delta_{\dot{B}}{}^{\dot{C}} & 0 \end{pmatrix} \tag{2.72}$$

$$P_{\overline{q},q} = \frac{i}{p^2-m^2}\begin{pmatrix} m\delta_{A}{}^{B} & p_{A\dot{B}} \\ p^{\dot{A}B} & m\delta^{\dot{A}}{}_{\dot{B}} \end{pmatrix}. \tag{2.73}$$

Note that in the massless case the fermion-propagator is diagonal, i.e. one may treat both spinors independent from each other. This is also reflected by the structure of

the dirac spinors which decompose into two Weyl-spinors in the massless case:

$$\Psi(p) = \begin{pmatrix} \phi_A(p) \\ \psi^{\dot{A}}(p) \end{pmatrix} \equiv \begin{pmatrix} p_A \\ p^{\dot{A}} \end{pmatrix} \tag{2.74a}$$

$$\overline{\Psi}(p) = \begin{pmatrix} \psi^A(p) & \phi_{\dot{A}}(p) \end{pmatrix} \equiv \begin{pmatrix} p^A & p_{\dot{A}} \end{pmatrix} \tag{2.74b}$$

2.4 The Helicity Method

It is typical for analytical calculations of Feynman diagrams that explicit expressions for polarization vectors and dirac spinors are not needed. Instead, one uses the spin sum identities

$$\sum_s u_s(p)\overline{u}_s(p) = \not{p} + m \tag{2.75}$$

$$\sum_s v_s(p)\overline{v}_s(p) = \not{p} - m \tag{2.76}$$

and helicity sums in Equation 2.4 **after** squaring the matrix element to get rid of these objects.

This is neither possible nor desirable in numerical computations. Therefore, we need explicit formulas for polarization states for massless spin-1 bosons with momentum p and polarization $\lambda = \pm 1$ which satisfy the following identities (λ is not summed over):

$$\begin{aligned} p \cdot \varepsilon_\lambda(p) &= 0 & \varepsilon_\lambda(p) \cdot \varepsilon_\lambda(p) &= 0 \\ \varepsilon_{-\lambda}(p) &= \left(\varepsilon_\lambda(p)\right)^* & \varepsilon_\lambda(p) \cdot \varepsilon_{-\lambda}(p) &= -1 \end{aligned} \tag{2.77}$$

We use the following expression for polarization states[12, 16, 26]:

$$\varepsilon_+^\mu(p,q) = \frac{1}{\sqrt{2}} \frac{\langle q-|\gamma^\mu|p-\rangle}{\langle q-|p+\rangle} \tag{2.78}$$

$$\varepsilon_-^\mu(p,q) = \frac{1}{\sqrt{2}} \frac{\langle q+|\gamma^\mu|p+\rangle}{\langle p+|q-\rangle} \tag{2.79}$$

which use the following defintions:

$$|p\pm\rangle = \frac{1}{2}(1 \pm \gamma_5)u(p) \qquad \langle p\pm| = \bar{u}(p)\frac{1}{2}(1 \mp \gamma_5). \tag{2.80}$$

This special choice of olarization vectors lead to a simplification of analytical calculations[27]. We are interested in this choice because it allows us to rewrite the polarization vectors to bispinors in the Weyl–van der Werden formalism:

$$\varepsilon_+^{\dot{B}A}(p,q) = \frac{p^{\dot{B}} q^A}{q^C p_C} \tag{2.81}$$

$$\varepsilon_-^{\dot{B}A}(p,q) = \frac{q^{\dot{B}} p^A}{p_{\dot{A}} q_{\dot{A}}} \tag{2.82}$$

The momenta q^μ are arbitrary four-momenta and determine the gauge one uses.

We now have rewritten the polarization vectors into objects made of weyl-spinors. The polarization states for massless fermions are trivially rewritten into spinors (see Equation 2.74). For massive fermions it is possible to derive similar formulas[16, 28].

2.5 Multiple Gauge Groups

Theoretical models such as the standard model do not consist of a single gauge group but of two or even more. In the following we will discuss the important properties of such a model using the example of QCD coupled with QED. Though we specialize to a specific model, much will also apply to the more general case of two coupled Yang-Mills gauge theories (one can look at QED as a trivial example of a Yang-Mills theory).

We start with a look at the set of vertex Feynman rules in Appendix C. There is only one vertex which provides a mixing between the two theories: the quark-antiquark-photon vertex, which is due to the fact that quarks also carry electric charge.

2.5.1 Coupling constants

Let us now take a closer look at the coupling constants of leading-order matrix elements. We define the number of particles n to be the sum of every QCD and QED particle:

$$n = n_\gamma + n_\ell + n_{\overline{\ell}} + n_{\mathrm{g}} + n_q + n_{\overline{q}}. \tag{2.83}$$

Because our model has two different coupling constants, e and g, the factorization[2] of coupling constant (see Equation 2.1) will no longer hold. Instead, the the leading-order matrix element $\mathscr{M}^{(0)}$ will be a sum of gauge-invariant matrix elements $\mathscr{M}^{(0,k)}$

[2] This is a nice property which enables us to carry out numerical computations without coupling constants. Instead, they are multiplied at the end of the computation to the result, see Equation 2.1.

for which the coupling constants factorize:

$$\mathscr{M}^{(0)} = \sum_{k=k_{\min}}^{n-2} e^k \left(\frac{g}{\sqrt{2}}\right)^{n-(2+k)} \mathscr{M}^{(0,k)}. \tag{2.84}$$

Note that the QED coupling constant is not accompanied with a factor $1/\sqrt{2}$, because QED is abelian and thus the color matrices have to be replaced with ones.

The gauge-invariance of $\mathscr{M}^{(0,k)}$ follows from the gauge invariance of $\mathscr{M}^{(0)}$. One interpretes $e^k g^{n-(2+k)}$ as the basis of a polynomial vector space and $\mathscr{M}^{(0,k)}$ as elements in that vector space. Since the basis is linear independent, $\mathscr{M}^{(0,k)}$ must be gauge-invariant to make $\mathscr{M}^{(0)}$ invariant.

The smallest power $e^{k_{\min}}$ is determined by the number of particles coupling to QED only, i.e. the number of external photons n_γ and lepton-antilepton-pairs n_ℓ,

$$k_{\min} = n_\gamma + n_\ell. \tag{2.85}$$

By making use of the fact that QCD couples much stronger than QED, $g \gg e$, one can approximate the full leading-order matrix element by

$$\mathscr{M}^{(0)} = e^{k_{\min}} \left(\frac{g}{\sqrt{2}}\right)^{n-(2+k_{\min})} \left\{\mathscr{M}^{(0,k_{\min})} + O\left(\frac{e}{g}\right)\right\}, \tag{2.86}$$

where one can again factorize the coupling constants from matrix elements.

2.5.2 Matrix elements

The matrix elements for two coupled gauge groups 1 and 2, for which the color decomposition holds, are computed with the following formula[29]:

$$\mathscr{M} = \sum_{P_i^1} \sum_{P_j^2} C^1\left(P_i^1\right) C^2\left(P_j^2\right) A(\underbrace{1,2,\ldots,n^1}_{\text{particles in 1}};\underbrace{n^1+1,\ldots,n^1+n^2}_{\text{particles in 2}}); \tag{2.87}$$

P_i^1 denotes the noncyclical permutations of pseudo particles belonging gauge group 1, P_j^2 denotes the permutations of pseudo particles belonging to gauge group 2, **including** cyclical permutations. If there are no particles of gauge group 2 involved, this formula is identical to that one in Equation 2.53. Note that we assume that there are always particles of 1 present. C^1 is the color factor of gauge group 1. For the special case of QCD this was discussed in subsection 2.2.5. For QED the color factor is

$$C^{\text{QED}} = 1, \tag{2.88}$$

because the photon does not carry charges.

To illustrate the Equation 2.87 for SU(3)×U(1) (QCD with QED), let us look at a process involving a quark-antiquark-pair, two gluons, and two photons. The particles are identified with their indices:

$$0 \to \overline{q}_1 q_2 \mathrm{g}_3 \mathrm{g}_4 \gamma_5 \gamma_6 \tag{2.89}$$

The partial amplitudes that have to be computed are:

$$A\left(\overline{q}_1, q_2, \mathrm{g}_3, \mathrm{g}_4; \gamma_5, \gamma_6\right) \tag{2.90}$$
$$A\left(\overline{q}_1, q_2, \mathrm{g}_3, \mathrm{g}_4; \gamma_6, \gamma_5\right) \tag{2.91}$$
$$A\left(\overline{q}_1, q_2, \mathrm{g}_4, \mathrm{g}_3; \gamma_5, \gamma_6\right) \tag{2.92}$$
$$A\left(\overline{q}_1, q_2, \mathrm{g}_4, \mathrm{g}_3; \gamma_6, \gamma_5\right). \tag{2.93}$$

These are the four partial amplitudes belonging to different color factors. Note that we have fixed the first the particle instead of the last — this is possible, since the partial amplitudes are invariant under cyclic permutations[30]. In this case the amplitudes denote every amplitude which contribute to the same color factor:

$$A\left(1, \ldots, n_{\mathrm{QCD}}; n_{\mathrm{QCD}}+1, \ldots, n\right) = \sum_{\text{shuff}} a\left(1, \ldots, n\right), \tag{2.94}$$

where the sum runs over every permutation of particles that does not change the positions between particles belonging to the same gauge group. We call these permutation “shuffle permutations”. There are exactly

$$n^{\text{shuffle}} = \frac{\left(n^{\mathrm{QCD}} - 1 + n^{\mathrm{QED}}\right)!}{\left(n^{\mathrm{QCD}} - 1\right)! n^{\mathrm{QED}}!} \tag{2.95}$$

shuffle permutations if n^{QCD} denotes the number of quarks and gluons and n^{QED} the number of leptons and photons. The subamplitudes are computed using Berends-Giele-type recursion relations.

Note that the shuffle permutations complete the two permutations P^1 and P^2 of subsets of particles to the non-cyclical permutation of every pseudo particle. This can be seen in the number of total summations, which is

$$n^{\text{shuffle}} \left(n^{\mathrm{QCD}} - 1\right)! n^{\mathrm{QED}}! = (n-1)! \tag{2.96}$$

the usual number if n denotes the number of pseudo particles. Thus an alternative definition of Equation 2.87 reads

$$\mathscr{M} = \sum_P C^1\left(P\right) C^2\left(P\right) A\Big(\underbrace{1, 2, \ldots, n^1}_{\text{particles in 1}}; \underbrace{n^1+1, \ldots, n^1+n^2}_{\text{particles in 2}}\Big), \tag{2.97}$$

P being the pseudo particle permutation (excluding the first particle) and the partial amplitudes A defined as usual (no summing over shuffle permutations). This formulas makes clear that the introduction of additional gauge groups does not add a significant change to the method. However, we will use Equation 2.87 in an implementation because the amplitudes a with the same color factor are summed up and thus the computation is faster.

Chapter 3

Implementation

In this chapter we describe the implementation of a matrix element computation library in C++[31–34] using the color-flow decomposition[11] for QCD[12], QED and QCD with photons. We focus on an algorithmic description of concepts of the previous chapter. Additionally, we discuss the run-time complexity of the algorithms involved and present optimization techniques.

3.1 Basic Building Blocks

The basic building blocks relevant for the computation of curents are the spinors and Feynman rules described in section 2.3. We need to implement at least six different objects carrying two spinor indices at most:

$$p_{A\dot{B}} \quad p^{\dot{B}A} \quad p_A \quad p_{\dot{A}} \quad p^A \quad p^{\dot{A}} \tag{3.1}$$

This is done by saving the spinor's components in vectors and matrices, respectively. Vertices and Propagators are objects with more than two spinor indices, but one does not need to implement those if the contraction with subcurrents is made. For example, a gluon current brances into three subcurrents which are connected by a four-gluon-vertex. The three subcurrents are immediately contracted with the vertex. The result is a bispinor with two indices.

3.1.1 Spinors

In section 2.3 we have shown that every polarization vectors may be expressed with spinors. Theses spinors are implemented using the following formulas,

$$p_B = |p+\rangle = \frac{1}{\sqrt{|p_+|}} \begin{pmatrix} -p_\perp^* \\ p_+ \end{pmatrix} \tag{3.2}$$

$$p^{\dot{B}} = |p-\rangle = \frac{\text{sign}\left(p^0\right)}{\sqrt{|p_+|}} \begin{pmatrix} p_+ \\ p_\perp \end{pmatrix} \tag{3.3}$$

$$p_{\dot{A}} = \langle p+| = \frac{\text{sign}\left(p^0\right)}{\sqrt{|p_+|}} \begin{pmatrix} -p_\perp & p_+ \end{pmatrix} \tag{3.4}$$

$$p^A = \langle p-| = \frac{1}{\sqrt{|p_+|}} \begin{pmatrix} p_+ & p_\perp^* \end{pmatrix} \tag{3.5}$$

which use the definitions for the light cone-coordinates:

$$p_+ = p^0 + p^2 \tag{3.6}$$

$$p_\perp = p^3 + i\,p^1 \tag{3.7}$$

which assumes p to be a lightlike momentum. Note that the definitions of the spinors with undotted indices are related by the spinor metric (Equation 2.60) as well as the spinors with dotted indices. The additional signs and absolute values of p_+ provide a proper analytical continuation for momenta with "negative energies" $p^0 < 0$, which arise when crossing symmetry is used to switch incoming particles to the outgoing channel. The light-cone coordinates are chosen such that p_+ is always different from zero, assuming that the beam axis (incoming momenta) are taken to lie on the 3-axis.

3.1.2 Reference Momenta

The polarization vectors are implemented using Equation 2.79. By doing so, one has to make sure that the denominator of $\varepsilon_+^\mu(p,q)$ does not introduce an unphysical pole if $\langle p+|q-\rangle = 0$. The pole is unphysical since q is arbitrary. Therefore, apart from being lightlike, the following contraints are imposed to q in order to obtain a parametrization $q \equiv q\left(p\right)$:

$$\langle p+|q-\rangle := s \neq 0 \tag{3.8}$$

The spinor product must be different from zero to avoid the pole and its length "large enough" and to ensure numerical stability. The spinor-product is defined to be

$$s = \frac{\left[q^3\left(p^0+p^2\right)-p^3\left(q^0+q^2\right)\right]+i\left[p^1\left(q^0+q^2\right)-q^1\left(p^0+p^2\right)\right]}{\sqrt{\left|\left(p^0+p^2\right)\left(q^0+q^2\right)\right|}}. \tag{3.9}$$

A simple choice[23] of q is

$$q^\mu = \left(p^0, -\vec{p}\right) \tag{3.10}$$

which avoids the pole for every momenta, assuming that

$$p^0 - p^2 \neq 0, \tag{3.11}$$

which we assume to hold true, since the beam axis is taken along the 3-axis.

3.2 Color-Flow Decomposition

Let us repeat the most important ingredients of our color-ordered algorithm to compute matrix elements $\mathscr{M}$:

$$\mathscr{M} = \sum_i C_i A_i \tag{3.12a}$$

$$\mathscr{M}^* = \sum_j C_j A_j^*. \tag{3.12b}$$

The objects C_i, C_j are called color-factors, A_i, A_j are the partial amplitudes computed with Berends-Giele-type recursion relations. The summations run over pseudo particle permutations (see subsection 3.3.1), quark permutations and the various cluster configurations (see subsection 3.3.2). The color factors C_i are constructed using cluster color factors c_{ij}:

$$C_i = \left(-\frac{1}{N}\right)^{n_{\text{cluster}}-1} \cdot \prod_{j=1}^{n_{\text{cluster}}} c_{ij}. \tag{3.13}$$

The c_{ij} are computed using the algorithm described in subsection 3.4.1. To compute a color-summed amplitude, one inserts the color projectors P_k in between the matrix elements:

$$\sum_{\text{color}} |\mathscr{M}|^2 = \left(\sum_j C_j A_j\right)\left(\prod_{k=1}^{n} P_k\right)\left(\sum_i C_i A_i\right). \tag{3.14}$$

The projector for particle k of type $\overline{q}$, q, g are defined as:

$$P_{q_k} = \delta_{i_k j_k} \tag{3.15}$$

$$P_{\overline{q}_k} = \delta_{\bar{\imath}_k \bar{\jmath}_k} \tag{3.16}$$

$$P_{\mathrm{g}_k} = \delta_{i_k j_k}\delta_{\bar{\imath}_k \bar{\jmath}_k} - \frac{1}{N}\delta_{i_k \bar{\jmath}_k}\delta_{\bar{\imath}_k j_k}. \tag{3.17}$$

It is possible to rewrite the color-summed squared matrix element to

$$\sum_{\text{color}} |\mathscr{M}|^2 = \sum_j \sum_i A_j \left(C_j C_i \prod_{k=1}^{n} P_k \right) A_i. \tag{3.18}$$

The indices i, j still denote the particle configurations described in section 3.3. We will present an algorithm which generates these particle configuration in a successive fashion, which allows us to map these configurations to numbers — the first configuration is mapped to number 0, the second to 1 and so on. Therefore, it is possible to define two new objects which makes the color factorization explicit:

- The *color matrix M* with elements

$$M_{ij} = C_j C_i \prod_{k=1}^{n} P_k \tag{3.19}$$

  and

- the *amplitude vector A* with elements A_i which are computed using the Berends-Giele-type recursion relations.

With these objects the squared matrix elements summed over colors reads

$$\sum_{\text{color}} |\mathscr{M}|^2 = \sum_{j=1}^{n_{\text{confs}}} \sum_{i=1}^{n_{\text{confs}}} A_j M_{ji} A_i. \tag{3.20}$$

n_{confs} is the number of particle configurations which are determined by counting the number of configurations the algorithm in the next section yields (in the all gluons case n_{confs} is $(n-1)!$, but for the case of arbitrary quarks we did not found an analytical formula).

The implementation of the matrix element computation thus breaks down into the following tasks:

- The implementation of the particle configuration algorithm (see section 3.3,
- the color matrix (in section 3.4) and
- the amplitude vector (see section 3.5).

3.3 Particle Configurations

The sum appearing in Equation 3.18 runs over the index i denoting the *particle configurations*. This notion includes the pseudo particle permutations and fermion permutations, as well as the color cluster configurations. In this section we will present an algorithm for the successive generation of every particle configuration.

The algorithm will be needed in at least two places: the calculation of amplitudes and the calculation of color factors. Thus, it is preferable to put it in a class so we can instantiate this class and use it everywhere it is needed. The class will be used in way similar to the following listing (the algorithm is implemented in the member function `next()`):

```
1 // create object which holds information about the process
2 process_info process("g g > g g");
3
4 // create particle configuration object for the process
5 particle_conf conf(process);
6
7 do {
8     // loop over every particle configuration, e.g. to compute the
9     // partial amplitudes
10 } while (conf.next());
```

In technical terms, the particle configurations are implemented using the iterator-pattern[35].

3.3.1 Particle Permutations

In short, the particle permutations are the set of every permutation of external particles that respects the cyclical ordering, namely

- permutations of pseudo particles, i.e. permutations of bosons with antifermion-fermion-pairs, and
- permutations of fermions. Since the fermion permutation sign is needed in the computation of matrix-elements (see section 2.1), we need to memorize it.

To generate these permutations, we will first permute the pseudo particles, excluding the first particle. In the resulting permutation, we will permute the fermions.

This is not a trivial task[1], because a pseudo particle permutation may include a transposition of a antifermion/fermion-pair with a gluon which is not straightfor-

[1] A lesson we learned from hours of debugging

ward to implement. To achieve that, we will first generate a vector of indices representing the pseudo particles, the *pseudo particle assignment*. As an example, consider the process

$$0 \to \overline{q}_0 q_1 \overline{q}_2 q_3 g_4. \tag{3.21}$$

The indices will be used to denote the particles (and their momenta, helicities, ...). The pseudo particle assignment for this process reads

1	3	4	.	(3.22)

These indices simply point to the fermions and bosons and skip antifermions. Since we assume cyclical ordering, we are able to reconstruct the antifermion indices.

We are now able to permute the pseudo particle. For example, the *pseudo particle permutation*

0	2	1	(3.23)

tells us to transpose the last pseudo particles. Using the pseudo particle assignment, we can reconstruct the particle permutation to

0	1	4	2	3	.	(3.24)

On top of this permutation, we may permute the fermions. Since the order in that the fermions appear depends on the pseudo particle permutation, we need to extract the *fermion indices* first. In our example, this is

1	3	.	(3.25)

A possible *fermion permutation* is

1	3	4	;	(3.26)

this tells us to swap the indices 1 and 3. The final particle permutation is

0	3	4	2	1	.	(3.27)

In summary, we need these four structures:

1. The *pseudo particle assignment* containing the indices of fermions and bosons,
2. the *pseudo particle permutation* which permutes the indices in pseudo particle assignment,

3. the *fermion indices*, containing the indices of fermions in the order they appear after applying the pseudo particle permutation (which thus needs to be updated after every pseudo particle permutation), and

4. the *fermion permutation* permuting the fermion indices.

Using this information we can now describe the general algorithm for the successive generation of particle permutations:

1. Generate the next non-cyclical pseudo particle permutation. That is done by computing the next permutation in *pseudo particle permutation*, excluding the first element. "Next" is understood in the sense of lexicographical ordering[2].

2. If the previous permutation was the last one (e.g. "| 1 | 3 | 4 |" is a last permutation, because the first element is not permuted), the permutation is reset to identity and the next fermion permutation is computed (proceed with item 3). Else, compose the particle permutation (proceed with item 4).

3. Compute the next permutation of elements in the *fermion permutation*. If the previous fermion permutation was the last one, reset the *fermion permutation* to identity. In that case, also the previous *particle permutation* was the last one. At the end of the algorithm signal that (e.g. by returning `false` to finish the loop in the listing above).

4. Clear the contents of *fermion indices*. Iterate over the elements of the *pseudo particle permutation* and look up their particle index i with the *pseudo particle assignment*.

   - If the particle is a fermion, add the particle index of the preceeding antifermion and i to *particle permutation*. Additionally, add i to *fermion indices*.
   - If the particle is a boson, add i to *particle permutation*.

5. Finally, permute the *fermion indices* using the *fermion permutation* and assign these indices to particle permutation where the indices denotes fermions.

[2]The ordering is important because it enables use to move from an existing permutation to the permutation with the next higher order. Starting from the permutation with the lowest order one can reach every possible permutation.

This method produces flavor-violating pairs if different fermion flavors are present (e.g. the following particle permutation: $\bar{u}d\bar{d}u$) which in turn leads to flavor violating currents. If these are not needed (e.g. in QCD or QED) one begins again with item 1 and repeats the procedure until an appropriate configuration is found or the end is reached.

The fermion permutation sign is obtained by counting the fermion permutations. Because they are constructed lexicographically, the sign changes every second permutation:

0	1	2	$\longmapsto$	$+1$ (3.28)
0	2	1	$\longmapsto$	-1 (3.29)
1	0	2	$\longmapsto$	-1 (3.30)
1	2	0	$\longmapsto$	$+1$ (3.31)
2	0	1	$\longmapsto$	$+1$ (3.32)
2	1	0	$\longmapsto$	-1 (3.33)

3.3.2 Color Clusters

To complete the discussion of the particle configurations, we need an algorithm generating the possible cluster configurations. We will represent a cluster configuration using a list of cluster numbers. This list will be called *cluster assignment*, because for each particle index it will store a cluster number.

For our example of quark scattering with a gluon, we could have the following cluster configuration:

$$\underbrace{\bar{u}_0 u_1 g_4}_{1}\underbrace{\bar{d}_2 d_3}_{2} \tag{3.34}$$

The particle permutation is

0	1	4	2	3

and the cluster assignment

1	1	2	2	1

because the first, the second, and the last particle belongs to cluster 1 and the remaining particles to cluster 2. Note that the cluster assignment is not permuted, i.e.

if one writes down the cluster assignment and the particle permutation underneath, the numbers in a column do not correspond to a single particle.

We defined color clusters as a set of external particles which are disconnected (by color) from every other particle. These clusters are connected by colorless U(1)-gluons which couple to quarks, but not to gluons itself. Thus, there are as many cluster as quarks, but at least one (every particle in a single cluster).

Since the cluster numbering has no physical meaning, we define that cluster "1" always begins with the first particle. The cluster with the next higher number must always start after after a lower number. As an example, take the following process with three pairs:

$$0 \to \overline{\mathrm{u}}_0 \mathrm{u}_1 \overline{\mathrm{d}}_2 \mathrm{d}_3 \overline{\mathrm{s}}_4 \mathrm{s}_5 \overline{\mathrm{c}}_6 \mathrm{c}_7 \tag{3.35}$$

A valid cluster assignment for the identity particle permutation is

1	1	2	2	3	3	4	4

but not

1	1	3	3	2	2	4	4

because cluster "3" starts before "2". We declare those configurations invalid because they only differ by numbering but not in physics — in both examples above every quark/antiquark-pair is in a different cluster.

Since clusters may contain several quark/antiquark-pairs, it is possible that clusters may be contained:

1	1	2	2	1	1	1	1

.

It is not possible that clusters may partially overlap:

1	1	2	2	1	1	2	2

.

because it is not possible to draw the corresponding (color-ordered) tree.

To generate cluster assignments in a successive fashion, we need the following structures:

- The number of current clusters n_{cluster}:

$$n_{\text{cluster}} \in \Big[1, \max\big(n_q - 1, 1\big)\Big), \tag{3.36}$$

- the *starting indices*
$$\left(n_2^{\text{start}}, \ldots, n_{n_{\text{cluster}}}^{\text{start}}\right) \tag{3.37}$$
containing the indices where each cluster begins ($n_1^{\text{start}} := 0$) and

- the *cluster lengths*
$$\left(n_1^{\text{cluster}}, \ldots, n_{n_{\text{cluster}}}^{\text{cluster}}\right) \tag{3.38}$$
containing the number of (pseudo) particles in each cluster. The sum of every cluster length must, of course be the number of (pseudo) particles:
$$\sum_{i=1}^{n_{\text{cluster}}} n_i^{\text{cluster}} = n_q + n_{\text{g}}. \tag{3.39}$$
Thus, the cluster lengths are generated using an integer-partition-like algorithm. The difference to integer partitions is that order matters, e.g. $2+1 = n_q + n_{\text{g}}$ means two particles in the first cluster and one in the second - this differs from the reversed case, which must also be taken into account.

The starting indices n_i^{start} are generated using a tuple of numbers m_i^{start} which satify the following conditions:

$$m_i^{\text{start}} \leq m_{j+1}^{\text{start}} \tag{3.40}$$
$$0 \leq m_i^{\text{start}} \leq 1 - i + \sum_{k=1}^{i-1} n_k^{\text{cluster}} \tag{3.41}$$
$$n_i^{\text{start}} = m_i^{\text{start}} + i - 1 \tag{3.42}$$

These numbers m_i^{start} are easier to generate than the n_i^{start}.

For a fixed particle permutation (see subsection 3.3.1), the algorithm proceeds as follows:

1. Generate the next tuple of *starting indices*. If the previous tuple of indices was the last one, or every particle is in a single cluster ($n_{\text{cluster}} = 1$), proceed with item 2. In any case, assign particles (continue with item 4).

2. Generate the next tuple of *cluster lengths*. If the the previous tuple was the last one or every particle is in a single cluster, proceed with item 3. In any case, reset the starting indices.

3. Increase the number of clusters. If the previous number of clusters was the maximum, set the number of cluster to one and reset the cluster assignment (everying in a single cluster). The cluster assignment now has reached the end - every configuration was generated. In any case, reset the *cluster lengths*.

#	Permutation	Sign	Cluster	#	Permutation	Sign	Cluster
0	(01)(23)4	+	(11)(11)1	6	(03)(21)4	-	(11)(11)1
1	(01)(23)4	+	(11)(22)1	7	(03)(21)4	-	(11)(22)1
2	(01)(23)4	+	(11)(22)2	8	(03)(21)4	-	(11)(22)2
3	(01)4(23)	+	(11)1(11)	9	(03)4(21)	-	(11)1(11)
4	(01)4(23)	+	(11)1(22)	10	(03)4(21)	-	(11)1(22)
5	(01)4(23)	+	(11)2(22)	11	(03)4(21)	-	(11)2(22)

Table 3.1: *Possible external particle configurations for a process with two pairs of quarks (in the table marked with braces) and a gluon, e.g the process* $0 \to \overline{q}_0 q_1 \overline{q}_2 q_3 g_4$. *The entry "Sign" denotes the fermion permutation sign and "Cluster" the cluster of each permuted particle. For example, line # 11 is interpreted as being the permutation* $\overline{q}_0 q_3 g_4 \overline{q}_2 q_1$ *which is equal to line # 5 up to a fermion-permutation. This difference is reponsible for the relative sign. Particles 0 and 3 are connected by color as well as particles 1, 2 and 4, but among each other those particles are disconnected by color (i.e. connected by a U(1)-gluon).*

4. Assign pseudo particles to clusters. This is done using the following procedure:
   (a) Start with the last cluster.
   (b) Assign pseudo particles beginning at the index stored in *starting indices* for the current cluster. Assign as much particles as stored in *cluster lengths* which are not already assigned to a cluster. The last condition permits cluster with a lower number to contain cluster(s) with a higher number.
   (c) Repeat procedure in item 4b for the next cluster with lower cluster number. Stop if every pseudo particle is assigned.
5. Check if every cluster in the current configuration includes at least one antiquark/quark-pair. If this is not the case, stop and start the procedure again with item 1.
6. By using the pseudo particle assignment computed in item 4, assign the (real) particles to clusters.

For a comprehensive example, including both particle and cluster configurations, look at Table 3.1.

3.3.3 Shuffle Permutations

The coupling of two gauge groups (see section 2.5) requires the shuffle permutations of particles belonging to different gauge groups. Shuffling permutations are generated using the following algorithm:

1. Assume that the initial particle order is such that every particle in gauge group 1 is left, and particles in gauge group 2 right, e.g.:

$$\underbrace{\boxed{(\overline{q}q)_0}\boxed{g_2}\boxed{g_1}}_{\text{QCD particles}}\underbrace{\boxed{\gamma_3}\boxed{\gamma_5}\boxed{\gamma_4}}_{\text{QED particles}} \tag{3.43}$$

2. Fill a list with "1" for every pseudo particle in gauge group 1, and "2" for every pseudo particle in 2:

$$\boxed{1}\boxed{1}\boxed{1}\boxed{2}\boxed{2}\boxed{2} \tag{3.44}$$

3. Generate the next permutation in the newly created list:

$$\boxed{1}\boxed{1}\boxed{2}\boxed{1}\boxed{2}\boxed{2} \tag{3.45}$$

4. Overwrite the list with the permuted pseudo particle indices, i.e. overwrite "1"s with $0,2,1$ and "2"s with $3,5,4$:

$$\boxed{(\overline{q}q)_0}\boxed{g_2}\boxed{\gamma_3}\boxed{g_1}\boxed{\gamma_5}\boxed{\gamma_4} \tag{3.46}$$

By repeating the procedure we successively generate shuffle permutations and thus configurations belonging to the same color factor which can be added up.

Since QED does not have a color factor, one could also add up the particle configurations which involve the QED pseudo particle permutations.

3.4 Color Matrix

The color matrix is defined to be

$$M_{ij} = C_j^* C_i \prod_{k=1}^{n} P_k, \tag{3.47}$$

with C_i and C_j the color factors and P_k the color projectors. The color factors are computed using the cluster color factors c_{ij},

$$C_i = \left(-\frac{1}{N}\right)^{n_{\text{cluster}}-1} \cdot \prod_{j=1}^{n_{\text{cluster}}} c_{ij} \tag{3.48a}$$

$$C_j^* = \left(-\frac{1}{N}\right)^{n_{\text{cluster}}-1} \cdot \prod_{k=1}^{n_{\text{cluster}}} c_{ik}^* \tag{3.48b}$$

The projectors are defined to be

$$P_{q_k} = \delta_{i_k j_k} \tag{3.49}$$

$$P_{\overline{q}_k} = \delta_{\bar{\imath}_k \bar{\jmath}_k} \tag{3.50}$$

$$P_{\mathrm{g}_k} = \delta_{i_k j_k}\delta_{\bar{\imath}_k \bar{\jmath}_k} - \frac{1}{N}\delta_{i_k \bar{\jmath}_k}\delta_{\bar{\imath}_k j_k}. \tag{3.51}$$

Note that the gluon projector is a sum; multiple gluons in a process will thus yield a sum of strings of delta symbols with a power of $1/N$ and a possible sign in front for the color factor. Therefore, we will first describe the generation of cluster color factors and then how to evaluate the strings of deltas.

3.4.1 Generation of Cluster Color Factors

The cluster color factors are generated using the particle permutation and the cluster assignment. For each pseudo particle in cluster c we have to generate one Kronecker delta symbol. The rules are

1. For every gluon with particle index k which is followed by a particle belonging to the same cluster c with particle index l:

   $$\delta_{i_k \bar{\jmath}_l} \tag{3.52}$$

   If the gluon is the last particle in the cluster, the next particle is the first one in cluster c (modulo).

2. For every antiquark-quark-pair with indices k (antiquark) and l (quark) which is followed by a particle in the same cluster c with index m:

   $$\delta_{i_l \bar{\jmath}_m}. \tag{3.53}$$

   If the quark is the last particle in cluster c, the next particle is the first particle in cluster c.

For example, take the following process (numbers printed below denote the clusters-assignment):

g_0	$\overline{q}_1$	q_5	g_3	$\overline{q}_4$	q_2
1	2	2	2	1	1

(3.54)

There are two pseudo particles in cluster 1, namely g_0 and $\overline{q}_4 q_2$. In cluster 2 there are also two pseudo particles, $\overline{q}_1 q_5$ and g_3. The rules for cluster color factors yield:

$$c_{i1} = \delta_{i_0 \bar{j}_4} \delta_{i_2 \bar{j}_0} \tag{3.55a}$$

$$c_{i2} = \delta_{i_5 \bar{j}_3} \delta_{i_3 \bar{j}_1}. \tag{3.55b}$$

For the matrix element we also need the cluster color factors for the matrix element $\mathscr{M}^*$ which are obtained by simply putting the bar on the other index:

$$c^*_{i1} = \delta_{\bar{i}_0 j_4} \delta_{\bar{i}_2 j_0} \tag{3.56a}$$

$$c^*_{i2} = \delta_{\bar{i}_5 j_3} \delta_{\bar{i}_3 j_1} \tag{3.56b}$$

3.4.2 Representation of Kronecker-Delta Symbols

It remains to discuss how string of Kronecker deltas are evaluated. The speed of these computations is not crucial because the matrices are computed once for a process. Nevertheless, it is desirable to have fast color matrix computation routines in order to quickly perform tests when debugging the program.

The Kronecker-deltas we need are

$$\delta_{i_k j_l} \quad \delta_{i_k \bar{j}_l} \quad \delta_{\bar{i}_k j_l} \quad \delta_{\bar{i}_k \bar{j}_l}. \tag{3.57}$$

and contractions of them. There are four different types of indices: i and j for fermions, $\bar{\imath}$ and $\bar{\jmath}$ for antifermions. The color factors of gluons involve a delta with a fermion and an antifermion index. For each of these four indices there n different ones, n being the number of particles:

$$i_1, i_2, \ldots, i_n \tag{3.58a}$$

$$j_1, j_2, \ldots, j_n \tag{3.58b}$$

$$\bar{\imath}_1, \bar{\imath}_2, \ldots, \bar{\imath}_n \tag{3.58c}$$

$$\bar{\jmath}_1, \bar{\jmath}_2, \ldots, \bar{\jmath}_n. \tag{3.58d}$$

Alltogether, we need to represent Kronecker-deltas with $4n$ different indices. The order in which the indices appear does not make a difference. The only operation

we need to implement is contraction of two deltas. We can distinguish the following three cases:

1. The symbols do not share any index, e.g. $\delta_{i_1 i_2}\delta_{i_3 i_4}$. In that case "contraction" has nothing to do.

2. The symbols share a single index, e.g. $\delta_{i_1 i_2}\delta_{i_2 i_3} = \delta_{i_1 i_3}$.

3. The symbols are the identical, e.g. $\delta_{i_1 i_2}\delta_{i_2 i_1} = N$. This is a special case, since the result is a number. Note also, that this number can not be unambiguously defined with delta symbols, since $N = \delta_{i_1 i_1} = \delta_{i_2 i_2} = \cdots$.

We implemented the delta symbols with bitsets of $4n$ bits, each index having n bits of which one or no bit can be set. For example, the Kronecker delta symbol $\delta_{\bar{\imath}_1 \bar{\jmath}_0}$ is represented with the following bitset (bits are read from right to left):

$$\delta_{\bar{\imath}_1 \bar{\jmath}_0} = \underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{1}}_{\bar{\jmath}_0}\underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{0}}_{\bar{\imath}_1}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{j_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{i_k \text{ not set}}. \tag{3.59}$$

Another example is

$$\delta_{\bar{\imath}_1 i_2} = \underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\bar{\jmath}_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{0}}_{\bar{\imath}_1}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{j_k \text{ not set}}\underbrace{\boxed{0}\boxed{1}\boxed{0}\boxed{0}}_{i_2}. \tag{3.60}$$

The contraction of both deltas is

$$\delta_{\bar{\imath}_1 \bar{\jmath}_0}\delta_{\bar{\imath}_1 i_2} = \delta_{i_2 \bar{\jmath}_0}. \tag{3.61}$$

What we need is a bit-operation that has the same behavior of the contraction. This is (bitwise) exclusive-OR, which maps both bitsets to the desired result

$$\delta_{\bar{\imath}_1 i_2} = \underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{1}}_{\bar{\jmath}_0}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\bar{\imath}_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{j_k \text{ not set}}\underbrace{\boxed{0}\boxed{1}\boxed{0}\boxed{0}}_{i_2}. \tag{3.62}$$

Full contraction of a delta symbol is mapped to the zero-bitset which is thus defined to represent the number N:

$$\delta_{\bar{\imath}_1 i_2}\delta_{\bar{\imath}_1 i_2} = \underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\bar{\jmath}_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\bar{\imath}_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{j_k \text{ not set}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{i_k \text{ not set}}. \tag{3.63}$$

A problematic case is the contraction of uncontractible deltas, e.g. $\delta_{i_1 i_2}\delta_{i_3 i_4}$. In this case exclusive OR would yield an object with four indices, which does not make

sense. Thus, we have to check first if two deltas are contractible. We do this by applying bitwise AND to both bitsets and check if the result is different from zero. If this is the case, the bitsets share at least one bit (the deltas have at least one index in common) and contraction is possible.

Strings of delta symbols are fully contracted by the following algorithm:

1. Take the first uncontracted delta symbol.
2. Look for another delta which is contractible with the current one and contract.
3. If the current delta is not fully contracted, repeat item 2.
4. Repeat item 1 until every delta is fully contracted.

The result is N^m, $N = \sum_i \delta_{ii}$ and m the number of fully contracted deltas that remain at the end of the algorithm.

3.4.3 Color Matrix Examples

With the algorithms how to generate the color factors and how to evaluate the strings, one can now compute color matrices. In the following we show some simple examples of color matrics, for the gluon process gg → gg and for the quark process

$\bar{u}u \to \bar{d}d\bar{c}c$:

$$C_{gggg} = \begin{pmatrix} 81 & 9 & 9 & 9 & 9 & 9 \\ 9 & 81 & 9 & 9 & 9 & 9 \\ 9 & 9 & 81 & 9 & 9 & 9 \\ 9 & 9 & 9 & 81 & 9 & 9 \\ 9 & 9 & 9 & 9 & 81 & 9 \\ 9 & 9 & 9 & 9 & 9 & 81 \end{pmatrix} \tag{3.64}$$

$$C_{\bar{u}u\bar{d}d\bar{c}c} = \begin{pmatrix} 27 & -3 & -3 & -3 & \frac{1}{3} & 3 & -3 & -3 & -3 & \frac{1}{3} \\ -3 & 3 & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -3 & 3 & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \\ -3 & \frac{1}{3} & 3 & \frac{1}{3} & -\frac{1}{3} & -3 & \frac{1}{3} & \frac{1}{3} & 3 & -\frac{1}{3} \\ -3 & \frac{1}{3} & \frac{1}{3} & 3 & -\frac{1}{3} & -3 & \frac{1}{3} & 3 & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \\ 3 & -3 & -3 & -3 & \frac{1}{3} & 27 & -3 & -3 & -3 & \frac{1}{3} \\ -3 & 3 & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -3 & 3 & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \\ -3 & \frac{1}{3} & \frac{1}{3} & 3 & -\frac{1}{3} & -3 & \frac{1}{3} & 3 & \frac{1}{3} & -\frac{1}{3} \\ -3 & \frac{1}{3} & 3 & \frac{1}{3} & -\frac{1}{3} & -3 & \frac{1}{3} & \frac{1}{3} & 3 & -\frac{1}{3} \\ \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \end{pmatrix} \tag{3.65}$$

The four gluon process is special, since the off-diagonal entries are same — in general, this is not the case. However, we observed that diagonal values in gluon-processes are N^n.[3].

Another example is a color matrix for SU(3)×U(1), for example for the process $u\bar{u} \to gg\gamma\gamma$. Since there are two particle permutations for the photons and the color factor for U(1) always is 1, the matrix consists of 2×2 submatrices with the same entries:

$$C_{u\bar{u}gg\gamma\gamma} = \begin{pmatrix} \frac{64}{3} & \frac{64}{3} & -\frac{8}{3} & -\frac{8}{3} \\ \frac{64}{3} & \frac{64}{3} & -\frac{8}{3} & -\frac{8}{3} \\ -\frac{8}{3} & -\frac{8}{3} & \frac{64}{3} & \frac{64}{3} \\ -\frac{8}{3} & -\frac{8}{3} & \frac{64}{3} & \frac{64}{3} \end{pmatrix} \tag{3.66}$$

Note that the notion of a color matrix is not a new one; color matrices were already used in analytical calculations were one may diagonalize the matrices in order to simplify the color summationBarger, Stange, and Phillips [36]. However, on computers we do expect the diagonalization yield a performance boost.

[3] We have checked that formula up to $n = 8$

3.5 Partial Amplitudes

The partial amplitudes A_i are computed using Berends-Giele-type recursion relations. These recursion relations are similar to the Berends-Giele recursion relations for gluons (see Equation 2.36), but for QCD one has to add quark- and antiquark-currents. These currents branch into themselves and U(1)/U(N)-gluon-currents, the end condition is the corresponding polarization spinors. The gluon-currents may now also branch into antiquark-quark-currents. The current particle permutation determine which branchings are possible (if at all).

3.5.1 Identification of current types

The most simple recursion relations are those of the all-gluon-amplitudes, because they involve only one type of particles: the gluons itself. Thus, every subcurrent is a gluon current. In general, additional fermions and antifermions take part in the process, so that one has to answer the question of which type the subcurrent. As an example, consider a process with two gluons and three antiquark-quark-pairs:

$$0 \to \mathrm{g}_0 \mathrm{g}_1 \overline{q}_2 q_3 \mathrm{g}_4 \overline{q}_5 q_6 \tag{3.67}$$

Amongst others, that requires us to compute the following gluon-current:

$$J\left(\overline{q}_1, q_2, \mathrm{g}_3, \overline{q}_4, q_5, \mathrm{g}_6, \overline{q}_7, q_8\right) \tag{3.68}$$

which in turn requires the computation of

$$\vdots$$

$$J\left(\overline{q}_1, q_2, \mathrm{g}_3,\right) J\left(\overline{q}_4, q_5, \mathrm{g}_6, \overline{q}_7, q_8,\right) \tag{3.69}$$

$$J\left(\overline{q}_1, q_2, \mathrm{g}_3, \overline{q}_4\right) J\left(q_5, \mathrm{g}_6, \overline{q}_7, q_8\right) \tag{3.70}$$

$$J\left(\overline{q}_1, q_2, \mathrm{g}_3, \overline{q}_4, q_5\right) J\left(\mathrm{g}_6, \overline{q}_7, q_8\right) \tag{3.71}$$

$$\vdots$$

We will call the *split position* that index the particles are divided into two parts for subcurrents. In the examples above, the split positions are 4, 5, and 6. The cyclic order allows us to deduce the subcurrent type from

- the type of the particle at the split position, and
- the type of the current where the subcurrent is called.

In the examples above, we are computing a gluon current. Splitting at position 4 yields two gluon currents (fermion number is zero), splitting at position 5 an antiquark- and a quark-current (fermion number is -1 and 1 respectively), at position 6 we again have two gluon currents. Note this example also shows cyclic order implies that a gluon-current will never branch into a quark-current and an antiquark-current (but vice-versa).

We conclude that in gluon currents a quark at the split position yields antiquark-quark-currents, else gluon-currents.

Let us now look at the antiquark- $J(\overline{q}_1, q_2, \mathrm{g}_3, \overline{q}_4)$ and quark-current $J(q_5, \mathrm{g}_6, \overline{q}_7, q_8)$. The antiquark-current splits into

$$J(\overline{q}_1)\, J(q_2, \mathrm{g}_3, \overline{q}_4) \tag{3.72}$$

$$J(\overline{q}_1, q_2)\, J(\mathrm{g}_3, \overline{q}_4) \tag{3.73}$$

$$J(\overline{q}_1, q_2, \mathrm{g}_3)\, J(\overline{q}_4). \tag{3.74}$$

The first pair of currents are vetoed (i.e. they do not contribute to the process), because they have the wrong order (antiquark-gluon; a quark-current must follow an antiquark-current). The next pairs have the correct order and are gluon-currents and quark-currents. We can conclude, that in an antiquark-current the particle at the split position must not be a quark. The same holds true for the quark currents.

3.5.2 SU(N) Vetos

In Equation 2.41 we showed that the presence of more than one antiquark-quark-pairs gives rise to U(1)-gluons. They have a different color factor than U(N)-gluons and couple only between quarks/antiquarks. This in turn leads to the introduction of color clusters, which help us to identify "where" U(1) gluons are. In the computation of amplitudes we have to take this into account. In particular, we need to

- determine the cluster-membership of a current,
- check if a certain split must be *vetoed*, and
- identify whether a gluon current is of type U(N) or U(1)

These tests are explained in the following sections.

3.5.2.1 Cluster membership detection

To answer the question if a split is possible, we first have to explain the meaning of a "current belonging to a cluster" or the "cluster membership of a current". We already know this notion from the external particles. The cluster assignment (see subsection 3.3.2) tells us which particle belongs to which cluster. Now we have to extend this notion to currents. Since they contain external particles we are able to derive the cluster membership from the contained particles.

We begin with the on-shell-current J_{n-1}. This current is contracted with the first polarization vector/spinor to yield the partial amplitude. Since the first particle belongs always to cluster 1, the current also belongs to cluster 1.

Subcurrents also belong to the same cluster, as long as they are not connected by U(1)-gluon-currents. U(1)-currents are the only place where the cluster membership changes. Since U(1)-gluons do not carry colors, they do not belong to any cluster (this information is not needed either). The subcurrents of a U(1)-gluon-current belong to a cluster which is determined by the particle at the split position. As an example, consider the following split (numbers denote the cluster-assignment for the particles above):

$$\overbrace{\begin{array}{|c|}\hline \overline{q}_0 \\ \hline 1 \\ \hline \end{array}}^{u(0)} \overbrace{\underbrace{\begin{array}{|c|c|c|}\hline q_1 & \overline{q}_2 & q_3 \\ \hline 1 & 2 & 2 \\ \hline \end{array}}_{J_3 \text{ in cluster } 1} \underbrace{\begin{array}{|c|c|c|}\hline \overline{q}_4 & q_5 & \mathrm{g}_6 \\ \hline 1 & 1 & 1 \\ \hline \end{array}}_{J_3^v \text{ in cluster } 1}}^{J_6 \text{ in cluster } 1} \tag{3.75}$$

The amplitude is computed using an antiquark-current J_6 which is contracted with the polarization spinor $u(0)$ of the first particle. In the example the split happens at the antiquark with index 4, therefore the first current J_3 is antiquark-current, the second is a gluon-current. Both subcurrents belong to cluster 1, because no U(1)-gluon-current is involved. This happens when the first current is computed and split at index 2:

$$\overbrace{\underbrace{\begin{array}{|c|}\hline q_1 \\ \hline 1 \\ \hline \end{array}}_{J_1} \underbrace{\begin{array}{|c|c|}\hline \overline{q}_2 & q_3 \\ \hline 2 & 2 \\ \hline \end{array}}_{J_2^\rho \colon \mathrm{U}(1)}}^{J_3 \text{ in cluster } 1} \tag{3.76}$$

Because the pair $\overline{q}_2 q_3$ is assigned to a different cluster, the split yields the quark-current J_1 and U(1)-gluon-current J_2^ρ. In this U(1)-current, the subcurrents belong to a cluster different to 1. In the example above there is only one split possible (at index 2), thus the subcurrents belong to cluster 2.

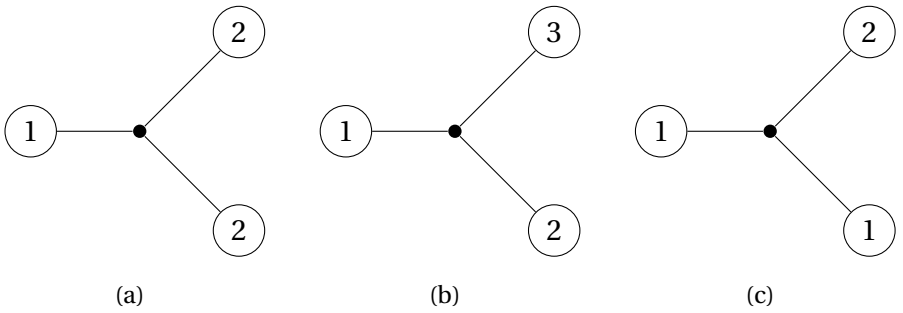


Figure 3.1: *Illustration of the splitting veto. The first case in 3.1(a) is vetoed, because the a current belonging to cluster "1" splits to two currents belonging to different clusters "2". 3.1(b) and 3.1(c) are not vetoed.*

3.5.2.2 Splitting Veto

The *splitting veto* may forbid the computation of subcurrents where splits are not possible. Suppose that a current belongs to cluster a and splits into two subcurrents belonging both to cluster $b \neq a$. This is not possible, since the currents must be cluster-connected unless they are connected by a colorless U(1)-current. This corresponds to the situation illustrated in Figure 3.1.

The splitting veto must be checked in gluon-currents splitting to two or three gluon currents (two splits, therefore two checks) and in quark and antiquark currents.

3.5.2.3 Particle-in-Cluster Veto

Gluon-currents can split into two or three gluon-subcurrents. These subcurrents must contain at least one particle belonging to same cluster. Gluon-currents splitting into an antiquark-quark-subcurrents must split at a position were the cluster assignment is equal to the cluster of the current.

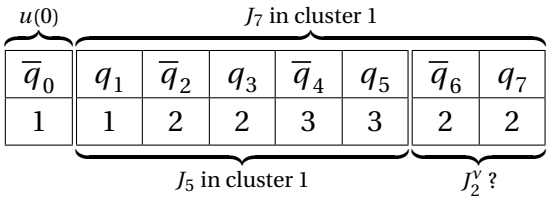
3.5.2.4 U(1)-Gluon Current Identification

In quark- and antiquark-currents both U(N)- and U(1)-gluon-subcurrents are possible, and from the particle types it is not clear which type is the correct one. This is again decided by looking at the cluster assignment of the gluon-current's particles — if the cluster of the (super-)current also appears in the subcurrents, both currents are connected by color, i.e. the current must be a U(N)-gluon-current. If this is not the case, the subcurrent is a U(1)-current. However, the split may be vetoed by the

"Cluster-complete veto".

3.5.2.5 Cluster-Complete Veto

Consider the following example:



(3.77)

In this split the current J_7 branches into two subcurrents, J_5 and J_2^v. With the U(1)-gluon current identification from the last section J_2^v is identified as a U(1)-gluon current, because it does not contain a particle beloning to cluster 1. This is wrong, because it would imply that this pair is disconnected from the pair $\overline{q}_2 q_3$ which is in the same cluster! Therefore, we have to veto U(1)-currents, which do not include every particle of the new cluster.

3.5.3 SU(N)×U(1) Modifications

The inclusion of QED-particles requires only little modification to the SU(N)-vetos. Since QED particles do not carry color, one can put these particles in a separate cluster called *color-less cluster* which will take the number 0. The vetoes must then be modified to simply ignore particles in cluster 0.

3.5.4 Virtual U(1)-photons and U(1)-gluons

If one wants to compute the matrix element to full orders of the QED-coupling constant, internal photon currents have to be taken into account. These couple like U(1)-gluon currents. One can make use of that property and treat the photon-current in the implementation equally. However, photon propagators do have a color factor of

$$-\frac{1}{N} \tag{3.78}$$

but couple with the strength of the (electric-)quark-charge which is $2/3$ (up-type quarks) or $-1/3$ (down-type quarks).

3.6 Program structure

In this section we describe some details of our implementation. Important classes are shown in Figure 3.2.

The interface for generating the particle configurations (see section 3.3) is found in class "`particle_conf`". Different implementations of this class are available, since different gauge-theories require different particle configurations. For example, an abelian gauge theory such as QED only needs particle permutations, which are generated by `simple_particle_conf`. SU(N)-gauge-theories such as QCD also need cluster configurations which are generated by `sun_particle_conf`. This class inherits from `simple_particle_conf` in order to avoid code duplication. The class `sun_simple_particle_conf` is meant for SU(N)×U(1) gauge-theory and uses both classes to generate the configurations for particles of both gauge-groups. Additionally, a method is provided that generates the shuffling permutations.

The abstract class `amplitude_helper` implements the vetos described in subsection 3.5.2. Classes deriving from this class implement the recursion relations which are used to compute on- and off-shell currents. These currents are used to compute the whole amplitude vector.

An instance of the class `squared_amplitude` computes the squared amplitudes. It uses the functionality of the `amplitude_helper` classes. If the gauge group is non-abelian, a color matrix is computed and the color summation is worked out with Equation 3.20 in class `default_squared_amplitude`.

3.7 Optimizations

Optimizations are a crucial part of matrix element computation. To illustrate why, let us assume that evaluation of a helicity matrix element takes about a second. If we want to compute physically meaningful objects, i.e. observables, we need to integrate the matrix elements over phase space. Let us assume this integration is carried out with a simple monte carlo integration which has a standard deviation of

$$\sigma = \frac{1}{\sqrt{N}}, \tag{3.79}$$

N the number of integration steps. If we want to achieve an accuracy of about one per cent, we need to evaluate ten thousand matrix elements. That means the program needs ten thousands seconds to finish; this is about three hours. If we want to

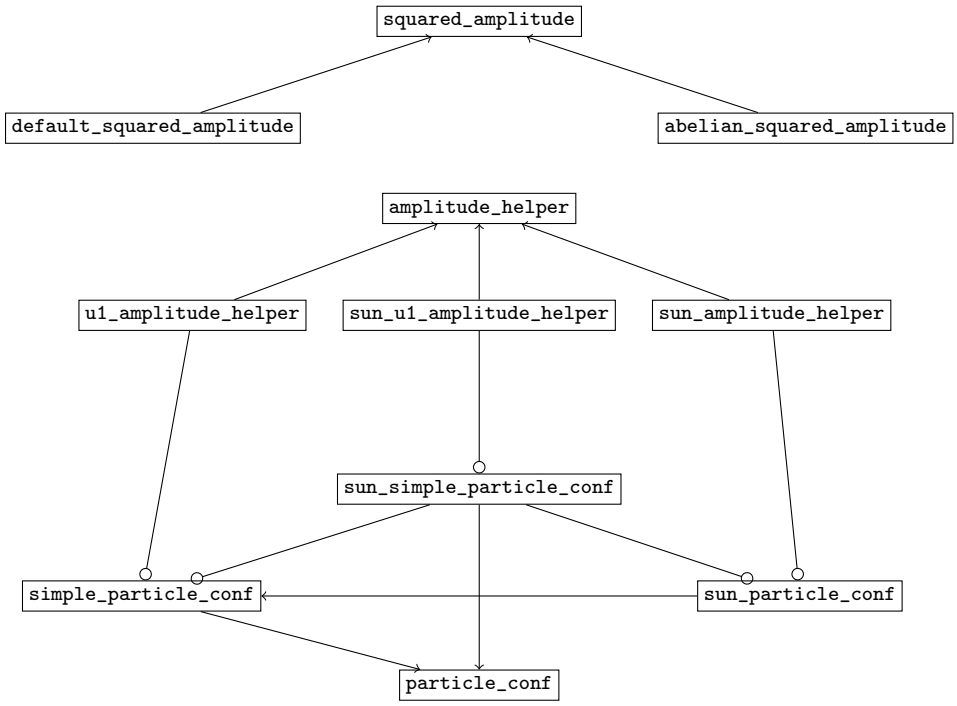


Figure 3.2: *Important classes of our implementation. "$x \longrightarrow y$" means x derives from y and "$x \multimap y$" means x uses an instance of y. The "`squared_amplitude`" classes compute squared matrix elements $\mathscr{M}$, "`amplitude_helper`" class compute the amplitude vectors A, and "`particle_conf`" class are used to iterate over the particle configurations*

compute an observable such as the differential cross section we have to compute a function and computing a single point of this function is clearly not sufficient. Ten points require the program to run a full day and now one can conclude that evaluation of a matrix element in a second is too slow.

We will now analyze the runtime complexity of the algorithm for evaluation of partial amplitudes for the worst case (gluon-processes). The runtime complexity for an implementation without optimizations, expressed with Big-Oh notation[37], is $O(2^n(n-1)!4^n)$:

- Summation over every possible helicity / spin direction accounts for the factor 2^n.[4] In the presence of an phase-space integral this can be optimized using monte-carlo techniques, see subsection 3.7.1.

- The maximum number of partial amplitudes is $(n-1)!$, which is the case when computing an all-gluon process. Processes with fermion/antifermion pairs have a lower number of partial amplitudes because fermions are not permuted with antifermions. In QCD, there are additional particle configurations because of the U(1)-gluon. To significantly lower the number of amplitudes one has to include color into the computed currents. This framework is then called "color-dressed"[38, 39] and the resulting algorithm has exponential complexity.

- The computation of a partial amplitude requires evaluation of a *tree* with two to three branches, possibly of a different (particle) type. The worst case is again the all-gluon process, where every current branches into at most three subcurrents. This results[23] in a complexity of $O(4^n)$. Saving intermediate results sinificantly lowers the number of computations performed, see subsection 3.7.2.

In the following sections we will present various optimization techniques ordered by importance.

3.7.1 Monte Carlo Techniques

As we pointed out in the beginning of section 3.7, a performance problem is the fact that one has to compute amplitudes with for every helicity configuration. For n par-

[4] An external vector boson has three polarization states, thus the helicity summation involves $2^{n_1}3^{n_2}$ steps, n_2 being the number of massive vector bosons.

ticles, one has to compute 2^n helicity amplitudes:

$$\sum_{\lambda_1=\pm}\sum_{\lambda_2=\pm}\cdots\sum_{\lambda_n=\pm}|\mathscr{M}(\lambda_1,\lambda_2,\ldots,\lambda_n)|^2. \tag{3.80}$$

One may make use of parity invariance of QED and QCD which relates matrix elements of reversed helicities:

$$\mathscr{M}(\lambda_1,\lambda_2,\ldots,\lambda_n) = -\left[\mathscr{M}(-\lambda_1,-\lambda_2,\ldots,-\lambda_n)\right]^* \tag{3.81}$$

However, this lowers the effort only by a factor of two. We can get rid of all of these summations if we replace our polarization vector with a vector depending not depending on the helicity h_k but on a polarization angle θ_k:

$$\varepsilon^{\mu_k}(p_k,q_k,h_k) \longrightarrow \varepsilon^{\mu_k}(p_k,q_k,\theta_k). \tag{3.82}$$

The new polarization vector is defined as a superposition of both possible polarizations:

$$\varepsilon^{\mu_k}(p_k,q_k,\theta_k) = \varepsilon^{\mu_k}(p_k,q_k,h=+)\,e^{2\pi i\theta_k} + \varepsilon^{\mu_k}(p_k,q_k,h=-)\,e^{-2\pi i\theta_k}. \tag{3.83}$$

Using this definition we need to integrate over the angle θ_k to compute the polarization summed amplitude. For an amplitude where we replaced a single polarization vector this reads

$$\int_0^1 \mathrm{d}\theta|\varepsilon^\mu(\theta)\mathscr{M}_\mu|^2 = \mathscr{M}_\mu\mathscr{M}_\nu^*\int_0^1\mathrm{d}\theta\varepsilon^\mu(\theta)\varepsilon^{*\nu}(\theta). \tag{3.84}$$

The integral evaluates to the helicity sum, because terms with mixed polarization states cancel:

$$\begin{aligned}\int_0^1\mathrm{d}\theta\varepsilon^\mu(\theta)\varepsilon^{*\nu}(\theta) &= \int_0^1\mathrm{d}\theta\left\{\varepsilon_+^\mu\varepsilon_+^{*\nu}+\varepsilon_-^\mu\varepsilon_-^{*\nu}+e^{4\pi i\theta}\varepsilon_+^\mu\varepsilon_-^{*\nu}+e^{4\pi i\theta}\varepsilon_-^\mu\varepsilon_+^{*\nu}\right\}\\ &= \sum_{\lambda=\pm}\varepsilon_\lambda^\mu\varepsilon_\lambda^{*\nu}.\end{aligned} \tag{3.85}$$

The same procedure is applied to dirac spinors:

$$u^s(p) \longrightarrow u(p,\theta) = u^{s=+\frac{1}{2}}(p)\,e^{2\pi i\theta} + u^{s=-\frac{1}{2}}(p)\,e^{-2\pi i\theta} \tag{3.86}$$

By applying this technique to every polarization vector and dirac spinor we replace the helicity/spin summation with a *helicity/spin integration*:

$$\sum_{\lambda_1=\pm}\cdots\sum_{\lambda_n=\pm} \longrightarrow \int_0^1\mathrm{d}\theta_1\cdots\int_0^1\mathrm{d}\theta_n \tag{3.87}$$

Up to now, this is not an optimization yet because the numerical evaluation of the multidimensional integral is usually slower than the summation over helicity configurations. However, one is not so interested in bare matrix elements but rather in observables $\mathcal{O}$ which require a phase-space integration over the squared matrix element:

$$\int_{\mathbb{R}^3} \frac{\mathrm{d}^3\vec{p}_3}{(2\pi)^3}\frac{1}{2E_{\vec{p}_3}} \cdots \int_{\mathbb{R}^3} \frac{\mathrm{d}^3\vec{p}_n}{(2\pi)^3}\frac{1}{2E_{\vec{p}_n}} \tag{3.88}$$

The integrals are evaluated with monte carlo integration[40]. The important feature of monte carlo integrals is that they converge *independent with the number of dimensions*. This fact can now be exploited by merging the phase space integral with the helicity angle integration. Thus, the helicity summation drops and we do not need to perform 2^n steps but only one.

3.7.2 Saving intermediate results

The calculation of amplitudes requires the computation of off- and on-shell-currents. These can be obtained recursively, as presented in (ref). This recursive approach has the advantage that it is easily understood and also easily implemented on computers. Despite this, recursive functions share a general disadvantage: they are slow. There are two main reasons for this behavior:

1. The obvious reason is that recursive functions do not remember intermediate results. For example, the process $0 \to \mathrm{gggg}$ forces the recursive function to compute the first polarization state $4 \cdot 3! = 24$ times (once would be sufficient). More particles increase the number of duplicate computations.

2. The second reason is a technical one; recursive functions require a growing and shrinking stack on which the variables of the functions are placed. In addition, a lot of functions are called. Both is to be considered expensive.

The first obstacle to overcome is to elimiate duplicate function calls, i.e. to save intermediate results. This is called *memoization*[41, 42] and can be done e.g. with a technique called Dynamic programming[37]. Before we go into details we will picture the general problem:

Given a set of recurring functions $f_1, \ldots, f_m$ with an arbitrary number of arguments $a_1, a_2, \ldots$ which alone determine the functions's value. How does one make

sure that every function in the recursion is called only once with a set of arguments?

$$f_1(a_1,\ldots,a_{n_1}) \tag{3.89}$$
$$f_2(a_1,\ldots,a_{n_2}) \tag{3.90}$$
$$\vdots$$
$$f_m(a_1,\ldots,a_{n_m}) \tag{3.91}$$

An important condition of the functions above is that the arguments alone determine the function's result. It must not depend on global variables.

3.7.2.1 Dynamic programming

Dynamic programming overcomes the problem by trading memory vs. time. Since we know that for a given function the set of arguments uniquely determines the function value, we map this set to a number h. We will call this number *hash value*:

$$f_x(a_1,\ldots,a_x) \longmapsto h. \tag{3.92}$$

The function which maps a functions arguments to a hash value will be called *hash function*. This function will allow us to modify our recurring functions:

- At the beginning of the function, compute a hash value of the arguments. Using the hash value, look up if the function was already called with the set of arguments. This can be done e.g. with a simple hashtable of boolean values.
  - If the functions was already called, return the saved function value.
  - If the function is called for the first time, compute the value.
- At the end of the function, save the result of the function and store it using the hash value.

To make that memoizing work, the hash function must be *perfect*.[5]

This method can be easily applied to optimize existing implementations. The most difficult part is to find a hash function for the function arguments. If general purpose hash functions providing $O(\log n)$ lookup-time do not suffice, hash tries may be used to create $O(1)$ lookup-time hash functions[43–45].

[5]It must not produce hash collsions. In mathematical terms the function must be injective

To gain a significant speedup, one has to make sure that the additional hash value computations and lookups are sufficiently fast compared to the actual computation. If this is not the case, it is possible that this method may yield even slower computations.

3.7.2.2 Iterative programming

Iterative programming attacks the problem of saving intermediate results in a different way. If we are able to find out

1. which sets of arguments are used to call the recursive functions

$$f_x(a_1, \ldots, a_x), \tag{3.93}$$

2. and we additionally know the functions call-graphs, i.e. which functions are called first and needed by other functions,

we can compute them iteratively in a bottom-to-top direction. For example, in our case we know that every current $J_1^\mu(p)$ with one external particle can be computed independently from each other. This holds true for currents with more particles $J_n(p_1, \ldots, p_n)$. Since they are constructed of currents with less particles, we will compute every current with one particle first, then currents with two particles (which needs the one-particle-currents), and so on.

However, we need to know the sets of arguments before we do our computations. That can be difficult — in Dynamic programming this is not a problem, because the recursion generates these sets "on the fly". We found a simple way to generate the arguments for gluon-amplitudes (see subsubsection 3.7.2.3); for amplitudes including quarks there is no simple way, because vetos and possible quark permutations are involved. However, it is always possible to run the recursion once and thereby to save the functions that are called and their arguments.

The iterative construction of currents will thus always be similar to the following scheme:

1. Set number of particles $k = 1$.

2. Compute every current with k particles and generate the hash value to save the result.

3. Increase the number of particles $k = k+1$ and repeat item 2 until every current is computed (if there are n particles, the last currents to be computed are those with $n-1$ particles; one can save some space by contracting the results directly with the first polarization vector, because the currents with $n-1$ particles are needed only once).

3.7.2.3 Iterative Construction of Gluon Amplitudes

The implementation of an iterative computation of gluon amplitudes raises the important question: How does one generate a hash value from the gluon current's arguments and uses this value to efficiently save the intermediate results? To answer this question, one has to know first which gluon currents are actually needed. As an example, take a four-gluon-process; the amplitude vector consists of the following entries:

$$A(0,1,2,3) = \varepsilon_\mu(0)\, J_3^\mu(1,2,3) \tag{3.94}$$
$$A(0,1,3,2) = \varepsilon_\mu(0)\, J_3^\mu(1,3,2) \tag{3.95}$$
$$A(0,2,1,3) = \varepsilon_\mu(0)\, J_3^\mu(2,1,3) \tag{3.96}$$
$$A(0,2,3,1) = \varepsilon_\mu(0)\, J_3^\mu(2,3,1) \tag{3.97}$$
$$A(0,3,1,2) = \varepsilon_\mu(0)\, J_3^\mu(3,1,2) \tag{3.98}$$
$$A(0,3,2,1) = \varepsilon_\mu(0)\, J_3^\mu(3,2,1)\,. \tag{3.99}$$

The non-cyclical permutations in the amplitude's arguments translate into permutations of gluon current's arguments (with one argument less but including cyclical permutations). The gluon currents J_3^μ with three particles in turn are computed using partitions of these arguments, e.g. for the first current

$$J_3^\mu(1,2,3) = V_{\text{gggg}}^{\mu\nu\rho\sigma} \underbrace{J_\nu(1)\, J_\rho(2)\, J_\sigma(3)}_{1+1+1=3} + V_{\text{ggg}}^{\mu\nu\rho} \underbrace{J_\nu(1)\, J_\rho(2,3)}_{1+2=3} + V_{\text{ggg}}^{\mu\nu\rho} \underbrace{J_\nu(1,2)\, J_\rho(3)}_{2+1=3}, \tag{3.100}$$

which is computed using a partition of the three arguments into three parts (four-gluon-current) and into two parts (three-gluon-vertex). Because order matters, there are two different partitions of two parts: $1+2=3$ and $2+1=3$. If we write down every

current needed for the four gluon process

$$J_1^\mu(1) \quad J_1^\mu(2) \quad J_1^\mu(3), \tag{3.101}$$

$$\begin{matrix} J_2^\mu(1,2) & J_2^\mu(1,3) & J_2^\mu(2,1) \\ J_2^\mu(2,3) & J_2^\mu(3,1) & J_2^\mu(3,2) \end{matrix} \tag{3.102}$$

$$\begin{matrix} J_3^\mu(1,2,3) & J_3^\mu(1,3,2) & J_3^\mu(2,1,3) \\ J_3^\mu(2,3,1) & J_3^\mu(3,1,2) & J_3^\mu(3,2,1) \end{matrix} \tag{3.103}$$

we see that the arguments are *partial permutations* of size $n = 1,2,3$. These can be easily constructed from permutations.[46] There is also a method to map the partial permutations of fixed size k to numbers, e.g. for $n = 3, k = 2$

$$\boxed{1}\boxed{2} \longmapsto 0 \tag{3.104}$$

$$\boxed{1}\boxed{3} \longmapsto 1 \tag{3.105}$$

$$\boxed{2}\boxed{1} \longmapsto 2 \tag{3.106}$$

$$\boxed{2}\boxed{3} \longmapsto 3 \tag{3.107}$$

$$\boxed{3}\boxed{1} \longmapsto 4 \tag{3.108}$$

$$\boxed{3}\boxed{2} \longmapsto 5. \tag{3.109}$$

This method maps the partial permutations in lexicographical order to ascending hash values h. That is done by reading the partial permutation $P = \{p_1, p_2, \ldots, p_k\}$ of size k with n digits $p_i \in \{1, 1, \ldots, k\}$ and following the algorithm:

1. Set h to 0 and i to 1.
2. Multiply h with $n - i$.
3. Add $p_i - 1$ to h.
4. Subtract 1 from h for every $p_j \in \{p_1, \ldots, p_{i-1}\}$ which is smaller than p_i.
5. Increment i and repeat item 2 for until i is $k + 1$.

This hash function maps partial permutation to integers. The function has some important properties, it is

- *minimal* (surjective), i.e the set of partial permutations (there are $\binom{n}{k} n!$ partial permutations) are "tightly" mapped to an interval $\left[0, \binom{n}{k} n!\right)$,

- *perfect* (injective), i.e. every partial permutation is mapped to a different integer and
- *order preserving*, i.e. the k-th partial permutation is mapped to k.

The first and second property allow to store the results for gluon-currents in an array-like structure, because the hash-value can be used as an index. The advantage of arrays over hash-table (which had to be used instead if these properties were not given) is the fast constant-time lookup. The third property allows to generate the currents and to save them in an array without looking up their hash value. Since we are generating the partial permutations beginning with the first one (assuming lexicographical order) the hash value is zero and will increase always by one for the next partial permutation.

3.7.3 Parallelization

Contemporary computers contain central processing units with multiple cores, or even multiple processors. Additionally, clusters of computer are often available in scientific communities providing sizable computational power. To make effective use of these resources, one has to prepare computer programs for parallelization.

We used Open MPI[47] for this task. Open MPI is an implementation of the Message Passing Interface[48] (MPI) which enables the user to spawn multiple instances of a program. The instances are run in parallel and may be spread over a network of computers. To achieve parallelization, the programmer has to implement the communication between these instances (message passing).

We have to chosen to parallelize the helcitiy integration, because the computations

1. are idependent from each other (it does not matter in which order the helicity amplitudes are computed and there no same subcurrents between parallel processes) and
2. they can be easily distributed among several parallel tasks:

When we are computing with helicity angles (i.e. a helicity integration, see subsection 3.7.1), we simply initialize the random number generator needed for the monte-carlo-integration with a different seed. Thus, in every integration step a different random number is chosen and the integration-steps can be carried out independent from each other.

Both properties are important to achieve the best scaling of the parallelization, i.e. one may expect half of the execution time if one doubles the number processes.[6]

An example for parallel monte carlo integration using Open MPI is found in section E.2. A speed comparison is found Appendix B.

[6] On different cores/cpus/computers, of course

Chapter 4

Conclusion

4.1 Results

In this thesis we presented and implemented a color-ordered algorithm described in Weinzierl [12] for the computation of leading order matrix elements in QCD. Berends-Giele-type recursion relations[15, 20] are used to compute partial amplitudes, because this method was reported[49] to perform fast enough even for a high number of external particles. The color-ordered algorithm was also used to compute QED amplitudes and QCD amplitudes with photons[29].

We implemented the algorithm in the form of a C++ program and checked our results against those of MadGraph 5[14, 50–52] (in Appendix D we describe how results were obtained). The comparison is found in section A.1. We have checked a minimal set of processes chosen such that an arbitrary process with at most eight particles is related by crossing symmetry. For example, the process $g\bar{d} \to \bar{d}g$ is equivalent to the process $d\bar{d} \to gg$. Since QCD does not distinguish between quark-flavors and we treat the light flavors (in accordance with MadGraph's default settings) massless, this process is equivalent to $u\bar{u} \to gg$ which is checked and listed in Table A.1. These tables also include the matrix elements obtained by using monte carlo polarizations (see subsection 3.7.1). Plots of some monte carlo convergences are found in Appendix B. We selected processes with show a "good" convergence and a processes with "worse" convergence.

In addition, we discussed the run-time-complexity of the algorithm and presented several techniques how to speed up the computation. Figure B.1 shows the performance difference between different implementations.

If the matrix element is integrated over phase space, which is the case in compu-

tations of observables, the use monte carlo polarizations is beneficial. section A.2 list the performance of both methods in comparison.

Finally, it is desirable to utilize every core available on a computer or even use the computational power of a cluster. For this reason, we employed Open MPI which makes it possible to run multiple instances of our program in parallel; Appendix B gives an overview of the speedup gained thereby.

4.2 Discussion

Our helicity summed matrix elements agree with those computed by Madgraph at least to 10^{-7} for every process tested. The matrix elements obtained by using monte carlo polarization converge against these — Table A.5 shows how many results differ by one, two, three or four standard variations. This table shows that the results very roughly follow a normal distribution. We thus can conclude, that the color-flow decomposed algorithm for computing matrix elements works for at least eight particles.

Figure B.1 shows that saving intermediate results for a matrix element with n particles reduces the computational time to about the time needed for a matrix element with $n-1$ particles. The difference between the Dynamic programming and the iterative approach is not very big (about 3–17 %) and lowers with many (eight) particles. We found this fact very interesting, since Dynamic programming still uses recursive functions for computing currents. To this end, we have to conclude that in our situation the use of recursive functions does not have a huge impact.

The use of multiple processors speeds up computation of about the number of (physical) processors, in our case four cores. Doubling the number of processes yields a small speedup because the processor comes with Hyperthreading support.

Appendix A

Tables

In this section we list the results of our program compared against those of MadGraph5. We use the set of momenta displayed in Table A.4. Every process has been calculated with fermion and antifermion masses set to zero.

"MC Dev." denotes the deviation of the monte carlo result from those obtained from MadGraph, "Fac." denotes the factor common to every result. The monte carlo error is not included in these tables, because they are always one per mill (for one million steps), one per cent (for ten thousand steps), and ten per cent (for hundred steps).

A.1 Results

Process	Result (GeV^{4-2n})	MC Result (GeV^{4-2n})	MG 5 (GeV^{4-2n})	Fac.	MC Dev. (%)
$gg \to gg$	5.51792506	5.51968689	5.51792506	10^{+01}	0.03
$u\bar{u} \to gg$	1.89410151	1.89334470	1.89410151	10^{+00}	0.04
$u\bar{u} \to \bar{d}d$	5.66450061	5.66658273	5.66450061	10^{-01}	0.04
$u\bar{u} \to \bar{u}u$	1.77514799	1.77493013	1.77514800	10^{+01}	0.01
$gg \to ggg$	2.31366804	2.31514434	2.31366804	10^{+00}	0.06
$u\bar{u} \to ggg$	5.13651260	5.13416708	5.13651260	10^{-02}	0.05
$u\bar{u} \to \bar{d}dg$	2.11900541	2.11719012	2.11900541	10^{-02}	0.09
$u\bar{u} \to \bar{u}ug$	1.05910976	1.05550073	1.05910976	10^{+00}	0.34
$gg \to gggg$	2.42797224	2.42301728	2.42797224	10^{+00}	0.20
$u\bar{u} \to gggg$	4.52856363	4.51413456	4.52856363	10^{-02}	0.32

Process	Result (GeV^{4-2n})	MC Result (GeV^{4-2n})	MG 5 (GeV^{4-2n})	Fac.	MC Dev. (%)
$u\bar{u} \to \bar{d}dgg$	5.65512384	5.65292638	5.65512396	10^{-04}	0.04
$u\bar{u} \to \bar{u}ugg$	1.62338088	1.62374626	1.62338088	10^{-01}	0.02
$u\bar{u} \to \bar{d}d\bar{c}c$	7.86493266	7.86229647	7.86493268	10^{-06}	0.03
$u\bar{u} \to \bar{d}d\bar{d}d$	2.44424558	2.44777436	2.44424561	10^{-05}	0.14
$u\bar{u} \to \bar{u}u\bar{u}u$	1.13460730	1.13570874	1.13460730	10^{-03}	0.10
$gg \to ggggg$	1.25582835	1.24823850	1.25582835	10^{+00}	0.60
$u\bar{u} \to ggggg$	1.54699963	1.52409404	1.54699963	10^{-02}	1.48
$u\bar{u} \to \bar{d}dggg$	1.95774177	1.95397187	1.95774181	10^{-04}	0.19
$u\bar{u} \to \bar{u}uggg$	5.29886772	5.20990104	5.29886772	10^{-02}	1.68
$u\bar{u} \to \bar{d}d\bar{c}cg$	6.98128792	7.09317879	6.98128799	10^{-06}	1.60
$u\bar{u} \to \bar{d}d\bar{d}dg$	3.47979190	3.40960456	3.47979193	10^{-05}	2.02
$u\bar{u} \to \bar{u}u\bar{u}ug$	1.57346579	1.57390226	1.57346580	10^{-03}	0.03
$u\bar{u} \to gggggg$	4.63844365	5.02442446	4.63844365	10^{-03}	8.32
$u\bar{u} \to \bar{d}dgggg$	8.42577960	8.66134245	8.42577974	10^{-05}	2.80
$u\bar{u} \to \bar{u}ugggg$	1.91870428	2.32419125	1.91870428	10^{-02}	21.13
$u\bar{u} \to \bar{d}d\bar{c}cgg$	5.43271062	5.63250050	5.43271065	10^{-06}	3.68
$u\bar{u} \to \bar{d}d\bar{d}ddgg$	3.48592176	3.98648587	3.48592179	10^{-05}	14.36
$u\bar{u} \to \bar{u}u\bar{u}ugg$	9.16567149	7.93465324	9.16567151	10^{-04}	13.43
$u\bar{u} \to \bar{d}d\bar{c}c\bar{s}s$	1.73463672	1.86695949	1.73463673	10^{-07}	7.63
$u\bar{u} \to \bar{d}d\bar{d}d\bar{c}c$	1.06902584	1.26562131	1.06902585	10^{-06}	18.39
$u\bar{u} \to \bar{d}d\bar{d}d\bar{d}d$	1.23326471	1.45091382	1.23326472	10^{-07}	17.65
$u\bar{u} \to \bar{u}u\bar{d}d\bar{d}d$	1.32905684	1.28745473	1.32905686	10^{-06}	3.13
$u\bar{u} \to \bar{u}u\bar{u}u\bar{u}u$	4.24345831	4.16652423	4.24345832	10^{-06}	1.81

Table A.1: *Comparison of our program with MadGraph 5 for QCD processes with up to eight external particles. The Monte Carlo results are produced using a naïve summation and one million steps for processes with up to six particles, ten thousand steps for processes with seven particles and hundred steps for processes with eight particles. Note that the gluon-process with eight gluons is not included since MadGraph was not able to produce results (compilation needs more than 8 GigaBytes RAM).*

Process	Result (GeV^{4-2n})	MC Result (GeV^{4-2n})	MG 5 (GeV^{4-2n})	Fac.	MC Dev. (%)
$e^-e^+ \to \gamma\gamma$	2.66226014	2.66119639	2.66226014	10^{-02}	0.04
$e^-e^+ \to \mu^+\mu^-$	1.11910037	1.11951172	1.11910037	10^{-02}	0.04
$e^-e^+ \to e^+e^-$	2.66845846	2.66776058	2.66845846	10^{-01}	0.03
$e^-e^+ \to \gamma\gamma\gamma$	1.91085162	1.90677912	1.91085162	10^{-05}	0.21
$e^-e^+ \to \mu^+\mu^-\gamma$	3.44944788	3.44683799	3.44944788	10^{-05}	0.08
$e^-e^+ \to e^+e^-\gamma$	7.23144982	7.22504706	7.23144982	10^{-04}	0.09
$e^-e^+ \to \gamma\gamma\gamma\gamma$	1.23190750	1.23216122	1.23190750	10^{-07}	0.02
$e^-e^+ \to \mu^+\mu^-\gamma\gamma$	1.67853674	1.67835798	1.67853674	10^{-08}	0.01
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-$	2.08620003	2.08489079	2.08620003	10^{-09}	0.06
$e^-e^+ \to e^+e^-\mu^+\mu^-$	1.73251823	1.72941084	1.73251823	10^{-08}	0.18
$e^-e^+ \to e^+e^-e^+e^-$	5.95305848	5.94747741	5.95305848	10^{-08}	0.09
$e^-e^+ \to \gamma\gamma\gamma\gamma$	8.48152759	8.48279833	8.48152759	10^{-10}	0.01
$e^-e^+ \to \mu^+\mu^-\gamma\gamma$	2.31395460	2.31033329	2.31395460	10^{-10}	0.16
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-\gamma$	1.64367483	1.64241144	1.64367483	10^{-11}	0.08
$e^-e^+ \to e^+e^-\mu^+\mu^-\gamma$	2.58659938	2.58512468	2.58659938	10^{-10}	0.06
$e^-e^+ \to e^+e^-e^+e^-\gamma$	1.10049307	1.10090374	1.10049307	10^{-09}	0.04
$e^-e^+ \to \gamma\gamma\gamma\gamma\gamma$	6.99895206	6.98537119	6.99895206	10^{-12}	0.19
$e^-e^+ \to \mu^+\mu^-\gamma\gamma\gamma$	3.68773951	3.68408083	3.68773951	10^{-12}	0.10
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-\gamma\gamma$	3.63508310	3.63224032	3.63508306	10^{-13}	0.08
$e^-e^+ \to e^+e^-\mu^+\mu^-\gamma\gamma$	4.22608582	4.23656958	4.22608582	10^{-12}	0.25
$e^-e^+ \to e^+e^-e^+e^-\gamma\gamma$	1.86644119	1.86451073	1.86644119	10^{-11}	0.10
$e^-e^+ \to e^+e^-\mu^+\mu^-\tau^+\tau^-$	1.54093032	1.54281303	1.54093032	10^{-12}	0.12
$e^-e^+ \to e^+e^-\mu^+\mu^-\mu^+\mu^-$	3.54215370	3.54509182	3.54215373	10^{-13}	0.08
$e^-e^+ \to e^+e^-e^+e^-\mu^+\mu^-$	5.89007979	5.87168784	5.89007979	10^{-12}	0.31
$e^-e^+ \to e^+e^-e^+e^-e^+e^-$	7.12116530	7.10575090	7.12116527	10^{-13}	0.22

Table A.2: *Comparison of our program with MadGraph 5 for QED processes with up to eight external particles. The Monte Carlo results are produced using a naïve summation and one million steps.*

Process	Result (GeV^{4-2n})	MC Result (GeV^{4-2n})	MG 5 (GeV^{4-2n})	Fac.	MC Dev. (%)
$u\bar{u} \to g\gamma$	1.58733284	1.58669859	1.58733284	10^{-01}	0.04
$u\bar{u} \to \gamma\gamma$	1.75292849	1.75222808	1.75292849	10^{-03}	0.04
$u\bar{u} \to gg\gamma$	1.81882104	1.81512716	1.81882104	10^{-03}	0.20
$u\bar{u} \to g\gamma\gamma$	7.59544489	7.57925712	7.59544489	10^{-05}	0.21
$u\bar{u} \to \gamma\gamma\gamma$	5.59188403	5.57996634	5.59188403	10^{-07}	0.21
$u\bar{u} \to \bar{d}d\gamma$	6.90793109	6.89203814	6.90793109	10^{-04}	0.23
$u\bar{u} \to \bar{u}u\gamma$	2.13546781	2.13214271	2.13546782	10^{-02}	0.16
$u\bar{u} \to ggg\gamma$	4.19402730	4.17828624	4.19402730	10^{-03}	0.38
$u\bar{u} \to gg\gamma\gamma$	1.51387483	1.51455303	1.51387483	10^{-05}	0.04
$u\bar{u} \to g\gamma\gamma\gamma$	2.90175385	2.90235149	2.90175385	10^{-07}	0.02
$u\bar{u} \to \gamma\gamma\gamma\gamma$	1.60223705	1.60256705	1.60223705	10^{-09}	0.02
$u\bar{u} \to \bar{d}dg\gamma$	9.59062023	9.58452305	9.59062023	10^{-05}	0.06
$u\bar{u} \to \bar{d}d\gamma\gamma$	4.85357270	4.85171389	4.85357270	10^{-07}	0.04
$u\bar{u} \to \bar{u}ug\gamma$	1.21767421	1.21700535	1.21767421	10^{-03}	0.05
$u\bar{u} \to \bar{u}u\gamma\gamma$	1.34506130	1.34540593	1.34506130	10^{-05}	0.03
$u\bar{u} \to gggg\gamma$	6.65240363	6.63869790	6.65240363	10^{-04}	0.21
$u\bar{u} \to ggg\gamma\gamma$	2.95176114	2.93609074	2.95176114	10^{-05}	0.53
$u\bar{u} \to gg\gamma\gamma\gamma$	7.55969500	7.36840497	7.55969500	10^{-08}	2.53
$u\bar{u} \to g\gamma\gamma\gamma\gamma$	1.10990049	1.07959702	1.10990049	10^{-09}	2.73
$u\bar{u} \to \gamma\gamma\gamma\gamma\gamma$	4.90275548	4.76889616	4.90275548	10^{-12}	2.73
$u\bar{u} \to \bar{d}ddgg\gamma$	6.33355988	6.31766121	6.33356000	10^{-06}	0.25
$u\bar{u} \to \bar{d}ddg\gamma\gamma$	5.22000444	5.12662826	5.22000444	10^{-07}	1.79
$u\bar{u} \to \bar{d}dg\gamma\gamma\gamma$	1.86963444	1.85805992	1.86963444	10^{-09}	0.62
$u\bar{u} \to \bar{u}uggg\gamma$	4.34434146	4.23846924	4.34434146	10^{-03}	2.44
$u\bar{u} \to \bar{u}ug\gamma\gamma$	1.36046477	1.36471194	1.36046477	10^{-05}	0.31
$u\bar{u} \to \bar{u}u\gamma\gamma\gamma$	9.74915472	9.65800030	9.74915473	10^{-08}	0.93
$u\bar{u} \to \bar{d}d\bar{c}c\gamma$	2.92378473	2.94094535	2.92378473	10^{-08}	0.59
$u\bar{u} \to \bar{d}dd\bar{d}d\gamma$	4.75651770	4.70796164	4.75651775	10^{-07}	1.02
$u\bar{u} \to \bar{u}u\bar{u}u\gamma$	7.59180937	7.60140520	7.59180937	10^{-06}	0.13
$u\bar{u} \to ggggg\gamma$	2.39322452	2.37603711	2.39322452	10^{-04}	0.72
$u\bar{u} \to gggg\gamma\gamma$	6.24448016	6.26473107	6.24448016	10^{-06}	0.32
$u\bar{u} \to ggg\gamma\gamma\gamma$	1.43170216	1.45841878	1.43170216	10^{-07}	1.87
$u\bar{u} \to gg\gamma\gamma\gamma\gamma$	3.89029069	3.97356282	3.89029069	10^{-10}	2.14

Process	Result (GeV^{4-2n})	MC Result (GeV^{4-2n})	MG 5 (GeV^{4-2n})	Fac.	MC Dev. (%)
$u\bar{u} \to g\gamma\gamma\gamma\gamma$	4.88474292	4.98234435	4.88474292	10^{-12}	2.00
$u\bar{u} \to \gamma\gamma\gamma\gamma\gamma$	1.79811200	1.83403949	1.79811165	10^{-14}	2.00
$u\bar{u} \to \bar{d}dggg\gamma$	2.59181660	2.63275737	2.59181664	10^{-06}	1.58
$u\bar{u} \to \bar{d}dgg\gamma\gamma$	5.26125094	5.32694935	5.26125103	10^{-08}	1.25
$u\bar{u} \to \bar{d}dg\gamma\gamma\gamma$	2.76339379	2.80415076	2.76339379	10^{-09}	1.47
$u\bar{u} \to \bar{d}d\gamma\gamma\gamma\gamma$	8.70374316	8.82088330	8.70374316	10^{-12}	1.35
$u\bar{u} \to \bar{u}uggg\gamma$	1.47585283	1.49627627	1.47585283	10^{-03}	1.38
$u\bar{u} \to \bar{u}ugg\gamma\gamma$	5.23213887	5.30424288	5.23213887	10^{-05}	1.38
$u\bar{u} \to \bar{u}ug\gamma\gamma\gamma$	9.65381828	9.66483176	9.65381828	10^{-08}	0.11
$u\bar{u} \to \bar{u}u\gamma\gamma\gamma\gamma$	4.97559712	5.02030291	4.97559713	10^{-10}	0.90
$u\bar{u} \to \bar{d}d\bar{c}cg\gamma$	5.33409745	5.33606463	5.33409751	10^{-08}	0.04
$u\bar{u} \to \bar{d}d\bar{c}c\gamma\gamma$	1.60472175	1.55894925	1.60472175	10^{-10}	2.85
$u\bar{u} \to \bar{d}d\bar{d}dg\gamma$	7.24587841	7.17274399	7.24587848	10^{-07}	1.01
$u\bar{u} \to \bar{d}d\bar{d}d\gamma\gamma$	5.88656235	5.79592239	5.88656241	10^{-09}	1.54
$u\bar{u} \to \bar{u}u\bar{u}ug\gamma$	1.65186057	1.64426577	1.65186058	10^{-05}	0.46
$u\bar{u} \to \bar{u}u\bar{u}u\gamma\gamma$	5.40503769	5.35173302	5.40503769	10^{-08}	0.99

Table A.3: *Comparison of our program with MadGraph 5 for QCD with additional photons with up to eight external particles. The Monte Carlo results are produced using a naïve summation and one million steps for processes with up to six particles, ten thousand steps for processes with seven and eight particles.*

A.2 Timings

The following tables list the performance of our program. These timings were obtained on a laptop with a Intel Core i7-720QM processor (1.73 GHz) with Turbo Boost disabled[1]

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)
$gg \to gg$	0.15	0.55	0.05

[1] Turbo Boost overclocks the processor in single core modus; since this is not controllable, we disabled the feature.

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)
$u\bar{u} \to gg$	0.16	0.41	0.04
$u\bar{u} \to \bar{d}d$	0.14	0.36	0.04
$u\bar{u} \to \bar{u}u$	0.16	0.43	0.06
$gg \to ggg$	0.51	3.85	0.13
$u\bar{u} \to ggg$	0.54	2.86	0.11
$u\bar{u} \to \bar{d}dg$	0.24	1.33	0.05
$u\bar{u} \to \bar{u}ug$	0.81	2.32	0.20
$gg \to gggg$	9.03	36.02	0.57
$u\bar{u} \to gggg$	11.17	40.88	1.08
$u\bar{u} \to \bar{d}dgg$	3.20	11.75	0.27
$u\bar{u} \to \bar{u}ugg$	11.10	20.27	0.50
$u\bar{u} \to \bar{d}d\bar{c}c$	0.38	3.81	0.09
$u\bar{u} \to \bar{d}d\bar{d}d$	0.80	6.23	0.14
$u\bar{u} \to \bar{u}u\bar{u}u$	4.78	15.77	0.37
$gg \to ggggg$	298.58	847.61	6.40
$u\bar{u} \to ggggg$	466.46	1500.04	19.30
$u\bar{u} \to \bar{d}dggg$	153.41	252.45	3.31
$u\bar{u} \to \bar{u}uggg$	749.64	553.09	6.73
$u\bar{u} \to \bar{d}d\bar{c}cg$	8.34	53.12	0.70
$u\bar{u} \to \bar{d}d\bar{d}dg$	26.59	103.27	1.36
$u\bar{u} \to \bar{u}u\bar{u}ug$	214.66	351.80	5.01
$u\bar{u} \to gggggg$	40495.97	65391.93	453.60
$u\bar{u} \to \bar{d}dgggg$	15798.97	10011.52	60.99
$u\bar{u} \to \bar{u}ugggg$	61942.32	24301.38	138.65
$u\bar{u} \to \bar{d}d\bar{c}cgg$	696.95	1554.09	14.45
$u\bar{u} \to \bar{d}d\bar{d}dgg$	2729.44	3401.95	20.59
$u\bar{u} \to \bar{u}u\bar{u}ugg$	24264.26	17095.57	90.00
$u\bar{u} \to \bar{d}d\bar{c}c\bar{s}s$	11.12	237.75	1.67
$u\bar{u} \to \bar{d}d\bar{d}d\bar{c}c$	36.60	491.41	3.34
$u\bar{u} \to \bar{d}d\bar{d}d\bar{d}d$	299.38	1834.69	10.35
$u\bar{u} \to \bar{u}u\bar{d}d\bar{d}d$	144.55	1026.96	6.73
$u\bar{u} \to \bar{u}u\bar{u}u\bar{u}u$	4977.39	12493.35	64.45

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)

Table A.6: *Performance comparison for the computation of color matrices, color and spin-summed squared amplitudes, and a single color summed squared amplitude with monte carlo polarizations.*

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)
$e^-e^+ \to \gamma\gamma$	0.06	0.32	0.03
$e^-e^+ \to \mu^+\mu^-$	0.13	0.26	0.03
$e^-e^+ \to e^+e^-$	0.06	0.27	0.03
$e^-e^+ \to \gamma\gamma\gamma$	0.07	1.17	0.06
$e^-e^+ \to \mu^+\mu^-\gamma$	0.06	0.67	0.04
$e^-e^+ \to e^+e^-\gamma$	0.06	0.84	0.05
$e^-e^+ \to \gamma\gamma\gamma\gamma$	0.07	12.88	0.36
$e^-e^+ \to \mu^+\mu^-\gamma\gamma$	0.07	3.82	0.11
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-$	0.07	1.60	0.05
$e^-e^+ \to e^+e^-\mu^+\mu^-$	0.07	2.00	0.05
$e^-e^+ \to e^+e^-e^+e^-$	0.07	4.14	0.14
$e^-e^+ \to \gamma\gamma\gamma\gamma\gamma$	0.07	175.96	2.40
$e^-e^+ \to \mu^+\mu^-\gamma\gamma\gamma$	0.05	34.73	0.47
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-\gamma$	0.06	9.66	0.11
$e^-e^+ \to e^+e^-\mu^+\mu^-\gamma$	0.05	16.38	0.35
$e^-e^+ \to e^+e^-e^+e^-\gamma$	0.05	39.40	0.56
$e^-e^+ \to \gamma\gamma\gamma\gamma\gamma\gamma$	0.05	3755.12	25.37
$e^-e^+ \to \mu^+\mu^-\gamma\gamma\gamma\gamma$	0.06	680.24	4.84
$e^-e^+ \to \mu^+\mu^-\tau^+\tau^-\gamma\gamma$	0.07	167.94	0.93
$e^-e^+ \to e^+e^-\mu^+\mu^-\gamma\gamma$	0.05	249.57	1.80
$e^-e^+ \to e^+e^-e^+e^-\gamma\gamma$	0.05	745.73	5.25
$e^-e^+ \to e^+e^-\mu^+\mu^-\tau^+\tau^-$	0.07	51.72	0.34
$e^-e^+ \to e^+e^-\mu^+\mu^-\mu^+\mu^-$	0.06	92.36	0.62
$e^-e^+ \to e^+e^-e^+e^-\mu^+\mu^-$	0.06	120.71	0.91
$e^-e^+ \to e^+e^-e^+e^-e^+e^-$	0.05	436.98	3.44

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)

Table A.7: *Performance comparison for the computation of spin-summed squared amplitudes, and a single squared amplitude with monte carlo polarizations. Note that the entries for the color matrices are meaningless since QED does not have a color matrix.*

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)
$u\bar{u} \to g\gamma$	0.13	0.73	0.06
$u\bar{u} \to \gamma\gamma$	0.11	1.17	0.10
$u\bar{u} \to gg\gamma$	0.29	3.74	0.15
$u\bar{u} \to g\gamma\gamma$	0.14	2.84	0.13
$u\bar{u} \to \gamma\gamma\gamma$	0.10	2.54	0.14
$u\bar{u} \to \bar{d}d\gamma$	0.15	2.81	0.13
$u\bar{u} \to \bar{u}u\gamma$	0.22	3.23	0.15
$u\bar{u} \to ggg\gamma$	0.65	30.60	0.67
$u\bar{u} \to gg\gamma\gamma$	0.17	18.34	0.35
$u\bar{u} \to g\gamma\gamma\gamma$	0.14	16.39	0.29
$u\bar{u} \to \gamma\gamma\gamma\gamma$	0.13	17.11	0.31
$u\bar{u} \to \bar{d}dg\gamma$	0.27	11.32	0.34
$u\bar{u} \to \bar{d}d\gamma\gamma$	0.16	11.02	0.24
$u\bar{u} \to \bar{u}ug\gamma$	0.73	21.00	0.43
$u\bar{u} \to \bar{u}u\gamma\gamma$	0.25	16.37	0.30
$u\bar{u} \to gggg\gamma$	12.47	801.79	9.39
$u\bar{u} \to ggg\gamma\gamma$	0.72	456.24	8.58
$u\bar{u} \to gg\gamma\gamma\gamma$	0.14	262.77	3.15
$u\bar{u} \to g\gamma\gamma\gamma\gamma$	0.15	219.12	2.60
$u\bar{u} \to \gamma\gamma\gamma\gamma\gamma$	0.18	244.73	2.80
$u\bar{u} \to \bar{d}dgg\gamma$	4.10	205.01	2.02
$u\bar{u} \to \bar{d}dg\gamma\gamma$	0.22	115.69	1.41
$u\bar{u} \to \bar{d}d\gamma\gamma\gamma$	0.15	107.20	1.18
$u\bar{u} \to \bar{u}ugg\gamma$	13.05	315.58	3.98
$u\bar{u} \to \bar{u}ug\gamma\gamma$	0.63	234.23	2.77

Process	Color matrix (ms)	Helicity amplitudes (ms)	MC amplitudes (ms)
$u\bar{u} \to \bar{u}u\gamma\gamma\gamma$	0.20	243.13	2.31
$u\bar{u} \to \bar{d}d\bar{c}c\gamma$	0.34	44.40	0.55
$u\bar{u} \to \bar{d}d\bar{d}d\gamma$	0.61	81.67	1.03
$u\bar{u} \to \bar{u}u\bar{u}u\gamma$	4.37	232.53	3.00
$u\bar{u} \to ggggg\gamma$	504.51	32965.75	202.80
$u\bar{u} \to gggg\gamma\gamma$	11.90	17159.05	99.36
$u\bar{u} \to ggg\gamma\gamma\gamma$	0.48	9536.68	54.13
$u\bar{u} \to gg\gamma\gamma\gamma\gamma$	0.22	6345.66	36.45
$u\bar{u} \to g\gamma\gamma\gamma\gamma\gamma$	0.17	5478.15	35.61
$u\bar{u} \to \gamma\gamma\gamma\gamma\gamma\gamma$	2.89	6369.00	34.79
$u\bar{u} \to \bar{d}dggg\gamma$	156.75	5530.72	32.40
$u\bar{u} \to \bar{d}dgg\gamma\gamma$	3.15	3305.34	19.99
$u\bar{u} \to \bar{d}dg\gamma\gamma\gamma$	0.27	2407.47	20.94
$u\bar{u} \to \bar{d}d\gamma\gamma\gamma\gamma$	0.18	2095.83	12.30
$u\bar{u} \to \bar{u}uggg\gamma$	637.45	10563.90	65.19
$u\bar{u} \to \bar{u}ugg\gamma\gamma$	12.79	6534.48	58.54
$u\bar{u} \to \bar{u}ug\gamma\gamma\gamma$	0.75	5052.26	33.72
$u\bar{u} \to \bar{u}u\gamma\gamma\gamma\gamma$	0.20	4113.82	24.44
$u\bar{u} \to \bar{d}d\bar{c}cg\gamma$	8.55	1076.41	6.66
$u\bar{u} \to \bar{d}d\bar{c}c\gamma\gamma$	0.31	790.43	4.84
$u\bar{u} \to \bar{d}d\bar{d}dg\gamma$	30.05	2122.89	13.13
$u\bar{u} \to \bar{d}d\bar{d}d\gamma\gamma$	0.86	1500.02	9.54
$u\bar{u} \to \bar{u}u\bar{u}ug\gamma$	303.03	6560.71	39.46
$u\bar{u} \to \bar{u}u\bar{u}u\gamma\gamma$	6.01	4614.99	28.71

Table A.8: *Performance comparison for the computation of color matrices, color and spin-summed squared amplitudes, and a single color summed squared amplitude with monte carlo polarizations.*

		p^0 (GeV)	p^1 (GeV)	p^2 (GeV)	p^3 (GeV)
4	p_1	$0.450000000000 \times 10^{+02}$	$0.000000000000 \times 10^{+00}$	$0.000000000000 \times 10^{+00}$	$0.450000000000 \times 10^{+02}$
	p_2	$0.450000000000 \times 10^{+02}$	$0.000000000000 \times 10^{+00}$	$0.000000000000 \times 10^{+00}$	$-0.450000000000 \times 10^{+02}$
	p_3	$0.450000000000 \times 10^{+02}$	$0.998318559994 \times 10^{+01}$	$0.400347710539 \times 10^{+02}$	$-0.179597636938 \times 10^{+02}$
	p_4	$0.450000000000 \times 10^{+02}$	$-0.998318559994 \times 10^{+01}$	$-0.400347710539 \times 10^{+02}$	$0.179597636938 \times 10^{+02}$
5	p_3	$0.412720909097 \times 10^{+02}$	$0.152507898279 \times 10^{+02}$	$0.341688295870 \times 10^{+02}$	$-0.174152227185 \times 10^{+02}$
	p_4	$0.327659958663 \times 10^{+02}$	$-0.164968823639 \times 10^{+01}$	$-0.312933871187 \times 10^{+02}$	$0.957146469828 \times 10^{+01}$
	p_5	$0.159619132240 \times 10^{+02}$	$-0.136011015915 \times 10^{+02}$	$-0.287544246829 \times 10^{+01}$	$0.784375802024 \times 10^{+01}$
6	p_3	$0.796961997491 \times 10^{+01}$	$-0.198906212589 \times 10^{+01}$	$0.360723178725 \times 10^{+01}$	$-0.682248878612 \times 10^{+01}$
	p_4	$0.295496477304 \times 10^{+02}$	$-0.934646506951 \times 10^{+01}$	$-0.271740379851 \times 10^{+02}$	$0.688454292484 \times 10^{+01}$
	p_5	$0.137122298521 \times 10^{+02}$	$-0.952928636999 \times 10^{+01}$	$-0.879386744943 \times 10^{+01}$	$0.445935467041 \times 10^{+01}$
	p_6	$0.387685024426 \times 10^{+02}$	$0.208648135654 \times 10^{+02}$	$0.323606736472 \times 10^{+02}$	$-0.452140880913 \times 10^{+01}$
7	p_3	$0.777186613294 \times 10^{+01}$	$-0.136920503882 \times 10^{+01}$	$0.338701961654 \times 10^{+01}$	$-0.685968504140 \times 10^{+01}$
	p_4	$0.252106363628 \times 10^{+02}$	$-0.748135004855 \times 10^{+01}$	$-0.236883471083 \times 10^{+02}$	$0.429741766044 \times 10^{+01}$
	p_5	$0.114770276613 \times 10^{+02}$	$-0.814041371664 \times 10^{+01}$	$-0.748604769328 \times 10^{+01}$	$0.306837390053 \times 10^{+01}$
	p_6	$0.372717061537 \times 10^{+02}$	$0.208931008451 \times 10^{+02}$	$0.299478993103 \times 10^{+02}$	$-0.746871766672 \times 10^{+01}$
	p_7	$0.826876368924 \times 10^{+01}$	$-0.390213204112 \times 10^{+01}$	$-0.216052412526 \times 10^{+01}$	$0.696261114714 \times 10^{+01}$
8	p_3	$0.769781023546 \times 10^{+01}$	$-0.739973901580 \times 10^{+00}$	$0.325474053914 \times 10^{+01}$	$-0.693652543202 \times 10^{+01}$
	p_4	$0.163285930449 \times 10^{+02}$	$-0.520739846534 \times 10^{+01}$	$-0.154677360678 \times 10^{+02}$	$-0.505067308633 \times 10^{+00}$
	p_5	$0.745643709697 \times 10^{+01}$	$-0.593185928612 \times 10^{+01}$	$-0.449056941477 \times 10^{+01}$	$0.496272024053 \times 10^{+00}$
	p_6	$0.343323346771 \times 10^{+02}$	$0.171166749337 \times 10^{+02}$	$0.262838646886 \times 10^{+02}$	$-0.139601970123 \times 10^{+02}$
	p_7	$0.488082663106 \times 10^{+01}$	$-0.280197145874 \times 10^{+01}$	$-0.713516991112 \times 10^{+00}$	$0.393221541250 \times 10^{+01}$
	p_8	$0.193039983145 \times 10^{+02}$	$-0.243547182196 \times 10^{+01}$	$-0.886678275408 \times 10^{+01}$	$0.169733023164 \times 10^{+02}$

Table A.4: *Momenta used for computation of matrix elements. The center-of-mass energy is* 90 GeV*, particles are massless.*

1σ	2σ	3σ	4σ
58	28	19	4
53 %	78 %	96 %	100 %

Table A.5: *Deviations of the monte carlo results.* $\sigma = 1/N$ *denotes the standard deviation of the monte carlo integration. The column* 2σ *shows that 28 results differ more than* 1σ *and less than* 2σ *and that 78 % of the results differ at* 2σ *at most.*

Appendix B

Figures

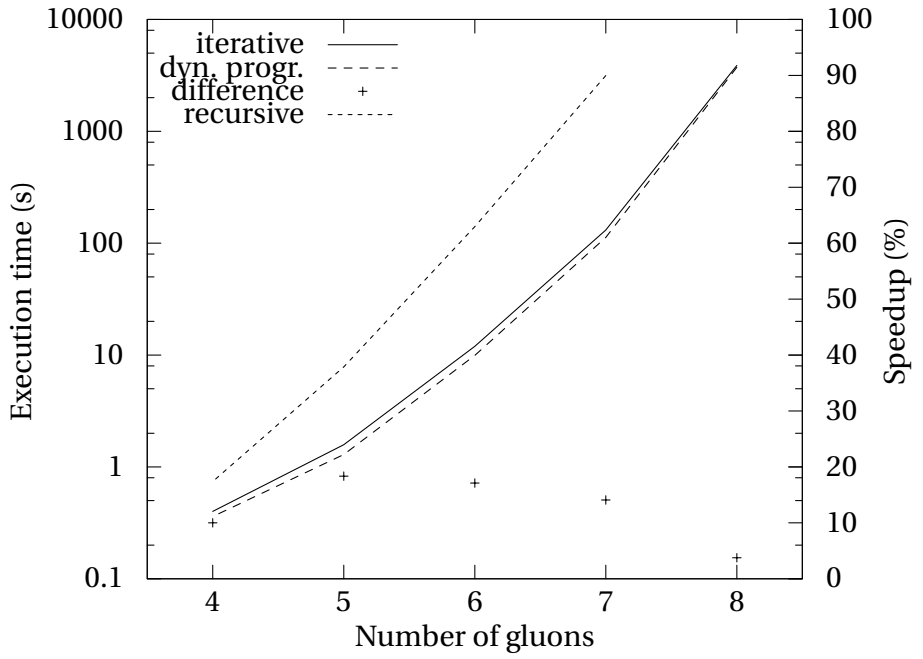


Figure B.1: *Execution time of gluon amplitudes using iterative construction, dynamic programming and naïve recursion for 10000 monte carlo integration steps. The difference is shown between iterative and dynamic programming.*

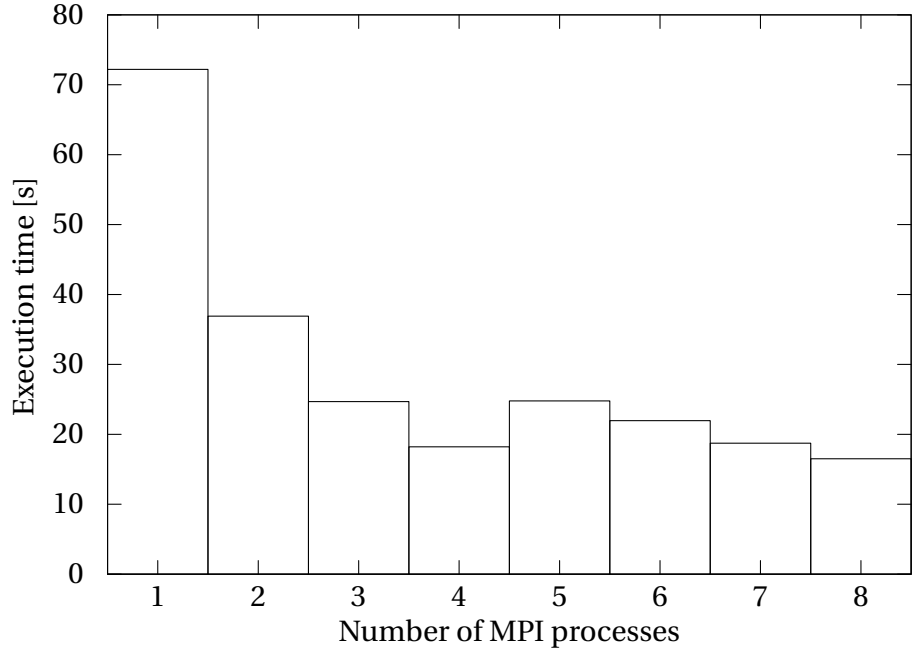


Figure B.2: *Speedup gained by running the program is parallel for the process $gg \to gg$.*

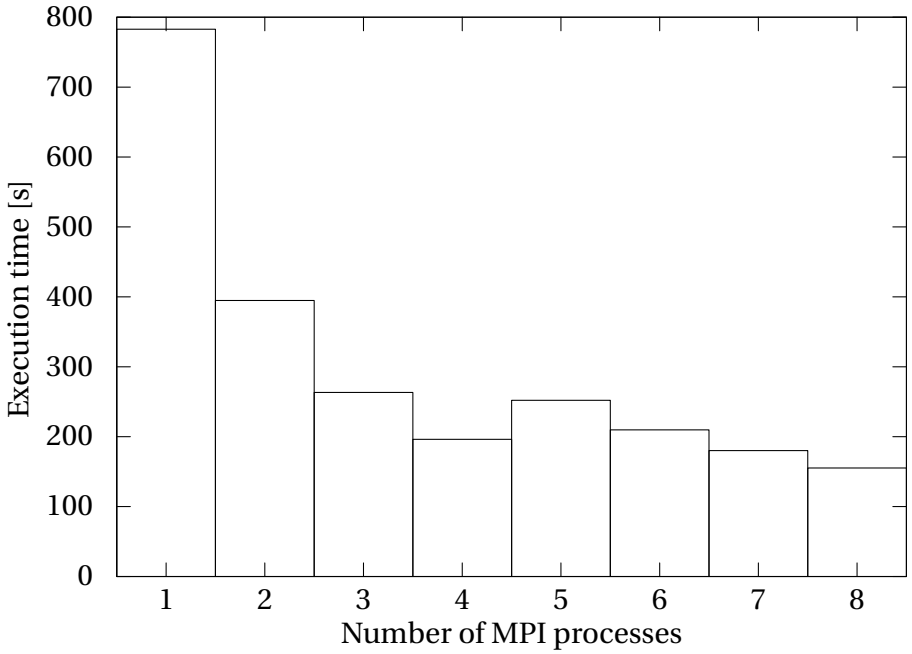


Figure B.3: *Speedup gained by running the program is parallel for the process $gg \rightarrow ggg$.*

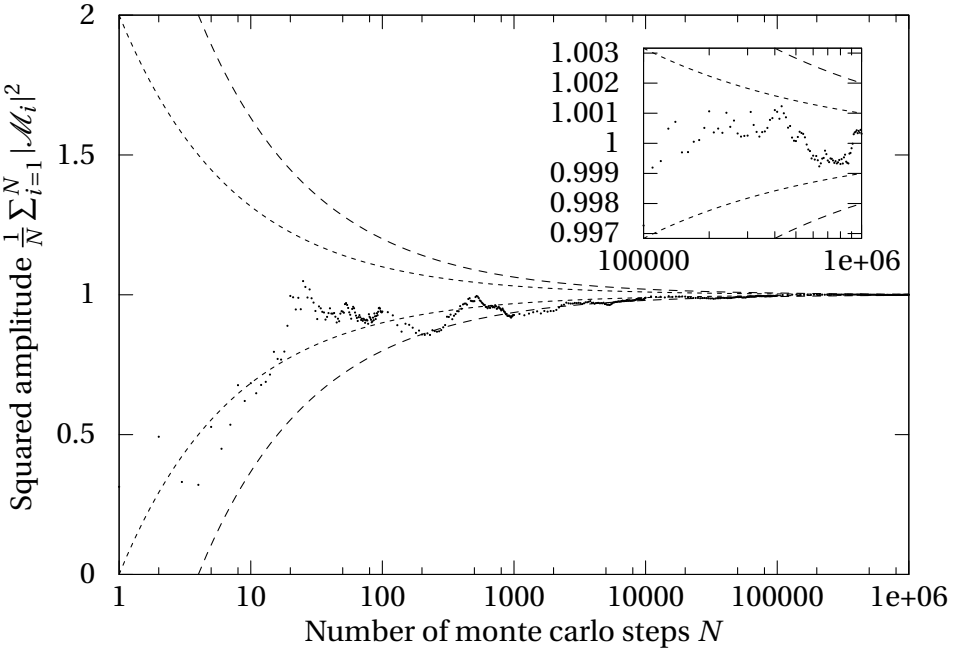


Figure B.4: *Convergence of the integration of a squared amplitude with monte carlo polarizations after N number of steps. The process is $gg \rightarrow gg$.*

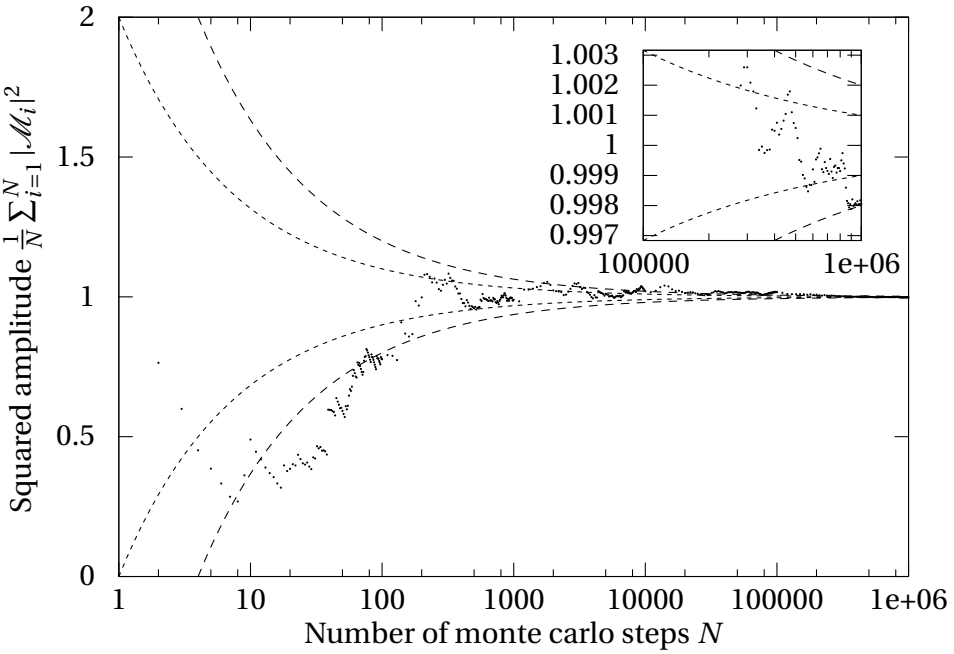


Figure B.5: *Convergence of the integration of a squared amplitude with monte carlo polarizations after N number of steps. The process is* $e^-e^+ \to \gamma\gamma\gamma\gamma\gamma$.

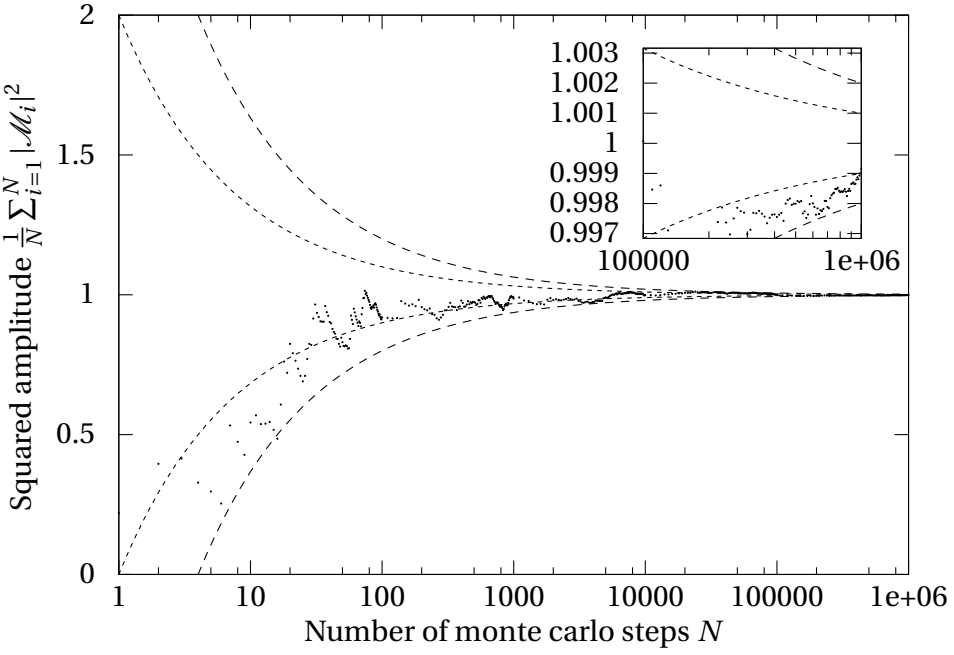


Figure B.6: *Convergence of the integration of a squared amplitude with monte carlo polarizations after N number of steps. The process is* $u\overline{u} \to \overline{d}\,d\,g$

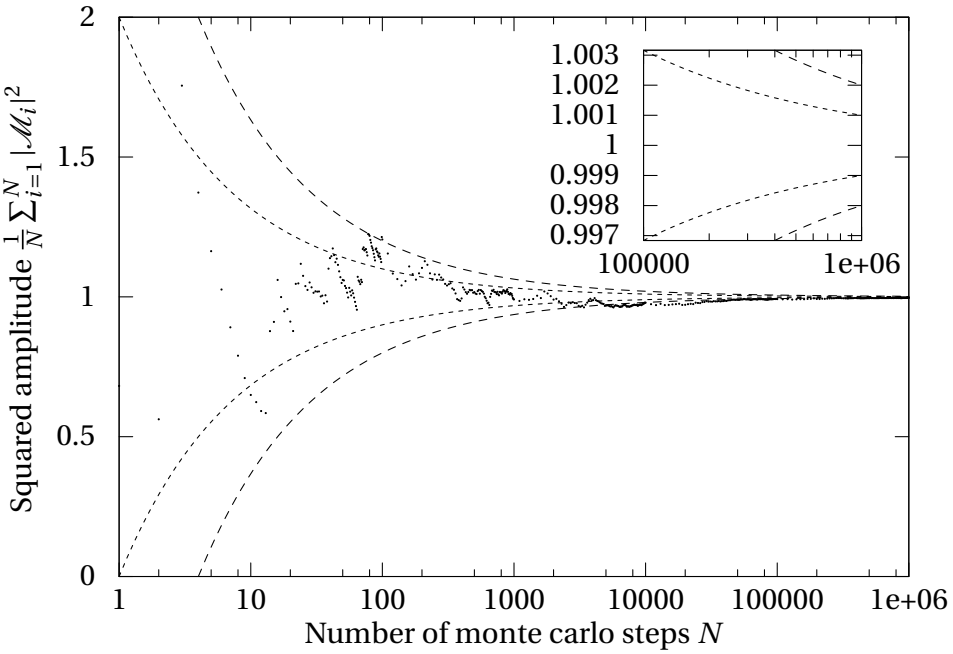


Figure B.7: *Convergence of the integration of a squared amplitude with monte carlo polarizations after N number of steps. The process is* $u\overline{u} \to ggg\gamma$

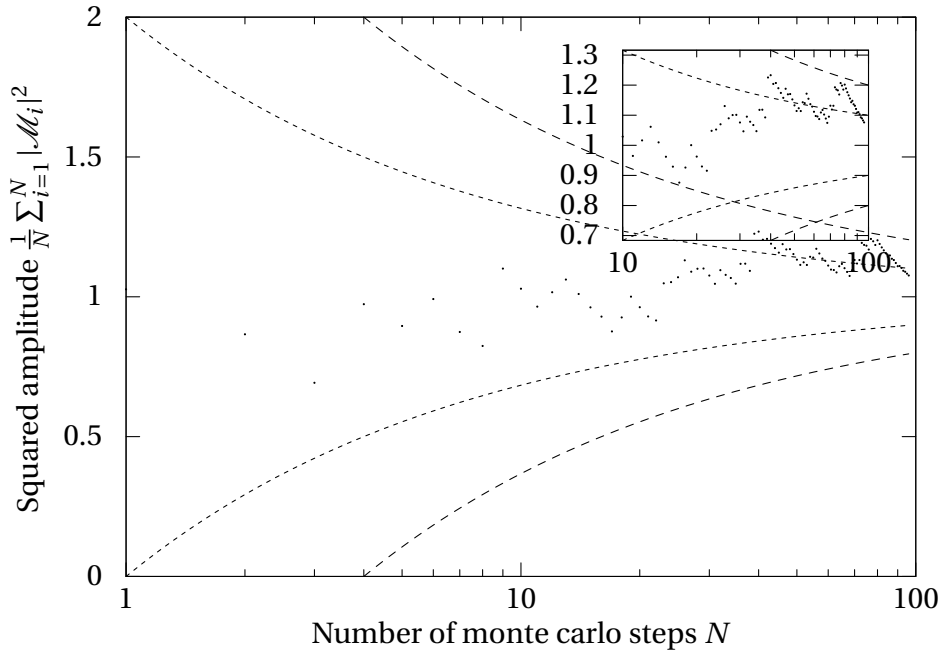


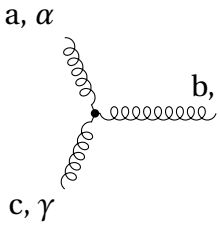
Figure B.8: *Convergence of the integration of a squared amplitude with monte carlo polarizations after N number of steps. The process is* $u\overline{u} \to \overline{d}\,d\overline{c}\,c\overline{s}\,s$

Appendix C

Feynman Rules

The Feynman rules for a general SU(N)-gauge-theory are:

- The *Three-gluon-vertex*:



$$= V_{\text{ggg}}^{abc,\alpha\beta\gamma} \tag{C.1}$$

The expression for the vertex

$$\begin{aligned} V_{\text{ggg}}^{abc,\alpha\beta\gamma}\left(p_1,p_2,p_3\right) = g f^{abc}\Big\{ & g^{\alpha\beta}\left(p_1-p_2\right)^{\gamma} + g^{\beta\gamma}\left(p_2-p_3\right)^{\alpha} + \\ & g^{\gamma\alpha}\left(p_3-p_1\right)^{\beta}\Big\} \end{aligned} \tag{C.2}$$

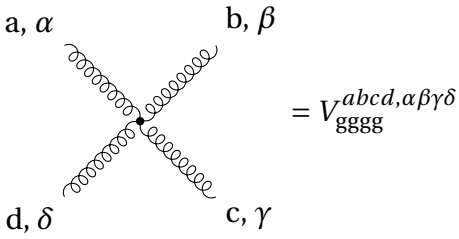
contains the structure constants f^{abc}, which may be rewritten into traces of the color matrix T^a to yield the color ordered three-gluon-vertex (see Equation 2.20 and Equation 2.17)

$$2g\sum_{P(\mathbf{1},\mathbf{2})}\text{Tr}\left(T^{\mathbf{1}}T^{\mathbf{2}}T^{\mathbf{3}}\right)V_{\text{ggg}}^{\mathbf{123}} \tag{C.3}$$

using the colorless vertex in spinor notation:

$$\begin{aligned} V_{\text{ggg}} = \frac{i}{\sqrt{2}}\Big(& \epsilon_{AC}\epsilon_{\dot{B}\dot{D}}\left(p_2-p_1\right)_{E\dot{F}} + \\ & \epsilon_{CE}\epsilon_{\dot{D}\dot{F}}\left(p_3-p_2\right)_{A\dot{B}} + \\ & \epsilon_{EA}\epsilon_{\dot{F}\dot{B}}\left(p_1-p_3\right)_{E\dot{F}}\Big). \end{aligned} \tag{C.4}$$

- The *four-gluon-vertex*:



$$\begin{aligned} V^{abcd,\alpha\beta\gamma\delta}_{\text{gggg}} = -ig^2\Big\{ & f^{abe}f^{cde}\left(g^{\alpha\gamma}g^{\beta\delta}-g^{\alpha\delta}g^{\beta\gamma}\right)+ \\ & f^{ace}f^{bde}\left(g^{\alpha\beta}g^{\gamma\delta}-g^{\alpha\delta}g^{\beta\gamma}\right)+ \\ & f^{ade}f^{bce}\left(g^{\alpha\beta}g^{\gamma\delta}-g^{\alpha\gamma}g^{\beta\delta}\right)\Big\} \end{aligned} \tag{C.5}$$

may also be rewritten into a color-ordered structure (see Equation 2.22 and Equation 2.23)

$$V^{abcd,\alpha\beta\gamma\delta}_{\text{gggg}} = 2g^2 \sum_{P(\mathbf{1},\mathbf{2},\mathbf{3})} \text{Tr}\left(T^{\mathbf{1}}T^{\mathbf{2}}T^{\mathbf{3}}T^{\mathbf{4}}\right) V^{\mathbf{1234}}_{\text{gggg}} \tag{C.6}$$

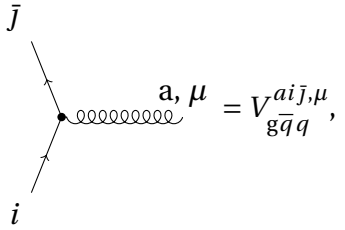
with the defintion

$$V^{\mathbf{1234}}_{\text{gggg}} = i\Big\{2g^{\mathbf{13}}g^{\mathbf{24}} - g^{\mathbf{14}}g^{\mathbf{23}} - g^{\mathbf{12}}g^{\mathbf{34}}\Big\}. \tag{C.7}$$

The kinematical part may be rewritten (see Equation 2.71) into an object carrying spinor indices only, replacing the indices $\mu_1 \to A, \dot{B}$, and so on:

$$i\Big(2\epsilon_{AE}\epsilon_{\dot{B}\dot{F}}\epsilon_{CG}\epsilon_{\dot{D}\dot{H}} - \epsilon_{AG}\epsilon_{\dot{B}\dot{H}}\epsilon_{CE}\epsilon_{\dot{D}\dot{F}} - \epsilon_{AC}\epsilon_{\dot{B}\dot{D}}\epsilon_{EG}\epsilon_{\dot{F}\dot{H}}\Big) \tag{C.8}$$

- the *antiquark-quark-gluon-vertex*:



$$V^{ai\bar{j},\mu}_{\text{g}\bar{q}q} = ig\gamma^{\mu}T^{a}_{i\bar{j}}. \tag{C.9}$$

In QED one replaces the factor $T^a_{i\bar{j}}$ by $\delta_{i\bar{j}}$ for quarks coupling to a photon, for leptons no color factor is needed.

- the *quark/antiquark-propagator*:

$$i \longrightarrow \bar{j} \;= \frac{i\left(\not{p}+m\right)\delta_{i\bar{j}}}{p^2-m^2} = \frac{i\delta_{i\bar{j}}}{p^2-m^2}\begin{pmatrix} m\delta_A{}^B & p_{A\dot{B}} \\ p^{\dot{A}B} & m\delta^{\dot{A}}{}_{\dot{B}} \end{pmatrix} \tag{C.10}$$

- the *gluon-propagator*:

$$a \sim\sim\sim b \;= \frac{ig^{\mu\nu}\delta^{ab}}{p^2} = -\frac{i\delta^{ab}}{k^2}\epsilon^{\dot{B}\dot{D}}\epsilon^{AC} \tag{C.11}$$

Appendix D

Generating Reference Results with MadGraph

We used MadGraph 5[50] v0.7.0 for generation of reference results because it allows to check matrix elements at certain phase space points. The momenta which are automatically generated by MadGraph's phase-space point generator RAMBO[53] are those we also used in Appendix A.

To generate reference results, one has to create the process first. In MadGraph 5 that is done via a Python interface, which is started by typing

```
# bin/mg5
```

in MadGraph main directory. This will start a new shell where one may enter several processes, e.g.

```
mg5>generate g g > g g @0
mg5>add process u u~ > g g @1
mg5>add process u u~ > d~ d @2
mg5>add process u u~ > u~ u @3
```

After every desired process is added in this fashion, one has to type

```
output standalone QCD
```

which creates a new directory `QCD/` where MadGraph will generate code for the computations. In order to change coupling constants, one edits the file `Cards/param_card.dat`. In our case, we changed QED's coupling constant to $1/127.899999995498$, which is entered inversely for the variable "`aEWM1`":

```
###################################
## INFORMATION FOR SMINPUTS
###################################
Block sminputs
    1 127.899999995498 # aEWM1
    2 1.166390e-05 # Gf
    3 1.180000e-01 # aS
```

and set the center-of-mass energy to 90 GeV in `SubProcesses/check_sa.f`:

```
      IF(nincoming.EQ.1) THEN
         SQRTS=PMASS(1)
      ELSE
         SQRTS=90d0              !CMS energy in GEV
      ENDIF
```

After the variables are properly set, matrix elements can be computed by changing into the corresponding directory in `SubProcesses/` and running

```
make && ./check
```

Building the program may take very long and may also consume a lot of memory, e.g. the eight-gluon-process needs about eight gigabyte RAM. In return, the execution of `check` is fast. This program prints the set of variables used, the momementa and the result for the squared matrix element.

Appendix E

Programming and Tools

The testing programs and libraries may be unpacked and compiled with:

```
# tar -xf particle_scattering-0.0.2.tar.gz
# cd particle_scattering-0.0.2
# ./configure CPPFLAGS="-DNDEBUG -DBOOST_UBLAS_NDEBUG" \
#     CXXFLAGS="-O2"
# make
# make check
```

Note that the Boost[54] libraries are required. The last make command compiles the test programs which were used to produce test results. For example

```
# tests/test_process 'g g > g g'
```

computes the helicity summed squared matrix element for the predefined momenta in Table A.4. The monte carlo results can be produced with:

```
# tests/test_mc 'g g > g g' --mc-steps 1000000
```

Invoking the program with

```
# texts/test_mc --help
```

prints a list of available options. A detailed documentation (`reference/html/index.html`) of the library can be generated with Doxygen[55]:

```
# doxygen
```

E.1 C++ and the Boost Libraries

The Standard Template Library[56] (STL) is C++'s main set of libraries that is available on every system with a conforming compiler. Though we find this library very useful, some things are missing which is provided by the Boost Libraries[54]. Libraries we used in our implementation are

- `Array` providing an STL-like class which maps an array. This is very useful for testing when data is known at compile-time,
- `Fusion` extending the concept of `std::pair`'s to an arbitrary number of elements. This is useful for returning more than one variable in a function.
- `Iterators` are a useful addition to those already available in the STL. For example, the `permutation_iterator` provides an easy access to a permutated range,
- `Math/Special Functions` are used for binomial coefficients,
- `MPI` is used to parallelize programs (see section E.2),
- `Test` is used to automatize testing precedures. We used it to register different process with reference results and to check them for correctness. Results are in Appendix A.
- `uBLAS` was employed for linear algebra (color-matrix amplitude vector computations).

E.2 Parallel integration with Open MPI

The following listing is a minimal working example showing how to use MPI for parallel integration using the Boost MPI[57] interface:

```
1 #include <cmath>
2 #include <cstdlib>
3 #include <functional>
4 #include <iostream>
5
6 #include <boost/mpi.hpp>
7
8 // function to be integrated --- f(x) = x^2
9 double square(double argument) {
```

```
10      return argument * argument;
11  }
12
13  // monte carlo integrator
14  template <typename UnaryFunction>
15  double monte_carlo_integral(UnaryFunction function, std::size_t steps) {
16      double value = 0.0;
17
18      // evaluate function at a random point and sum up result
19      for (std::size_t step = 0; step != steps; ++step) {
20          value += function(static_cast <double> (rand()) / RAND_MAX);
21      }
22
23      // average result
24      return value / steps;
25  }
26
27  int main(int argc, char* argv[]) {
28      boost::mpi::environment env(argc, argv);
29      boost::mpi::communicator world;
30
31      // seed random number generator with rank
32      std::srand(world.rank());
33
34      // steps for each process
35      const std::size_t steps = 100000000 / world.size();
36
37      std::cout << "process " << world.rank() << " will perform "
38                << steps << " steps" << std::endl;
39
40      // integrate function between [0, 1]
41      const double value = monte_carlo_integral(square, steps);
42
43      if (world.rank() == 0) {
44          double sum = 0;
45
46          // add up results from every process
47          boost::mpi::reduce(world, value, sum, std::plus<double>(), 0);
48
49          // average result and compute standard deviation
50          const double result = sum / world.size();
51          const double error = result / std::sqrt(world.size() * steps);
52
53          std::cout << "result is " << result << " +- " << error
54                    << std::endl;
55      } else {
56          // send results to process with rank 0
57          boost::mpi::reduce(world, value, std::plus<double>(), 0);
58      }
59
60      return 0;
61  }
```

To compile this program, one has to make sure that the Boost libraries and headers (see section E.1) are present and an implementation of MPI is installed, e.g. Open MPI[47].

After the program (`a.out`) has been compiled and linked (`-lboost_mpi`) it can be started as usual. Multiple instances are run with the following command:

```
# mpirun -np 8 ./a.out
```

The number (8) after the command line option `-np` specifies the number of instances which are run in parallel. The output is similar to:

```
process 0 will perform 12500000 steps
process 1 will perform 12500000 steps
process 3 will perform 12500000 steps
process 4 will perform 12500000 steps
process 5 will perform 12500000 steps
process 6 will perform 12500000 steps
process 7 will perform 12500000 steps
process 2 will perform 12500000 steps
result is 0.333356 +- 3.33356e-05
```

Bibliography

[1] Sheldon L. Glashow and Murray Gell-Mann. "Gauge theories of vector particles." In: *Annals of Physics* 15.3 (1961), pp. 437–460. ISSN: 0003-4916. DOI: 10.1016/0003-4916(61)90193-2.

[2] Steven Weinberg. "A Model of Leptons." In: *Phys. Rev. Lett.* 19.21 (Nov. 1967), pp. 1264–1266. DOI: 10.1103/PhysRevLett.19.1264.

[3] F. Englert and R. Brout. "Broken Symmetry and the Mass of Gauge Vector Mesons." In: *Phys. Rev. Lett.* 13.9 (Aug. 1964), pp. 321–323. DOI: 10.1103/PhysRevLett.13.321.

[4] Peter W. Higgs. "Broken Symmetries and the Masses of Gauge Bosons." In: *Phys. Rev. Lett.* 13.16 (Oct. 1964), pp. 508–509. DOI: 10.1103/PhysRevLett.13.508.

[5] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. "Global Conservation Laws and Massless Particles." In: *Phys. Rev. Lett.* 13.20 (Nov. 1964), pp. 585–587. DOI: 10.1103/PhysRevLett.13.585.

[6] M. Gell-Mann. *The Eightfold Way: A Theory of Strong Interaction Symmetry*. Tech. rep. California Institute of Technology Synchrotron Laboratory, Mar. 1961.

[7] G. Zweig. "An SU_3 model for strong interaction symmetry and its breaking; Part I." In: (Jan. 1964).

[8] Makoto Kobayashi and Toshihide Maskawa. "CP-Violation in the Renormalizable Theory of Weak Interaction." In: *Progress of Theoretical Physics* 49.2 (1973), pp. 652–657. DOI: 10.1143/PTP.49.652.

[9] Nicola Cabibbo. "Unitary Symmetry and Leptonic Decays." In: *Phys. Rev. Lett.* 10.12 (June 1963), pp. 531–533. DOI: 10.1103/PhysRevLett.10.531.

[10] Sebastian Becker, Christian Reuschle, and Stefan Weinzierl. "Numerical NLO QCD calculations." In: *Journal of High Energy Physics* 2010 (12 2010), pp. 1–61. DOI: 10.1007/JHEP12(2010)013. arXiv:1010.4187v2 [hep-ph].

[11] 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.

[12] S. Weinzierl. "Automated computation of spin- and colour-correlated Born matrix elements." In: *The European Physical Journal C - Particles and Fields* 45 (3 2006), pp. 745–757. ISSN: 1434-6044. DOI: 10.1140/epjc/s2005-02467-6. arXiv:hep-ph/0510157.

[13] Michael E. Peskin and Daniel V. Schroeder. *An Introduction to Quantum Field Theory*. Westview Press, 1995. ISBN: 0-201-50397-2.

[14] T. Stelzer and W. F. Long. "Automatic generation of tree level helicity amplitudes." In: *Computer Physics Communications* 81.3 (1994), pp. 357–371. DOI: 10.1016/0010-4655(94)90084-1. arXiv:hep-ph/9401258.

[15] Lance J. Dixon. "Calculating scattering amplitudes efficiently." In: (1996). arXiv:hep-ph/9601359.

[16] Stefan Weinzierl. "Automated calculations for multi-leg processes." In: *PoS* ACAT (2007), p. 005. arXiv:0707.3342 [hep-ph].

[17] G. Feldman and P. T. Matthews. "Crossing Symmetry in Meson-Nucleon Scattering." In: *Phys. Rev.* 102.5 (June 1956), pp. 1421–1422. DOI: 10.1103/PhysRev.102.1421.

[18] J. E. Paton and Chan Hong-Mo. "Generalized Veneziano model with isospin." In: *Nuclear Physics B* 10.3 (1969), pp. 516–520. ISSN: 0550-3213. DOI: 10.1016/0550-3213(69)90038-8.

[19] S. Mandelstam. "Determination of the Pion-Nucleon Scattering Amplitude from Dispersion Relations and Unitarity. General Theory." In: *Phys. Rev.* 112.4 (Nov. 1958), pp. 1344–1360. DOI: 10.1103/PhysRev.112.1344.

[20] 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.

[21] Stephen J. Parke and T. R. Taylor. "Amplitude for n-Gluon Scattering." In: *Phys. Rev. Lett.* 56.23 (June 1986), pp. 2459–2460. DOI: 10.1103/PhysRevLett.56.2459.

[22] G. t Hooft. "A planar diagram theory for strong interactions." In: *Nuclear Physics B* 72.3 (1974), pp. 461–473. ISSN: 0550-3213. DOI: 10.1016/0550-3213(74)90154-0.

[23] 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.

[24] Stefan Dittmaier. "Weyl–van der Waerden formalism for helicity amplitudes of massive particles." In: *Physical Review D* 59.1 (Dec. 1998), p. 016007. DOI: 10.1103/PhysRevD.59.016007. arXiv:hep-ph/9805445.

[25] V. I. Borodulin, R. N. Rogalev, and S. R. Slabospitsky. "CORE: COmpendium of RElations: Version 2.1." In: (1995). arxive: hep-ph/9507456.

[26] R. Kleiss and W. J. Stirling. "Spinor techniques for calculating $p\overline{p} \to W^{\pm}/Z^0 +$ JETS." In: *Nuclear Physics B* 262.2 (1985), pp. 235–262. ISSN: 0550-3213. DOI: 10.1016/0550-3213(85)90285-8.

[27] P. De Causmaecker et al. "Multiple bremsstrahlung in gauge theories at high energies (I). General formalism for quantum electrodynamics." In: *Nuclear Physics B* 206.1 (1982), pp. 53–60. ISSN: 0550-3213. DOI: 10.1016/0550-3213(82)90488-6; F. A. Berends et al. "Multiple bremsstrahlung in gauge theories at high energies (II). Single bremsstrahlung." In: *Nuclear Physics B* 206.1 (1982), pp. 61–89. ISSN: 0550-3213. DOI: 10.1016/0550-3213(82)90489-8; F. A. Berends et al. "Multiple bremsstrahlung in gauge theories at high energies : (III). Finite mass effects in collinear photon bremsstrahlung." In: *Nuclear Physics B* 239.2 (1984), pp. 382–394. ISSN: 0550-3213. DOI: 10.1016/0550-3213(84)90254-2; F. A. Berends et al. "Multiple bremsstrahlung in gauge theories at high energies : (IV). The process $e^+e^- \to \gamma\gamma\gamma\gamma$." In: *Nuclear Physics B* 239.2 (1984), pp. 395–409. ISSN: 0550-3213. DOI: 10.1016/0550-3213(84)90255-4; F.A. Berends et al. "Multiple bremsstrahlung in gauge theories at high energies: (V). The process $e^+e^- \to \mu^+\mu^-\gamma\gamma$." In: *Nuclear Physics B* 264 (1986), pp. 243–264. ISSN: 0550-3213. DOI: 10.1016/0550-3213(86)90481-5.

[28] Christian Schwinn and Stefan Weinzierl. “Scalar diagrammatic rules for Born amplitudes in QCD.” In: *Journal of High Energy Physics* 2005.05 (2005), p. 006. DOI: 10.1088/1126-6708/2005/05/006.

[29] Michelangelo L. Mangano and Stephen J. Parke. “Multi-parton amplitudes in gauge theories.” In: *Physics Reports* 200.6 (1991), pp. 301–367. ISSN: 0370-1573. DOI: 10.1016/0370-1573(91)90091-Y. arXiv:hep-th/0509223.

[30] Z. Bern, J. J. M. Carrasco, and H. Johansson. “New relations for gauge-theory amplitudes.” In: *Phys. Rev. D* 78.8 (Oct. 2008), p. 085011. DOI: 10.1103/PhysRevD.78.085011.

[31] Scott Meyers. *Effective C++. 55 Specific Ways to Improve Your Programs and Designs*. Third Edition. Addison-Wesley, 2005. ISBN: 0-321-33487-6.

[32] Scott Meyers. *Effective STL. 50 Specific Ways to Improve Your Use of the Standard Template Library*. Addison-Wesley, 2001. ISBN: 0-201-74962-9.

[33] David Vandervoorde and Nicolai M. Josuttis. *C++ Templates. The Complete Guide*. Addison-Wesley Professional, 2003. ISBN: 978-0-201-73484-3. URL: http://www.josuttis.com/tmplbook/.

[34] *C++ FAQ — Frequently Asked Questions*. URL: http://www.parashift.com/c++-faq-lite/.

[35] Erich Gamma. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1994. ISBN: 0-201-63361-2.

[36] V. Barger, A. L. Stange, and R. J. N. Phillips. “Numerical evaluation of color factors.” In: *Phys. Rev. D* 45.5 (Apr. 1992), pp. 1751–1753. DOI: 10.1103/PhysRevD.45.1751.

[37] Steven S. Skiena. *The Algorithm Design MANUAL*. Springer, 2010. ISBN: 978-1-84800-069-8. DOI: 10.1007/978-1-84800-070-4.

[38] 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.

[39] Tanju Gleisberg and Stefan Höche. “Comix, a new matrix element generator.” In: *Journal of High Energy Physics* 2008.12 (2008), p. 039. DOI: 10.1088/1126-6708/2008/12/039. arXiv:0808.3674 [hep-ph].

[40] Stefan Weinzierl. “Introduction to Monte Carlo methods.” In: (2000). arXiv:hep-ph/0006269.

[41] Donald Michie. “"Memo" Functions and Machine Learning.” In: 218 (1968), pp. 19–22. DOI: 10.1038/218019a0.

[42] Paul McNamee and Marty Hall. “Developing a tool for memoizing functions in C++.” In: *SIGPLAN Not.* 33 (8 Aug. 1998), pp. 17–22. ISSN: 0362-1340. DOI: 10.1145/286385.286386.

[43] Donald E. Knuth. *The Art of Computer Programming. Sorting and Searching.* Addison-Wesley, 1998. ISBN: 0-201-89685-0.

[44] Phil Bagwell. “Fast And Space Efficient Trie Searches.” In: (2000).

[45] Phil Bagwell. “Ideal Hash Trees.” In: (2001).

[46] *Boost combination library proposal.* URL: http://photon.poly.edu/~hbr/boost/combinations.html.

[47] *Open MPI: Open Source High Performance Computing.* URL: http://www.open-mpi.org/.

[48] *The Message Passing Interface (MPI) standard.* URL: http://www.mcs.anl.gov/research/projects/mpi/.

[49] 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.

[50] *The MadGraph Matrix Element Generator version 5.* URL: https://launchpad.net/madgraph5.

[51] Johan Alwall et al. “MadGraph/MadEvent v4: the new web generation.” In: *Journal of High Energy Physics* 2007.09 (2007), p. 028. DOI: 10.1088/1126-6708/2007/09/028. arXiv:0706.2334 [hep-ph].

[52] J. Alwall et al. “New Developments in MadGraph/MadEvent.” In: *AIP Conference Proceedings* 1078.1 (2008). Ed. by Pyungwon Ko and Deog Ki Hong, pp. 84–89. DOI: 10.1063/1.3052056. arXiv:0809.2410 [hep-ph].

[53] 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.

[54] *Boost C++ Libraries*. URL: http://www.boost.org/.

[55] *Doxygen*. URL: http://www.stack.nl/~dimitri/doxygen/index.html.

[56] *Standard Template Library Programmer's Guide*. URL: http://www.sgi.com/tech/stl/.

[57] *Boost MPI library documentation*. URL: http://www.boost.org/doc/libs/release/doc/html/mpi.html.

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Stefan Weinzierl, for giving me the opportunity to study such an interesting subject and guiding me through the course of this thesis.

I also want to thank my colleagues from our group Daniel Götz, Christian Reuschle and Sebastian Becker for many vital discussions and ideas which helped me to finish this thesis.

Last but not least I would like to thank my parents for their support. Without you this would not have been possible!