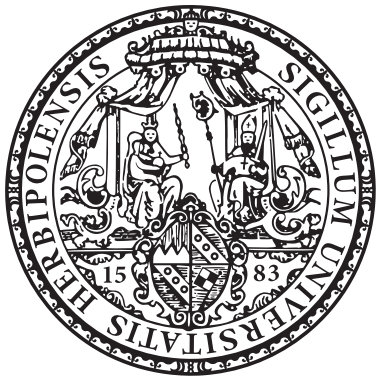
Master thesis

Numerical Calculations of Multi-Jet Cross Sections

Bijan Chokoufe Nejad

January 13, 2014



University of Würzburg

Institute for Theoretical Physics

Supervisor: Prof. Dr. T. Ohl

Contents

List of Acronyms

1POW one Particle Off-Shell Wave Function

APS Antenna Pole Structure

BSM beyond the SM

CMF Center of Mass Frame

CMC Color MC

HMC Helicity MC

LCA Leading Color Approximation

LHC Large Hadron Collider

LSZ Lehmann-Symanzik-Zimmermann

MC Monte Carlo

MHV Maximally Helicity Violating

MSSM Minimal Supersymmetric SM

OVM O'Mega Virtual Machine

PDG Particle Data Group

QCD Quantum Chromodynamics

QED Quantum Electrodynamics

SM Standard Model

SIMD Single Instruction Multiple Data

UAF Unitary Algorithm Formalism

VM Virtual Machine

List of Figures

1. Introduction

Particle physics seeks to understand the fundamental structure of nature, especially at high energies. The main tool to test these scales are scattering experiments whereby the so called high energy frontier is presently represented by the Large Hadron Collider (LHC) at CERN in Geneva [EB08]. In 2012, the LHC gained much attention due to the detection of a Higgs particle, which was earlier often called the missing part of the puzzle, at the two experiments ATLAS and CMS [ATL12; CMS12]. This most likely spin zero, parity even particle [ATL13] fits indeed very well in the Standard Model (SM) of particle physics. The SM describes reactions among all known forms of matter and its triumph is going on for almost half a century. Inspired by the discovery of broken parity by Wu [Wu57], Glashow, Weinberg and Salam formulated in the 60s a renormalizable Yang-Mills theory that unifies electromagnetism with the weak force that is responsible for beta decay [Gla61; Sal68; Wei67]. Nobody could have predicted that all of the 'many arbitrary features' [Wei67] are indeed in agreement with various measurements to the per cent level so far. The discovery of the Higgs plays hereby indeed a crucial role as it is the excitation of the field, which breaks the symmetry group $SU(2)_L \times U(1)_Y$ down to $U(1)_{\mathrm{EM}}$, where L is left, Y hypercharge and EM electromagnetic, and gives mass to the charged and neutral currents $W^{\pm}$ and Z^0 as well as the fermions.

Alongside the framework to discover all predicted particles, quantum field theory also allows to precisely compute observables. An example is the anomalous moment of electrons and muons. Theoretically predicted and experimentally measured values agree in nine and six digits, respectively [Aoy+07; Hag+11]. However, the anomalous moment plays a dual role as even this paragon of precision deviates significantly from the SM prediction by more than three standard deviations, in the case of the muon. Possible new physics beyond the SM (BSM) would affect the muon stronger than the electron due to its higher mass and could potentially explain this discrepancy. There is further hard evidence that the SM is not the complete theory of all particles. The observation of neutrino oscillations among all three generations [Abe+08; An+12] can only be reasonably explained if neutrinos have a non-zero mass, though the origin of this mass still has to be determined. Other important hints come from astronomy and astrophysics in form of anisotropies in the cosmic microwave background [Pla13], gravitational lensing [Clo+06] or rotation curves [BBS91] that could all be explained with cold dark matter, i.e. massive matter that is not emitting radiation, which is

not yet included in the SM.

Additionally, there are aesthetic and theoretical reasons that motivate to search for BSM physics. The number of parameters of the SM is rather large and does not offer any insight into the physics of flavor, e.g. we do not know why the top mass is so much larger than that of all other flavors. One might also prefer to fix all masses at one fundamental scale and compute the SM values at the electro weak scale with the renormalization group equations. A more pressing problem, however, is the lack of a stabilizing mechanism for the scalar Higgs boson. While gauge boson and fermion masses are protected from large radiative corrections through gauge and chiral symmetry, respectively, for scalar particles, like the Higgs boson of the SM, there is no such mechanism. Therefore, the mass may diverge quadratically with the highest energy scale of the theory [Ohl99a]. By plugging in the scale which is definitely not described by the SM, namely the Planck scale, we would have to absurdly fine-tune this mass term to obtain the measured value at the electro weak scale. Possible protection from radiative corrections could be e.g. offered by super symmetry or little Higgs models like the Littlest Higgs [Ark+02].

In the search for BSM physics, completely general Monte Carlo (MC) event generators are indispensable tools. When a new model is developed, the implementation should need no ingenuity on the side of the event generator but directly follow from the Feynman rules and particle content. Even the derivation and implementation of these rules from the Lagrangian, which can be tedious and error-prone for large models like the Minimal Supersymmetric SM (MSSM), has been automated with tools like FeynRules and Sarah [CD09; Sta13]. Tree-level computations are nowadays indeed fully automatized, cf. for instance CompHEP, MadGraph or Whizard [Boo+04; AHM11; KOR11], and mainly the computational complexity sets limits on the number of final states that can be computed. This work relies on and extends O'Mega [MOR01], the matrix element generator for Whizard, which has been designed with the physics of a linear collider in mind. While it can of course be used for event simulation at the LHC, there are some deficiencies. Especially the compilation of the matrix element, which is written as source code, fails due to lacking memory if the number of external gluons becomes too high, usually for eight and more. The phase space of Whizard is generated with Vamp [Ohl99b]. The sophisticated multi-channel algorithm, which tries to find appropriate mappings that render the integrand close to unity, works extremely well for some heavy particles in the final state while for many light particles, like in Quantum Chromodynamics (QCD), thousands of channels with roughly the same weight are created. To achieve sufficient statistics in each channel, the number of matrix element evaluations quickly becomes unbearable

for computations that should terminate in finite time. Therefore, we aim to develop a complementary approach that is especially suited for many light particles in the final state.

Many particle final states are indeed of pressing interest. At 14 TeV the LHC can produce large numbers of jets with sizeable cross sections, e.g. $\sigma(pp \to t\bar{t}+6\text{jets})/\sigma(pp \to t\bar{t}) \approx 0.059$ [GH08]. This trend will continue when the high energy frontier is further increased to explore nature's higher scales, either at the LHC or eventually at another hadron collider. This necessitates not only general but also fast event generators even for high multiplicities. Furthermore, recent progress on automated one-loop calculations using unitarity relies on tree-level amplitudes, not for the real emission of additional particles, but in order to compute the cut-constructable part of the one-loop amplitude as product of several tree-level amplitudes evaluated with partly complex momenta [EGK08]. Since e.g. for the computation of a six gluon one-loop amplitude the results of three, four, five and six gluon tree-level amplitudes are needed, a recursive computation, which reuses as much information as possible, should be most suitable.

In order to unleash the potential of O'Mega to its full extend, we address in this thesis the problems stated above. In Chapter 2, we show how to completely circumvent the tedious compilation of the matrix element by using a Virtual Machine (VM) without sacrificing speed compared to the compiled code. In fact, the VM can be even faster than the traditional approach due its improved memory layout. On the side of the phase space generation, we review the Sarge [HK00b] algorithm in Chapter 3, which allows to sample the phase space without further knowledge about the integrand. The generated phase space is not flat, like Rambo creates it, but has a density with the leading soft and collinear divergencies of QCD already implemented. Furthermore, we explore how a MC sampling of the discrete variables allows to speed up the calculation while maintaining the possibility to generate unweighted events with certain quantum numbers. Chapter 4 deals with helicities and the various possible choices to sample over this discrete space, whereby the samplings can be discrete or continuous. Furthermore, we gain some insight by using spin density matrices and are able to continuously vary between positive and negative helicity. Finally Chapter 5 concludes this work with some outlook to the possible extensions to the algorithms presented here.

1.1. Conventions and Notation

We use natural units where $\hbar = c = 1$. The metric corresponding to flat space time with the one true signature is $g^{\mu\nu} = \text{diag}(+1, -1, -1, -1)$ such that $p^\mu = (E(\boldsymbol{p}), \boldsymbol{p})$, $p^2 = m^2$ and $E(\boldsymbol{p}) = \sqrt{m^2 + \boldsymbol{p}^2}$. Three-vector valued objects are either notated bold-faced, like $\boldsymbol{a}$, or indexed by Latin indices, a^i. The indices of four vectors in Minkowskian space time are denoted by Greek indices or simply omitted, $p^\mu \to p$. The indices of a four vector go from 0 to 3 with the time as first component and the space components ordered as (a_x, a_y, a_z). This especially implies that four vectors with negative norm are called space-like, zero norm light-like and positive norm time-like. Pure Lorentz boosts without rotations are only a part of the proper, orthochronous Lorentz transformations and represented by $L(p)$ with the convention that $L(p)$ brings the rest vector k to p

$$p = L(p)k \equiv L(p)(m_p, \boldsymbol{0}) \Leftrightarrow L^{-1}(p)p = (m_p, \boldsymbol{0}) \ . \tag{1.1.1}$$

The subgroup is completed by including rotations, which will be denoted as $\mathcal{R}$, whereby $\mathcal{R}_p$ is the rotation that rotates the spatial part of p to the z axis

$$\mathcal{R}_p p = (E(\boldsymbol{p}), 0, 0, |\boldsymbol{p}|) \ . \tag{1.1.2}$$

The Feynman slash is defined by $\not{p} = p_\mu \gamma^\mu$, whereby here and everywhere else a sum over repeated indices is implied. This will, however, be made explicit where it improves clarity.

2. O'Mega Virtual Machine

A popular approach to compute cross sections is to derive the analytic expression itself in a higher level, often functional, programming language like `Mathematica`, `OCaml` or `Python` while the numerical evaluation is performed in high performance languages like `Fortran` or `C`. Examples for this are FormCalc, MadGraph or O'Mega [HP99; AHM11; MOR01]. Traditionally, the intermediate format is native source code which is compiled to obtain a fast program for the computation of a certain process. We show that the tedious compile step in between can be completely circumvented by using a VM without sacrificing speed compared to the compiled code.

A software VM, not to be confused with the various hardware solutions, is usually an interpreter that translates bytecode into instructions which act on its memory. In the calculation of a cross section, the number of distinct operations, which have to be performed, is related to the Feynman rules and therefore quite limited. Hence, it is a natural option to translate this typical calculation of high energy physics into an intermediate bytecode (HepBC) being later on executed by a VM. The fact that such a bytecode is portable and platform independent is a positive surplus when calculations are performed on clusters. The user is hereby freed from the time consuming necessity to compile each process, which is handy to check a lot of small processes. For a large number of external legs, $n > 8$, the attempt to compile may even fail since the compiler is unable to cope with gigabytes of native source code.

We have implemented the interpreter O'Mega Virtual Machine (OVM) in `Fortran` allowing to use the `Omega95` library for the construction, fusion and propagation of wave functions. Furthermore, `Fortran` has proven over decades to provide very fast code for number crunching tasks like the one we are facing. The production of the HepBC on the other hand is performed by O'Mega, which is written in the strongly typed, functional and object-oriented language `OCaml`. O'Mega is designed in a modular and abstract way which made it relatively straightforward to translate the native code output into HepBC. Hereby, everything is encoded in integers, which can be handled very efficiently by `Fortran`. In fact, the method of mapping distinct objects to integers, i.e. indices, is an easier and more natural output in `OCaml` than the cumbersome string handling that is necessary to obtain valid, compiling `Fortran` code.

The introduction of the HepBC format is a new level of abstraction that opens

different possibilities. The interpretation and/or production of bytecode is of course not limited to the implementations used in this context. The HepBC format could be used and extended to other event generators since calculations in terms of a VM are highly efficient, cf. Section 2.7, and offer very good parallelization potential, as demonstrated in Section 2.5. Hereby, the necessary synchronization points between threads in a calculation are represented in a clear and abstract way, independent of the implementation. Before we go into more detail, we will recap very briefly the Lagrangian that will be used to obtain scattering matrix elements in Section 2.1 and how the calculation with off-shell wave functions is performed in Section 2.2. Section 2.3 outlines the definition of the bytecode and how it is produced in O'Mega whereas Section 2.4 documents the implementation of the OVM in `Fortran`. Finally, we discuss the parallelization in Section 2.5, the possible gains in using auxiliary fields in Section 2.6 and the overall performance compared to the old `Fortran` code in Section 2.7.

2.1. Quantum Chromo Dynamics

The theoretical sand box for this thesis will be QCD. This allows to realize a faster proof of principle for the VM due to the small particle content and simple gauge structure and yields at the same time the bottleneck, with respect to computing time, background processes involved in jet production at the LHC. QCD is a $SU(N)$ gauge theory with $N = 3$. The Lagrangian is a sum of terms

$$\mathcal{L}_{\text{QCD}} = \mathcal{L}_{\text{Dirac}} + \mathcal{L}_{\text{kin}} + \mathcal{L}_{\text{gaugefix}} \,. \tag{2.1.1}$$

$\mathcal{L}_{\text{Dirac}}$ is nothing but the ordinary Dirac Lagrangian for spin 1/2 particles, which can be written with four spinors ψ as [BDJ01]

$$\mathcal{L}_{\text{Dirac}} = \bar{\psi}(\mathrm{i}\not{D} - m)\psi \,, \tag{2.1.2}$$

where we have omitted the color indices ψ_a that correspond to the representation of the matter fields, which is the fundamental and antifundamental representation of the Lie group $SU(3)$ for quarks and antiquarks, respectively, as well as the sum over flavors. The covariant derivative, which transforms as an adjoint and thereby renders Eq. (2.1.2) gauge invariant under local transformations, is given by $D_\mu = \partial_\mu + \mathrm{i}g_s A_\mu$ and can be decomposed in a basis of generators of the group $A_\mu = \sum_{C=1}^{N^2-1} A_\mu^C T^C$. An irreducible representation is given by the Gell-Mann-matrices λ^A via $T^A = \frac{1}{2}\lambda^A$.

Furthermore, the kinetic part of the non-abelian gauge fields reads

$$\mathcal{L}_{\text{kin}} = -\frac{1}{2}\,\text{Tr}\left[F^{\mu\nu}F_{\mu\nu}\right] \quad \text{with the stress tensor}$$
$$F_{\mu\nu} = -\frac{\text{i}}{g_s}\left[D_\mu, D_\nu\right] = \partial_\mu A_\nu - \partial_\nu A_\mu + ig_s\left[A_\mu, A_\nu\right] \ , \tag{2.1.3}$$

whereby the commutator seperates Quantum Electrodynamics (QED) from QCD as it vanishes in the former and gives $\text{i}A^a_\mu A^b_\nu f^{abc}T^c$ in the latter, where f^{abc} are the structure constants of $SU(3)$, which leads to three and four vertices between gauge fields and in the end to asymptotic freedom [ESW96]. The gauge fixing part

$$\mathcal{L}_{\text{gaugefix}} = -\frac{1}{2\xi}\left(\partial^\mu A_\mu\right)^2 + \mathcal{L}_{\text{ghost}} \tag{2.1.4}$$

explicitly breaks gauge invariance and choses a certain gauge. This is necessary in order to obtain a propagator in pertubative calculations, while physical results like cross sections will not depend on ξ. $\xi = 1$ is usually called Feynman or 't Hooft gauge and results in a very simple propagator and polarization sum. For our tree-level computations, we may usually use the unitary gauge with $\xi \to \infty$ and also savely neglect the Faddev-Popov term $\mathcal{L}_{\text{ghost}}$.

For small diagrammatic calculations it is sufficient to know some color algebra factors, like $\text{Tr}\left[T^aT^b\right]$, which factorize with the remaining matrix element. This is however not so easily implemented if the calculation is performed recursively with one Particle Off-Shell Wave Functions (1POWs) as reviewed in Section 2.2. Furthermore, common parton-showers expect information about the color flow of an event, in order to properly model the hadronization. In O'Mega the color flow basis is therefore implemented. The name stems from the fact that common interaction vertices reduce to weighted sums of Kronecker-δs that connect the colors [Kil+12]. The key idea is to use a $U(N) \times U(1)' \cong SU(N) \times U(1) \times U(1)'$ instead of an $SU(N)$ and subtracting the additional degrees of freedom with the help of color ghosts. More specifically, the algebraic constraints that have to be fulfilled are

$$A^\dagger = A \quad \text{and} \quad \text{Tr}\left[A\right] = 0\ . \tag{2.1.5}$$

The first can be fulfilled by using one real degree for each ordered index combination (i, j) of the $N \times N$ matrix $A^i{}_j$ instead of two. The trace condition can be satisfied by using a modified QCD theory with an additional singlet gluon A_0 as well as a phantom gluon field $\tilde{A}$ with the wrong sign in the propagator, which thereby allows to cancel the unphysical degree of freedom. $\tilde{A}$ does not couple to gluon fields but

to matter as a usual gluon with the strong coupling constant. Overall in the tree-level application, we have to compute for each different color flow the corresponding amplitude and perform a weighted sum with all interference terms

$$|\mathcal{M}|^2 = \sum_{n=1}^{F} c_{i_n j_n} \mathcal{M}_{i_n} \mathcal{M}^*_{j_n} \ , \tag{2.1.6}$$

whereby $\mathcal{M}_i$ is the matrix element for the ith color flow, F the number of color factors and c_{ij} the color factors or weights that result from the color flow basis. Explicitly it consists of the number of closed color lines L and external phantom particles E: $N_C^L(-1/N_C)^E$, whereby the weight of internal phantom propagators $(-1/N_C)^I$ has been absorbed in the phantom propagator itself [Kil+12]. Since not all matrix elements can interfere with each other, $F \neq O^2$ in general, where O is the number of color flows. For fully gluonic processes, however, the full color matrix is filled with non-zero entries.

2.2. Recursive Relations of Off-Shell Currents

Tree-level scattering amplitudes can be calculated in a variety of ways. The method known from text books is of course with the aid of Feynman diagrams. However, this approach possesses the unfavorable property that the computational complexity grows roughly factorially of the number of external particles n. This can be avoided by recursively using 1POWs [MOR01] leading to a mere exponential scaling since these are related to the number of distinct momenta that can be formed with n external particles: $P(n) = 2^{n-1} - 1$.

Given the general n-point Green's function, we can truncate $n-1$ external legs by applying the Lehmann-Symanzik-Zimmermann (LSZ) reduction formula [LSZ55] to obtain a 1POW: $W(x; p_1 \dots p_{n-1})$. These objects can be constructed recursively by dividing the set $\{p_i\}_{i=1,\dots n-1}$ into all possible partitions and summing up the respective 1POWs, e.g. for cubic couplings

$$W(x; p_1 \dots p_{n-1}) = \sum_{k+l=n-1} W(x; p_1 \dots p_k) W(x; p_1 \dots p_l) \ . \tag{2.2.1}$$

Each application of the recursive relation Eq. (2.2.1) lowers the rank, i.e. the number of summands in the momentum of the 1POW, until it terminates at the external wave functions. In order to replace the sum over Feynman diagrams T by a sum over

1POWs, one has to take care to avoid double counting, which is done in O'Mega via keystones K [MOR01]

$$T = \sum_{k,l,m=1}^{P(n)} K^3_{f_k f_l f_m}(p_k, p_l, p_m) W_{f_k}(p_k) W_{f_l}(p_l) W_{f_m}(p_m) \; . \tag{2.2.2}$$

We should mention that the crossing symmetry allows to compute

$$\begin{aligned} &\langle \bar{f}_1, -p_1; \bar{f}_2, -p_2; f_3, p_3; f_4, p_4 \,|\, T \,|0\rangle \quad \text{instead of} \\ &\langle f_3, p_3; f_4, p_4 \,|\, T \,|f_1, p_1; f_2, p_2\rangle \; , \end{aligned} \tag{2.2.3}$$

whereby the former is more symmetric and resembles a rooted tree as know from graph theory and computer science. Note finally that in the following, a fusion of two or more wave functions always implies momentum conservation and may denote either a combination to a 1POW, $\phi(p+q) = \phi(p)\phi(q)$, or a braket resulting in a complex number, $\phi(-p-q) \cdot \phi(p)\phi(q)$.

2.3. Bytecode Production in O'Mega

The bytecode has been designed such that it is in principle human readable if the meanings of the numbers are known. Tab. 2.1 summarizes the basic instructions that are needed. For the external particles, we have to distinguish between incoming and outgoing particle u and $\bar{u}$, antiparticle v and $\bar{v}$ as well as polarization vectors ϵ and ϵ^*, yielding the set

$$\texttt{X} \in \Big\{\texttt{U},\ \texttt{UBAR},\ \texttt{V},\ \texttt{VBAR},\ \texttt{VECTOR},\ \texttt{CONJ\_VECTOR}\Big\} \; . \tag{2.3.1}$$

Analogously, we have different propagators which depend on the Lorentz type and gauge as implemented in O'Mega [Ohl+12]

$$\texttt{Y} \in \Big\{\texttt{PSI},\ \texttt{PSIBAR},\ \texttt{UNITARITY},\ \texttt{FEYNMAN},\ \texttt{COL\_FEYNMAN}\Big\} \; . \tag{2.3.2}$$

The fusions should be read as `lhs_rhs` meaning that the `lhs` is constructed out of `rhs` objects or `lhs` is combined with `rhs` to a complex number which gives the possibilities

$$\begin{aligned} \texttt{Z} \in \Big\{&\texttt{VEC\_PSIBAR\_PSI},\ \texttt{PSI\_VEC\_PSI},\ \texttt{PSIBAR\_PSIBAR\_VEC}, \\ &\texttt{VEC\_VEC\_VEC},\ \texttt{VEC\_3VEC}\Big\} \; . \end{aligned}$$

Table 2.1. HepBC cheat sheet. Each instruction line consists of eight variables having a different meaning depending on the first control code. In general, the objects on the left hand side (lhs) are constructed from the right hand side (rhs). The possible values for `X`, `Y` and `Z` can be seen in Eqs. (2.3.1) to (2.3.3). The value for width indicates which type of width is used while its value and the one of the mass is inferred from the PDG code. outer_ind denotes spin and momentum index of the wave function. sym is computed from the number of identical particles involved.

code	coupl	coeff	lhs	rhs_1	rhs_2	rhs_3	rhs_4
ADD_MOMENTA	0	0	p_lhs	p_rhs_1	p_rhs_2	p_rhs_3	0
LOAD_X	PDG	0	wf	outer _ ind	0	0	amp
PROPAGATE_Y	PDG	width	wf	p	0	0	amp
FUSE_Z	coupl	coeff	lhs	rhs_1	rhs_2	rhs_3	rhs_4
CALC_BRAKET	sign	0	amp	sym	0	0	0

The above sets together with Tab. 2.1 define the language of the VM for QCD. Note that it only has to be extended to all Lorentz types in order to support arbitrary models. This very limited set of instructions as well as the objects in a calculation can each be identified unambiguously with an integer. The explicit values for the control codes of Tab. 2.1 can be found in the source code of the OVM [Cho13]. The construction of momenta with up to three summands is to some extent arbitrary but seems to be a sweet spot between caching intermediate results and minimizing lines of bytecode, thereby decode calls and memory. In any way it is not worth to heavily optimize the number of summands since the addition of momenta takes only a small amount of time compared to the whole amplitude. Due to the color flow formalism, we have a number of amplitudes, numbered by amp, which contribute to the full matrix element. The information about the corresponding amp in LOAD_X and PROPAGATE_Y is only useful for color MC and was not available in the native `Fortran` code.

The complete HepBC consists of a header, tables and a body with instructions. The header consists of the version of the VM and model library to be used as well as the numbers of objects that have to be allocated like momenta and wave functions. If these numbers are zero, an exception mechanism is provided to directly return zero as amplitude. This is motivated by the fact that if a user tries to compute a wide range of processes out of which some e.g. violate charge conservation, he should receive the appropriate, valid value zero as cross section and not be confronted with confusing `segmentation faults`. The appropriate model library has to specify `mass`, `width` and `coupl` arrays, which hold the numeric values for the different types of particles and interactions. While for particles the Particle Data Group (PDG) [Ber+12] particle codes can be used to identify the array entries, for the couplings an arbitrary but fixed

ordering has to be used. The header is followed by various tables: helicity, flavor, color flows, color ghosts, color factors and whether a flavor color combination is allowed. Color ghosts could be identified in QCD by the fact that they don't connect to a color flow since everything is charged under $SU(3)$. In more general models, containing colorless objects, like the SM, it is however important to distinguish between colorless objects and color ghosts.

The VM is now provided with all exterior information and the actual process in terms of instruction lines can follow. As discussed in more detail in Section 2.5, we can divide the computation into different levels, whereby in each level all computations may be executed in parallel. Such levels are separated by lines of zeros, whereby the first set of levels consists of summation of momenta and construction of external wave functions. Since we reuse momenta, as each momentum has at most three summands, the summation may consist of several levels if more than six external particles are involved. The following levels are defined by the number of external particle momenta in the propagator as shown in Fig. 2.2. Note, that for the calculation of a 1POW all contributing wave functions of the preceding level have to be fused and added before the wave function can propagate. To ensure this hierarchy also in a parallelized execution of the bytecode, the fusions are marked as sub instructions via a negative sign of the control code. This organisation also allows fusions to span over two or more lines which might be necessary for effective, higher dimensional operators with more than four wave functions on a vertex. Such parametrizations are useful to describe BSM physics effects in a model-independent way, see e.g. Ref. [BW86; BO12]. At the highest level of computation, the product of the remaining 1POWs is performed yielding the color dressed amplitudes: $\mathcal{M}^i$ where $i \in N_{\text{cflow}}$. For the full amplitude, a color sum over all color flows has to be carried out, taking into account the color factors given in the tables.

To export from O'Mega to HepBC, we have added an additional module to `targets`. Given `amplitudes` from the module `fusion`, sets of distinct wave functions and momenta are constructed. With such a set, it is easy to create a mapping from the abstract objects to integers from 1 to n, whereby n is the cardinality of the set, by simply folding through all objects while adding the object to a `map` and raising the `key`. These maps are then used throughout the remaining output as lookup dictionary to print the correct integers in the bytecode. A minor idiosyncrasy of wave functions is that they do not have one type in `Omega95`. Instead, there are `spinor`, `conjspinor` and `vector` wave functions in QCD as well as the other possible fields for other models. To assign the wave functions in their corresponding arrays, we sort them via pattern matching by their `Lorentz` ordering to obtain an array index. On

the side of the VM, the type can be deduced from the control code and only the index is needed. Further details can be found in the documented full module having around 1000 lines of code. Until it is part of O'MEGA, it will also be accessible over Ref. [Cho13].

2.4. Fortran Interpreter OVM

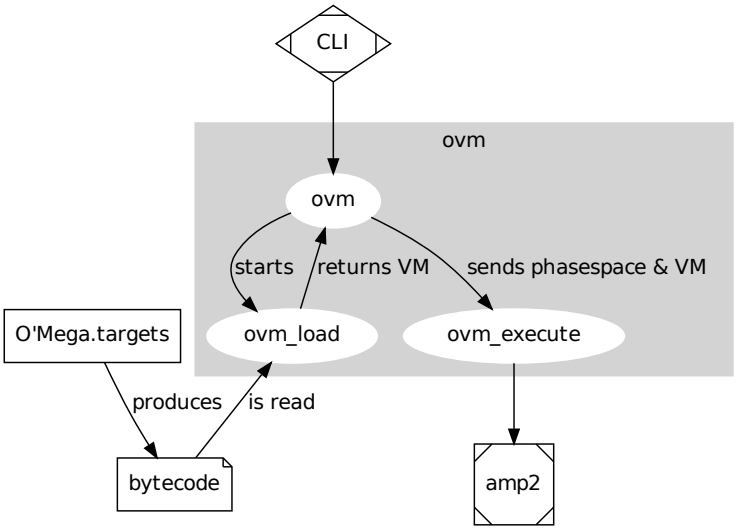


Figure 2.1. Schematic of the OVM from the command line interface to the output of the amplitude squared. While `ovm_load` is only called once for each process, `ovm_execute` is executed in each phase space point.

The layout of the interpreter is indicated in Fig. 2.1. The two main modules of the interpreter are `ovm_load` and `ovm_execute`. `ovm_load` is an encapsulation of the integer constants, the data type of a VM and run mode, public tables as well as one public method `start_VM`. The integer constants encode the different operations that can be performed, as documented in Tab. 2.1, number of integers per instruction line and header as well as codes for different modes in helicity, color and flavor MC and integrators. The data type `vm` consists of arrays for momenta, spin or θ phases[1], color, wave functions and amplitudes for different color flows. The `ovm_mode` allows to steer the way the OVM behaves. It includes a boolean for verbosity, the number of threads, used MC mode, bytecode filename, the kind of desired output as well as the file handle for this output.

By calling `start_VM(mode, ovm)`, the HEPBC file is opened and with the aid of the header, the arrays of the `vm` are allocated. Furthermore, the whole file is scanned and,

[1] See Chapter 4 for details.

to avoid speed losses, all instructions are saved to memory as `intblock`. The number of `OpenMP` threads are set once to the user defined value or estimated from the average number of instructions per level whereby the usage of one thread deactivates `OpenMP` completely. An adaptive change of the number of threads for each level has shown to be unfavorable due to the additional library calls. Finally, some sanity checks are performed to ensure that the bytecode has been loaded completely to memory. This is however not fail-safe and especially the deletion of lines in the middle of the calculation would produce no error but wrong results. A possible solution to this would be the use of checksums, like `MD5`, which are written in the head of the file.

`ovm_execute` contains the subroutines that are executed in each phase space point. `calculate_amplitudes` is public and expects the outer quantum numbers, i.e. momenta, helicities, colors and flavors, which are mapped to the results in the `amp` array. For this the external momenta of the `vm` are set, whereby the incoming obtain a minus sign due to crossing symmetry. Then, all wave functions and amplitudes are assigned to zero to allow a purely additive behaviour for the OVM. The `intblock` can now be translated into instructions by calling `decode` for each line. This subroutine can be seen as the main part of the OVM: Given a line of bytecode, it performs an operation: addition of momenta, loading external particles, propagation or scalar products of wave functions, i.e. brakets. In the latter two cases all following sub instructions, namely the associated fusions that are indicated by a negative control integer, are calculated first. These sub instructions are skipped over by the main decode function. Furthermore, `ovm_execute` contains the public methods `tr_col_hel_flv` and `tr_col` to trace out, i.e. sum over, the respective quantum numbers dependent on the selected MC mode.

Further modules are:

- The model library `ovm_parameters`, which can be produced automatically by O'Mega.
- The momentum phase space generator `ovm_phasespace` including massive and massless versions of Rambo, the massless generator Sarge, cf. Chapter 3, the cut-off on the invariant masses between all particles, setting of the incoming beams, Mandelstam variables and boosts.
- The wrapper module `ovm_spin` with various functions to generate and boost density matrices, as explained in Chapter 4, as well as generators for other phases in context of helicity.

With these extensions, the OVM source code has roughly 3000 lines of code and can be accessed over Ref. [Cho13].

2.5. Parallelization

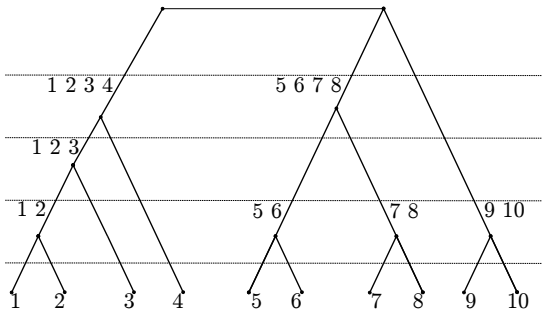


Figure 2.2. The classification of levels by the number of summands in the momenta yields an unambiguous organization of the calculation whereby each level can be calculated in parallel. Keep in mind, that this illustration is only one of thousands of possible partitions, whereby each 1POW is heavily reused.

The parallelization potential of a non-redundant, recursive algorithm is of course limited. One can only safely parallelize computations that do not depend on each other if one does not want to worry about memory coherence. However, the recursive structure of the algorithm naturally introduces levels of computation in which all instructions commute and therefore may also be executed in parallel. The level is given by the number of summands in the associated momentum of the wave function, which is depicted in Fig. 2.2. Note, that the parallelization of the native `Fortran` code has not been too successful. The OVM on the other hand has been designed with parallelization in mind, which is why the implementation could be realized by one single `OpenMP` directive:

```
!$omp parallel do
  do i = levels(j) + 1, levels(j + 1)
    call decode(intblock(:,i), i, mode, ovm, mts)
  end do
!$omp end parallel do
```

Hereby, `levels(j)` are the indices in the vector of instructions, `intblock`, at which the level changes, `j`$\in 1, \ldots N_{\text{levels}}$ and `parallel do` distributes the `do` loop to multiple threads. Furthermore, a `static schedule` is chosen, meaning that every thread gets the same number of instructions and no further communication is needed [Her02]. This

type of distribution introduces only small idle times, at the implied synchronization point after the `parallel` block, if each instruction takes roughly the same amount of computation time, which is a reasonable assumption in our case. In Fig. 2.3, it is shown that this approach indeed scales quite well with the number of processors. As a comparison, Amdahl's law [Amd67] is shown, which simply divides an algorithm into parallelizable parts p and strictly serial parts $1 - p$. Therefore, the possible speedup s for a computation with n processors is

$$s \equiv \frac{t^{(1)}}{t^{(n)}} = \frac{1}{(1-p) + \frac{p}{n}} \,. \tag{2.5.1}$$

This may also be called idealized Amdahl's law since communication costs between processors $\mathcal{O}(n)$ have been neglected in the denominator of Eq. (2.5.1). It is important to keep in mind that even in the optimal case for $n \to \infty$, the maximal speedup for an almost completely parallel program, e.g. $p = 0.95$, is only $s = 20.0$. To achieve the rather high p values of Fig. 2.3, also the color sum has been parallelized.

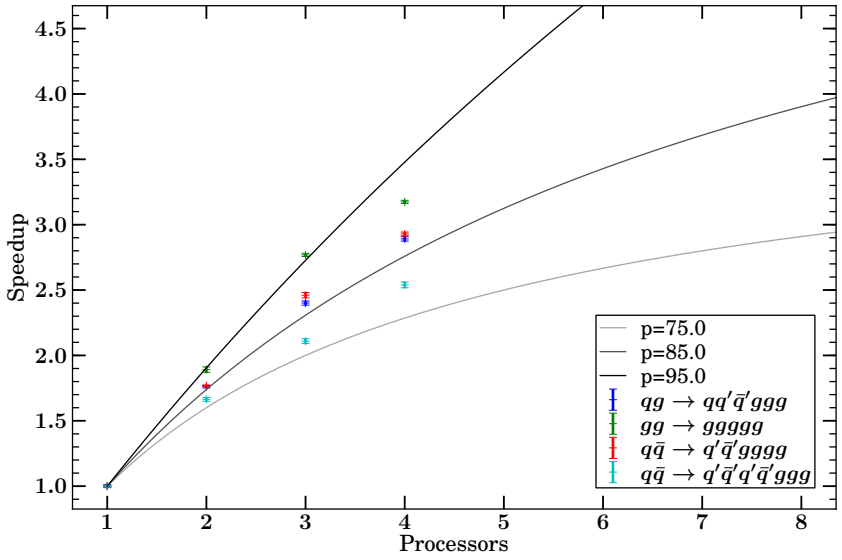


Figure 2.3. The speedup of multiple cores compared to single core execution. For the single core, the `OpenMP` calls have been deactivated. Additionally, the scaling according to Amdahl's law, Eq. (2.5.1), is shown for different fractions of parallelizable parts p of an algorithm. Depending on the process, this fraction lies for the OVM between 80 % and 95 %.

Speaking of parallelization of the calculation of a process, we should clarify whether shared or distributed memory is used. To get an idea if distributed memory is a viable option, let us consider a typical operation that is performed by the OVM:

```
pure function f_fv (gv, psibar, v) result (psibarv)
```

which performs the fusion of a `conjspinor` and a `vector` to a `conjspinor`. This operations needs $(1+4+4)$ complex numbers, which corresponds in standard double precision to 144 B, as input and produces 4 complex numbers (64 B) output. To calculate this function, 90 floating point operations are necessary [Ohl+12], if complex addition is counted twice and multiplication sixfold. Considering current commodity hardware, like a `Core-i7(3rd gen)` with ~ 100 GFLOPS, $\sim 231 \frac{\text{GB}}{\text{s}}$ average transfer rates are needed to use the full CPU performance. Such high rates are not reachable with current `DDR3-RAM` (up to $\sim 34 \frac{\text{GB}}{\text{s}}$) and probably not even with upcoming `DDR4`. Distributed memory over networks are usually at least a magnitude slower than local RAM which would result in bad performance.

The above Fermi like calculation makes two things clear. First, fast memory access is crucial for these type of calculations and locality of reference can be the deciding point for the overall performance of the algorithm. How the OVM improves on this criterion will be discussed in Section 2.7. Second, trying to compute a single phase point via distributed memory is not feasible. Distributed memory may however still be used for the full phase space integration. We might finally note that GPU programming is gaining more and more attention in the scientific community due to TFLOPS of computation power and support of double precision. The formulation of the calculation in terms of a VM can be seen as a first step in this direction. A possible implementation would once transfer the `intblock` as well as the decoder to the GPU. Then each phase point can be computed by sending the corresponding outer quantum numbers and receiving the amplitude. This minimizes the massive communication costs between GPU and main memory compared to attempts where only parts of the amplitude are computed on the GPU and other parts are sent from memory. Possible obstacles are the lacking support of `Fortran` libraries in `OpenCL` or `CUDA` and cumbersome memory management. A promising candidate for a straightforward implementation might be an open source compiler of `OpenACC`, which aims to simplify GPU programming in a similar style as `OpenMP` succeeded for parallel programming with CPUs.

2.6. Possible Gains in Using Auxiliary Fields

We digress in this section somewhat from the main topic of this chapter, to discuss a topic that is also related to the speed of the computation. Four vertices like the one in

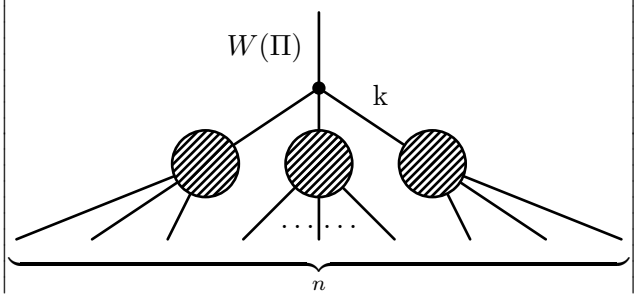
QCD can be reduced to three vertices by introducing auxiliary fields [DKP02]. This method had shown heuristically good results, i.e. a slower growth in computational complexity with the number of external legs n, though a full understanding was lacking. In Ref. [GH08] a combinatoric argument has been given which suggests that it is possible to reduce the number of vertices if the computation is performed recursively. The number of vertices or nodes is a reasonable measure for the computation time since the rest of the computation, propagator structure, external wavefunction, etc., remains untouched by the reduction method. We want to formalize the argument of Ref. [GH08] by making an important subtlety, which has been completely neglected, explicit and prove an upper bound for the number of legs n_0 of an n-point current, at which the theory with three vertices and an additional auxiliary field has less nodes than in the one with one particle type and four vertices. We emphasize that these results only apply to the recursive computation and give no information about possible gains or losses when applying this reduction in a diagrammatic computation.

Let $W(\Pi)$ be a 1POW or current with n external legs, i.e. the cardinality of the set $\Pi = \{p_i\}$ is $|\Pi| = n$. For the sake of the argument, we pretend that we have a theory with mere $N + 1$ vertices and one particle type. As mentioned earlier, the 1POW is recursively constructed. On the lowest level, the number of possible 1POW which can be formed with $(N + 1)-$vertices is

$$N_{\text{num}} = \binom{n}{N} . \tag{2.6.1}$$

On the top level, there is only one 1POW formed, $N_{\text{num}} = 1$, but there are multiple ways to partition the set Π. The second Stirling number $S(n, k)$ gives the number of possible ways to partition a set with n elements into k non-empty, mutually disjoint, unordered subsets[2]. Also the number of distinct momenta, which can be formed, is in fact a Stirling number: $S(n, 2) = 2^{n-1} - 1$. So, naïvely we have

$$S(n,k) \equiv \left\{ {n \atop k} \right\} = \left| \quad \right| \quad . \tag{2.6.2}$$



[2] In analogy to *from n chose k*, one might say *divide n in k*.

We notice however that $S(n,k)$ also counts subsets which are incommensurable to $N+1$ vertices. This is due to the fact that all trees in $W(\Pi)$ emerge from insertions of $N+1$ vertices. E.g. the subset $\{p_1,p_2\}$ cannot form a current if there are only four vertices. Let $N_{\text{not}}(n,k)$ be the number of partitions where a subset exists with cardinality $c \neq 1+i(k-1)$, $i \in \mathbb{N}_0$. This number vanishes obviously for three vertices, $N_{\text{not}}(n,2) = 0$ since all combinations are possible, and also if $n \leq k$. The total number of nodes in $W(\Pi)$ is therefore the sum over all possible currents

$$\tilde{V}(n,N) = \sum_{m=N}^{n} \binom{n}{m} \left(\left\{ {m \atop N} \right\} - N_{\text{not}}(m,N) \right) \ . \tag{2.6.3}$$

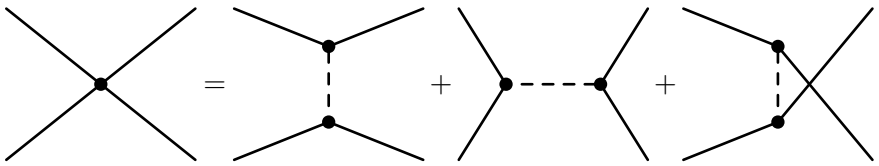
The concrete expression of $N_{\text{not}}(m,N)$ as analytic formula is rather involved. Though it is possible to obtain the number of partitions where subsets of size c are forbidden via generating functions, hereby we would have to forbid *all* $c \notin 1+i(N-1)$ since a simple sum over configurations in which each a certain c is forbidden introduces a double counting problem. The derivation with multiple c's forbidden on the other hand becomes pretty cumbersome. Hence, we simply resort to the built-in counting abilities of O'Mega to overcome this combinatorial obstacle. Furthermore, we can use Eq. (2.6.3) for all theories where ϕ^3 is included since $N_{\text{not}}(n,2) = 0$. So if we seek the number of vertices of a n point current in a theory with couplings of degree up to $d+1$, we obtain

$$\begin{aligned} V(n,d) &= \sum_{N=2}^{d} \sum_{m=N}^{n} \binom{n}{m} \left\{ {m \atop N} \right\} \\ &= \sum_{N=2}^{d} \left\{ {n+1 \atop N+1} \right\} \ , \end{aligned} \tag{2.6.4}$$

where a simple relation of Stirling numbers has been used. The validity of Eq. (2.6.4) as an exact result, has been cross checked with the counts from O'Mega. The concrete relation is that the number of fusions in an $n+1$ amplitude, without counting the sum over brakets, is equal to $V(n,d)$.

Let us now focus on the case of interest, $N = 3$. We introduce an auxiliary field without self interactions, couplings to the vanilla field and a propagator structure such that the four vertices are reproduced, whereby the original theory emerges by

integrating out the auxiliary field



(2.6.5)

The question is how Eq. (2.6.3) changes under this transformation. The counting

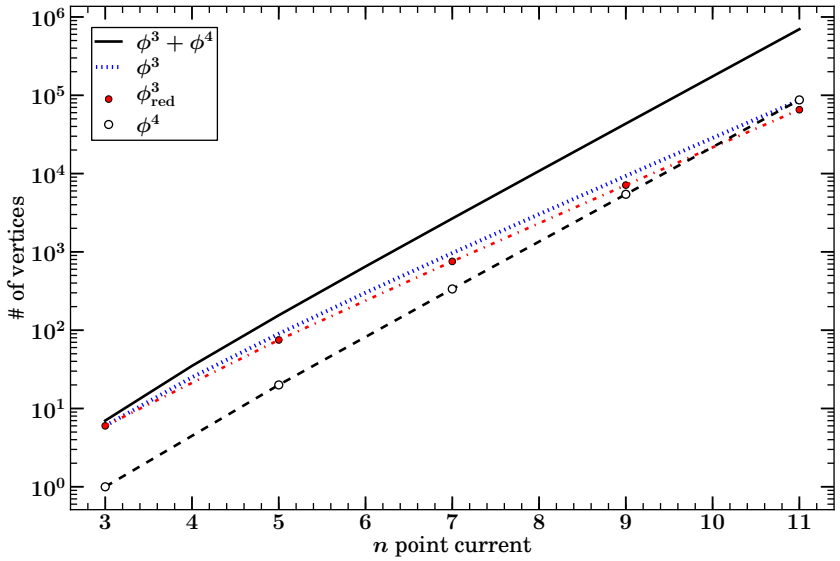
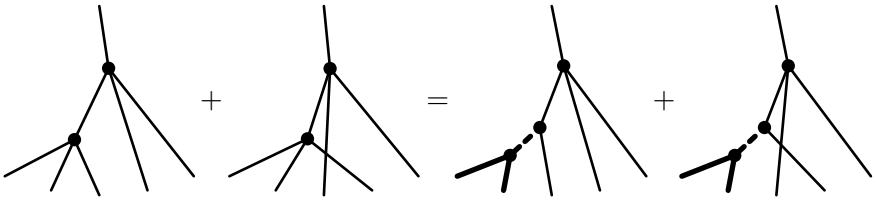


Figure 2.4. Number of vertices in different theories. The full and dotted lines for full $\phi^3 + \phi^4$ and ϕ^3 have been obtained from Eq. (2.6.4) and cross checked with O'Mega. Dash-dotted and dashed lines for $\phi^3_{\rm red}$ with the structure of Eq. (2.6.5) and ϕ^4 are counts from O'Mega. Due to the slope, pure ϕ^4 has more vertices than ϕ^3 for $n \geq n_0 = 11$. $\phi^3_{\rm red}$ has a slightly earlier break-even point $n_0 = 10$.

becomes now even more complicated due to the alternating structure of fields and auxiliary fields and the enumeration of partitions which don't fit the vertex structure in Eq. (2.6.5). It is though however to give an upper bound for the number of vertices by simply taking $V(n, 2)$, i.e. by completely neglecting this structure. The intuitive answer why there should be a gain at all in Eq. (2.6.5), is given by the possibility to reuse auxiliary fields between diagrams



(2.6.6)

The results for a range of n-point currents are shown in Fig. 2.4. We can see that the upper bound $V(n, 2)$ is actually quite close to the exact number of counts with auxiliary fields. It is herewith evident that the reduction of four vertices with auxiliary fields leads to a slower growth in computational complexity. However, this reduces, due to the intercept, only for $n \geq n_0 = 10$ the number of vertices. Since the computation time for the three vertices are not the same as for the four vertices, this does not necessarily mean that auxiliary fields are only useful when 11 or more external particles are involved. It is furthermore clear that the impact of using auxiliary fields or not is greatly reduced for arbitrary models, since three vertices remain untouched.

2.7. Performance

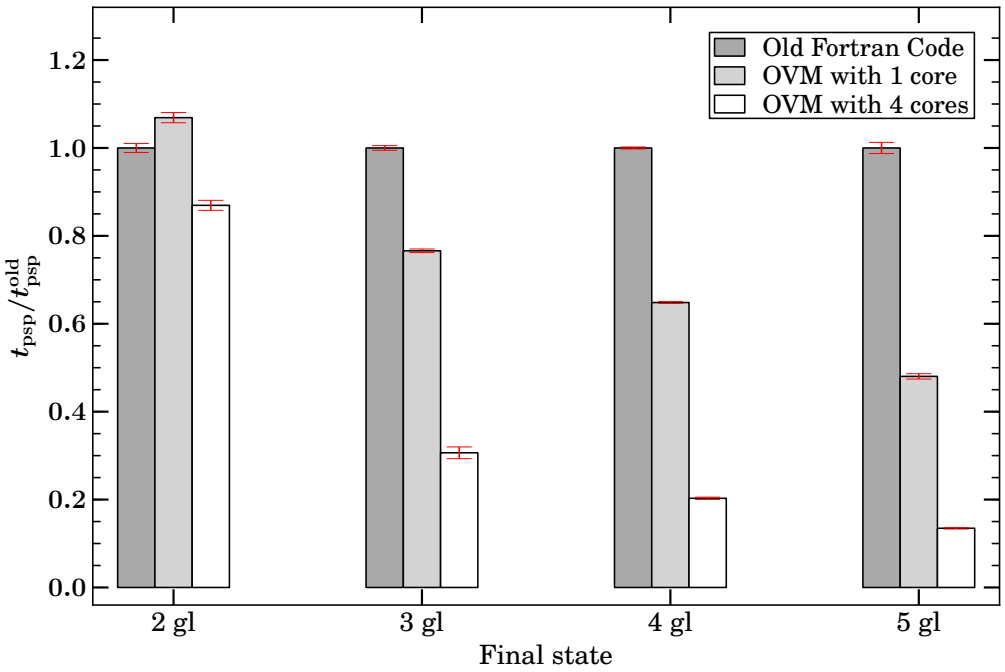


Figure 2.5. The execution time per phase space point t_{psp} of the OVM with one and four cores as well as the native `Fortran` code, normalized for each process to the native `Fortran` code. The additional overhead associated with the creation of a VM, i.e. reading the process from disc, allocation of memory and saving tables to memory, induces slightly slower execution times for the $2 \to 2$ process. However, already for three particles in the final state, the improved memory layout can compensate this and for five particles the OVM is more than a factor of two faster. This benchmark has been performed with mere calls to calculate the full amplitude while no color sum is performed in each phase space point.

To summarize this chapter, we will now report on the performance of the OVM. We note that the HepBC is about one order of magnitude smaller than the `Fortran` output. For convenience, some values together with their old compile times are shown in Tab. 2.2. The bytecode size has been furthermore roughly halfed in a later version, by using the symmetry of the color factor table, but this could have been also achieved with the `Fortran` output and is not shown here. Furthermore, the required RAM and time to produce the output with O'Mega has improved, quite considerably for many color flows, e.g. for $gg \to 6g$ from 2.17 GB to 1.34 GB and from 11 min 52 s to 3 min 35 s, while staying roughly the same for small processes.

In the native `Fortran` code, wave functions are identified uniquely by a string that is related to the color flow plus additional characters. The size of these is generally larger than the mere indices that are used in the OVM, explaining partially the reduced size of the HepBC compared with the native source code. The crucial part is that these individual variables are not as aligned in memory as the arrays of wave functions which are used in the OVM. This can significantly change the locality of reference and explain the speed up seen in Fig. 2.5. This speed up is particularly stunning since it shows that a high-performance VM can outperform a compiled code which has been optimized for years. Another possibility apart from the memory layout would be that the `Fortran` compiler leads to inefficient code compared with the line by line decoding in terms of a VM. There is though no clear evidence for this.

Table 2.2. The compilation times measured on a laptop with `i7-2720QM`, `6 GB PC3-10600 DDR3-RAM` and a `Samsung 840 SSD`. The $2g \to 6g$ process fails to compile due to lacking memory.

process	`BC` size	`Fortran` size	t_{compile}
$gg \to gggggg$	428 MB	4.0 GB	-
$gg \to ggggg$	9.4 MB	85 MB	483(18) s
$gg \to q\bar{q}q'\bar{q}'q''\bar{q}''g$	3.2 MB	27 MB	166(15) s
$e^+e^- \to e^+e^-e^+e^-e^+e^-e^+e^-e^+e^-$	0.7 MB	1.9 MB	32.46(13) s

3. Phase Space Integration

An efficient way to calculate matrix elements is a necessary but not sufficient condition for the computation of physical cross sections. A proper control of the phase space is of crucial importance for the calculation of multi-jet events. Apart from the growing complexity and dimensionality of the phase space with the number of external legs $N = n + 2$, there is also the complexity growth in the discrete variables that is at most $N_{\text{hel}} \cdot N_{\text{flv}} \cdot N_{\text{cflows}}$

$$N_{\text{hel}} = 2^N \qquad N_{\text{flv}} = 3^N \qquad N_{\text{cflows}} = (N-1)! \ . \tag{3.0.1}$$

Hereby, N_{hel} will be modified in theories with massive vector bosons, N_{flv} are the number of combinations of q, $\bar{q}$ and g without respecting charge conservation and vertex structure but assuming flavor symmetry, i.e. all flavors have the same mass, and N_{cflows} only has all permutations for pure gluon amplitudes. These discrete aspects will be partly handled in the following while the focus of this chapter is the mere momentum phase space or just phase space.

We use `noweb` [Ram94] as literate programming tool to implement SARGE from scratch and have a direct connection between source code and theory. Note that chunks of source code in `noweb` are designated by $\langle chunk \rangle$, can be concatenated via + and used within other chunks. Although implementations in `FORTRAN77` and `C` exist [HP02; Gle+09], a clear and concise minimal version of SARGE, which uses the benefits of `Fortran95`, seems to be missing. Some chunks like declarations of variables are not shown as they should be clear from the context. Section 3.1 and Section 3.2 set the stage with the necessary basics and notation for this chapter. Since the phase space integration will be performed as MC integration, we recall some general MC techniques in Section 3.3 and introduce the notion of the Unitary Algorithm Formalism (UAF), which is quite useful for the determination of the weight associated with a sampling in Section 3.4. To point out the large similarities between RAMBO and SARGE, RAMBO will be briefly reviewed in Section 3.5. Furthermore, this allows to reuse some of the RAMBO code in the formulation of SARGE in Section 3.6. In Section 3.7, we discuss the performance of the basic version of SARGE and the generated phase space density. Finally, we extend the basic version of SARGE by reducing the number of rejected points and including incoming momenta in the generation of the phase space density in Section 3.8.

3.1. Beams and Cuts

We set up our coordinate system such that the initial beams lie on the z-axis, i.e.

$$p_{1,2}^{\mu} = (E_{1,2}(p),\, 0,\, 0,\, \pm p) \quad \text{with} \quad E_{1,2}(p) = \sqrt{m_{1,2}^2 + p^2} \tag{3.1.1}$$

and hence

$$\sqrt{s} \equiv \sqrt{(p_1 + p_2)^2} = E_1 + E_2 \,. \tag{3.1.2}$$

To obtain non-divergent results that describe the physics we are interested in, i.e. the hard matrix-element of jet production, we have to introduce a cut-off in phase space. The most simple way to achieve this is via

$$s_0 \leq \left|(p_i + p_j)^2\right| \quad \forall\, i \neq j \,, \tag{3.1.3}$$

where i, j go over all external, incoming and outgoing, particles and s_0 is some fixed constant. Incomings should be crossed, $p_i = -p_i$ for $i = 1, 2$, since only those are relevant in combination with outgoing momenta due to momentum conservation. This condition on the invariant masses is sufficient to obtain an integrable cross section as it removes all parts of the phase space in which divergencies can occur in tree-level computations. These *soft* and *collinear divergencies* appear due to the propagator structure that emerges by combining external wave functions i and j, i.e. for gluons and fermions

$$\frac{1}{(p_i + p_j)^2 + \mathrm{i}\epsilon} \quad \text{and} \quad \frac{1}{\not{p}_i + \not{p}_j - m + \mathrm{i}\epsilon} = \frac{\not{p}_i + \not{p}_j - m}{(p_i + p_j)^2 - m + \mathrm{i}\epsilon} \,. \tag{3.1.4}$$

Note, that these infrared divergencies can occur both for low energies and for collinear momenta and that by restricting all external scalar products every propagator in the matrix element is regularized. One should not worry that the physics in the cut phase space regions are completely neglected. First of all, one has to keep in mind that the QCD coupling becomes strong, $\mathcal{O}(1)$, in the infrared where $Q \lesssim \Lambda_{\mathrm{QCD}}$, due to the β function of QCD and the renormalization group equation, meaning that perturbation theory fails in that regime. These strong interactions lead to a hadronization of quarks and gluons and to the observed final states in experiments. A reasonable description of the physics in the soft-collinear regions can be obtained with parton showers like Herwig [Bäh+08], Pythia [SMS08] or Sherpa [Gle+09]. These codes use tuned MC algorithms to generate the full complex final states of jets of hadrons.

By applying cuts to our computation, we therefore only delay the physics in these phase space regions for a later treatment. Notably, even detectors cannot resolve the finest collinear and infrared event structure, thereby giving even a physical cut-off.

In a massless theory, the squared sum of outgoing momenta equals

$$\begin{aligned} s &= \left(\sum_{i=1}^{n} p_i\right)^2 \\ &= \underbrace{2p_1p_2 + ... + 2p_{n-1}p_n}_{\frac{n(n-1)}{2}\text{ terms}} . \end{aligned} \tag{3.1.5}$$

Hence, a convenient parametrization of s_0 is given by

$$s_0(\tau) = \tau\, s\frac{2}{n(n-1)} , \quad \tau \in [0,1] . \tag{3.1.6}$$

This means that for $\tau = 0$ no cut is applied, while for $\tau = 1$ the cut-off is larger than kinematically allowed and every set of momenta will be rejected. This can be related to the more physically motivated cuts like minimal transverse momentum p_T and angle between partons θ_0 via

$$p_\mathrm{T} \leq \sqrt{p_x^2 + p_y^2} \tag{3.1.7}$$

$$\cos\theta_0 \leq \frac{\boldsymbol{p}_i \cdot \boldsymbol{p}_j}{\|\boldsymbol{p}_i\| \left\|\boldsymbol{p}_j\right\|} \tag{3.1.8}$$

$$s_0 = 2p_T^2 \cdot \min\left(1 - \cos\theta_0, \left(1 + \sqrt{1 - \frac{p_T^2}{s}}\right)^{-1}\right) \tag{3.1.9}$$

Since we need *all* scalar products for the weight of SARGE later on, we save them to `pdotp`. $(p_i + p_j)^2 = 2p_ip_j$ only holds in the massless case. The cut refers to the lhs but the rhs is less expensive which is why we use it if possible.

⟨*Implementation of* `ovm_phasespace` *procedures*⟩≡

```
subroutine update_pdotp(p,massless)
  ⟨Variables of update_pdotp⟩
  ⟨Initialization of pdotp(N,N)⟩
  ⟨Update pdotp(N,N)⟩
end subroutine update_pdotp
```

Now we can check if the momenta fulfill Eq. (3.1.3)

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
function mom_is_allowed (s0) result (yorn)
```

```
    ⟨phase space cuts declarations⟩
    ⟨Compare pdotp with s0⟩
  end function mom_is_allowed
```

Cutting the phase space reduces its volume of course. In a MC sampling this can be reflected by assigning to each rejected point a zero integrand. Herewith, one does not need to multiply the result with the ratio of accessible to total phase space volume in the end.

3.2. Notation and Conventions

A possible parametrization of a three vector with unit length is given in cylindrical coordinates $\hat{n}(z,\varphi)$ with

$$\hat{n}_1(z,\varphi) = \sqrt{1-z^2}\,\sin\varphi\;, \quad \hat{n}_2(z,\varphi) = \sqrt{1-z^2}\,\cos\varphi \quad \text{and} \quad \hat{n}_3(z,\varphi) = z\;. \tag{3.2.1}$$

⟨*standard vector n(z,phi)*⟩≡

```
  s = sqrt(one-z**2)
  n(1) = s * sin(phi)
  n(2) = s * cos(phi)
  n(3) = z
```

The standard boost $L(p)$ takes the rest frame momentum $k = m_p e_0$ to p. The function `std_boost` returns $q'^{\mu} = L^s(p)^{\mu}{}_{\nu} q^{\nu}$ where the `integer` $s \in \{+1, -1\}$ allows to switch between the boost and the inverse

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
  function std_boost (s, p, q) result (q_pr)
    ⟨standard boost implementation⟩
  end function std_boost
```

The restriction for physical momenta is denoted by

$$\vartheta_m(p) = \delta\left(p^2 - m^2\right)\theta(p^0) \quad \text{with} \quad \vartheta(p) = \vartheta_0(p)\;. \tag{3.2.2}$$

The hard cross section may be defined as

$$\mathrm{d}\sigma = \frac{(2\pi)^{4-3n}}{4\sqrt{(p_1 p_2)^2 - m_1^2 m_2^2}}\,\mathrm{d}V_n(\sqrt{s})\;\;|\mathcal{M}|^2\;, \tag{3.2.3}$$

which is a product of a flux factor, the differential phase space volume and the squared matrix element, summed and averaged over unobserved quantum numbers.

The factors of 2π imply that our states are normalized according to [BDJ01]

$$\langle p \,|\, p' \rangle = (2\pi)^3 2E(\boldsymbol{p})\delta\left(\boldsymbol{p} - \boldsymbol{p}'\right) \quad \text{and} \quad \bar{u}(p, s_3)u(p, s_3') = 2m\delta_{s_3, s_3'} \tag{3.2.4}$$

and are sometimes also absorbed in the Lorentz invariant phase space volume $\mathrm{dLIPS} = (2\pi)^{4-3n}\mathrm{d}V_n(\sqrt{s})$. The denominator of the flux factor significantly simplifies for massless momenta to $2s$. The phase space volume ensures overall momentum conservation as well as physical momenta, i.e. on the mass shell and with positive energies,

$$\mathrm{d}V_n\left(\sqrt{s}\right) = \delta\left(p_1 + p_2 - \sum_{i=1}^{n} p_i\right) \prod_{i=1}^{n} \underbrace{\left(\frac{\mathrm{d}^3 p_i}{2E(p)}\right)}_{= \mathrm{d}^4 p_i \, \vartheta_m(p_i)} \equiv \prod_{i=1}^{n} \Theta_n(\sqrt{s})\mathrm{d}^4 p_i \;, \tag{3.2.5}$$

where the underbraced identity follows by integrating over $\mathrm{d}p_i^0$ and using the properties of the Dirac delta distribution.

3.3. General Monte Carlo Techniques

A MC integration can be used to compute an integral

$$\mathrm{I}f = \frac{1}{\mathrm{Vol}\,\Omega} \int_\Omega \mathrm{d}\boldsymbol{x}\; f(\boldsymbol{x}) \equiv \langle f \rangle \;, \tag{3.3.1}$$

where Ω is the integration region, the volume is $\mathrm{Vol}\,\Omega = \left(\int_\Omega \mathrm{d}\boldsymbol{x}\right)$ and $\boldsymbol{x}$ a d dimensional vector. By dividing by the volume, we directly recognize the integration as an average of the function f. This deterministic averaging can be made stochastic by using random numbers $\boldsymbol{x}_i$. The law of large numbers states that the estimated mean of these random samplings $\bar{f}_N$ converges to the real mean $\langle f \rangle$ for $N \to \infty$. The associated error is

$$\delta I = \sqrt{\frac{\mathrm{Var}\, f}{N}}, \qquad \mathrm{Var}\, f = \left\langle f^2 \right\rangle - \langle f \rangle^2 \equiv \sigma_f^2 \;. \tag{3.3.2}$$

The important point is that this error is *independent* of the integration dimension opposed to the one of multiple quadrature rules that will grow with the number of dimensions. The explicit estimators of mean and variance in terms of the sampled points, including the recursive versions, can be found in Appendix A. Usually some method is employed to minimize the variance in Eq. (3.3.2), also known as *variance reduction*, to ensure faster convergence. The two main approaches are importance sampling and stratified sampling. Importance sampling introduces a non-uniform

sampling with a weight function $g(\boldsymbol{x})$, similar to $f(\boldsymbol{x})$, such that

$$\mathrm{I}f = \frac{1}{\mathrm{Vol}\,\Omega}\int_\Omega g(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\,\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \equiv \left\langle \frac{f}{g} \right\rangle_g \tag{3.3.3}$$

where $\langle \circ \rangle_g$ denotes a non-uniform sampling according to $g(\boldsymbol{x})$. If $f/g \approx 1$ the variance of the integrand and therefore the error according to Eq. (3.3.2) will be way smaller. As we will see below, to obtain a sampling with $g = f$, one needs to solve the integration problem itself, wherefore an approximation with a weight function which one *can* sample must suffice. The classical example is VEGAS [Lep78] which uses a piecewise constant weight function, whereby the bins are adapted such that many steps occur in regions where f is large. Stratified sampling on the other hand usually bisects the integration region along a chosen dimension and distributes the points according to the estimated variance in the subregions. This method is basically limited by the estimate of the variance. Using insufficient points in a region before subdividing can lead to a true underestimate of the variance and sampling of the function therein.

A suitable method to obtain an arbitrary probability density, which will be used on some occasions throughout this work, is called *inversion*. To generate $z \in \mathcal{M}$, where $\mathcal{M}$ is a $d-$manifold, with a probability density F, one needs a bijective function $\phi : \mathcal{K} \to \mathcal{M}$. Then one can generate $y \in \mathcal{K}$[1] uniformly to obtain $y \mapsto \phi(y) = z$. This generates a density for z according to the Jacobian $\left|J_{\phi^{-1}}(z)\right| = \int_\mathcal{K} \mathrm{d}y\, \delta\left(z - \phi(y)\right)$, which is easily proven with a variable substitution from $y \to \phi^{-1}(y)$. The mapping has to be chosen such that $F = \left|J_{\phi^{-1}}\right|$, which is an integration and inversion problem that represents often a non-trivial task, even in one dimension. An easy example is the generation of a density $F(z) = 1/z$. By integration follows $\phi^{-1}(z) = \log z$ and thusly the required mapping is $\phi(y) = \exp y$. The implicit constants are fixed by the support of F. If $z \in [z_0^{-1}, z_0]$, then $y \in [-\log z_0, \log z_0]$. Now, it should also be clear why one cannot use a weight function that is exactly the integrand without solving the problem itself.

3.4. Unitary Algorithm Formalism

A useful tool to translate algorithms into probability distributions while keeping track of the normalizations is the UAF, see for instance [Kle00]. The name stems from the

[1] Typically $\mathcal{K} = (0,1)^d$ since random numbers are most easily created on the unit hypercube.

fact that drawing a random number $x \in (0,1)$ results in

$$1 = \int_0^1 \mathrm{d}x \ . \tag{3.4.1}$$

The identification of an integral as a mean that can be replaced by a statistical mean, i.e. a MC sampling, has of course already been used in the last section. But here we are only integrating a 1 and are interested in the density that is created when we make assignments with a variable. In this way, we can compute the weight of an algorithm by translating it to resolved Dirac delta distributions. E.g. if we are not only generating a random number x, but also set $y = -\log\{ax\}$, we have

$$1 = \int_0^1 \mathrm{d}x \int_{-\infty}^{\infty} \mathrm{d}y\, \delta\Big(y + \log\{ax\}\Big) = \int_{-\infty}^{\infty} \mathrm{d}y \underbrace{\theta\Big(y + \log\{a\}\Big) \frac{1}{a} \mathrm{e}^{-y}}_{\mathcal{P}(y)} \ . \tag{3.4.2}$$

Since an integral over a delta distribution always yields one, we are sure that the whole expression is still unitary. As mundane as this may seem, we have hereby obtained the density $\mathcal{P}(y)$, which is generated for y, with the correct normalization and boundaries by simple Dirac distribution calculus. A more interesting example might be the proof of the correctness of the *rejection* method as it involves more than a single assignment. Say we want to generate y with a distribution proportional to an arbitrary, non-negative $g(y)$, with $g(y) \leq C$, $\forall y \in (0,1)$. This means that we know the desired distribution function but cannot integrate or invert it. The goal can still be achieved by sampling points x_1 uniformly in $(0,1)$ and accepting the result with probability $p = g(x_1)/C$ and otherwise drawing again. Note that we have hereby two random numbers involved x_1 and x_2, when we translate the recursive procedure literally with the UAF to an integral

$$\begin{aligned}
\mathcal{P}(y) &= \int \mathrm{d}x_1 \mathrm{d}x_2 \left[\theta\left(x_2 \leq \frac{g(x_1)}{C}\right) \delta\left(y - x_1\right) + \theta\left(x_2 > \frac{g(x_1)}{C}\right) \mathcal{P}(y)\right] \\
&= \int_0^{g(y)/C} \mathrm{d}x_2 \left(1\right) + \mathcal{P}(y) - \int \mathrm{d}x_1 \frac{g(x_1)}{C} \mathcal{P}(y) \\
&= \frac{g(y)}{\int \mathrm{d}x_1\, g(x_1)} \ .
\end{aligned} \tag{3.4.3}$$

Note that we have hereby omitted the integral over y which is why we have on the left hand side the generated density $\mathcal{P}(y)$ and not 1. We are thus not only generating a distribution proportional to $g(y)$ but also with the correct normalization such that $\int \mathrm{d}y\, \mathcal{P}(y) = 1$. The full power of the formalism will become more evident in the high-dimensional application in the next sections.

3.5. RAMBO - RAndom Momenta Beautifully Organized

The RAMBO algorithm allows to generate n massless momenta `p` with a provable flat phase space density, i.e. with constant weight for each sampled point, whereby the sum of the momenta is equal to $P = (\sqrt{s}, \mathbf{0}) \equiv \sqrt{s}\, e_0$ [KSE86]. This means that the phase space density is merely $\Theta_n(\sqrt{s})$ and has no dependency of the generated set $\{p\}_n$. Events, which are generated this way, have the weight of the phase space volume $W_0 = V_n(\sqrt{s})$. The algorithm consists of the following steps

1. *Generate massless vectors* q_j, i.e. $q_j^2 = 0$, with positive energy, which is drawn from a probability density $f(q_j^0)$, but without mutual constraints between different momenta. For this, we need an azimuthal angle `phi` and the `z` component between [-1,1]

   ⟨*Generate massless vectors*⟩≡
   ```
   do j = 1, size (p, dim = 2)
     call tao_random_number(ran)
     ⟨q(0,j) drawn from density⟩
     z = 2 * ran(3) - 1
     phi = twopi * ran(4)
     ⟨standard vector n(z,phi)⟩
     q(1:3,j) = q(0,j) * n
   end do
   ```

   As shown below, we have to draw q_j^0 from a density $xf(x) = x\exp(-x)$ for a unit weight in the whole phase space, which is generated by the mapping

   ⟨`q(0,j)` *drawn from density*⟩≡
   ```
   q(0,j) = - log(ran(1)*ran(2))
   ```

2. *Determine* the *Lorentz boost* and scaling `x` to bring $q_{(n)} = \sum q_j$ to $\sqrt{s}\, e_0$

   ⟨*Determine Lorentz boost*⟩≡
   ```
   qsum = sum (q, dim = 2)
   m_q = dot (qsum, qsum)
   ⟨Numerical checks⟩
   m_q = sqrt(m_q)
   x = roots / m_q
   ```

3. *Boost all* q_j with this Lorentz transformation and return them as `p`

   ⟨*Boost all*⟩≡
   ```
   do j = 1, size(p, dim = 2)
     r = dot (q(0:,j), qsum) / m_q
   ```

```
      zz = (q(0,j) + r) / (qsum(0) + m_q)
      q(1:3,j) = x * (q(1:3,j) - qsum(1:3) * zz)
      q(0,j) = x * r
    end do
```

Putting the pieces together, we obtain

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
subroutine massless_isotropic_decay(roots, p)
  ⟨RAMBO declarations⟩
  ⟨Generate massless vectors⟩
  ⟨Determine Lorentz boost⟩
  ⟨Boost all⟩
  p = q
end subroutine massless_isotropic_decay
```

A useful quantity to understand the RAMBO algorithm is

$$R_n \equiv \int \prod_{i=1}^{n} \mathrm{d}^4 q_i \, f(q_i^0)\vartheta(q_i) = \int \mathrm{d}R_n \ . \tag{3.5.1}$$

By performing the trivial integration over the solid angle and over the radius, which is fixed by the delta distribution, we obtain

$$R_n = \left(2\pi \int \mathrm{d}q^0 f(q^0) q^0\right)^n \tag{3.5.2}$$

which results for $f(x) = \mathrm{e}^{-x}$ in $R_n = (2\pi)^n$. This means, that if we sample random massless momenta with energies that are distributed according to $x \cdot \exp\{-x\}$, the probability to pick a certain set $\{q\}_n$ in a volume element ($\prod \mathrm{d}^4 q_i$) is $1/(2\pi)^n$. By formulating the whole algorithm with the UAF, we generate a density

$$\Phi(\{p\}_n) = \int \frac{\mathrm{d}R_n}{(2\pi)^n} \, \mathrm{d}^4 b \, \delta^4\left(b - \frac{q_{(n)}}{m_{q_{(n)}}}\right) \, \mathrm{d}x \, \delta\left(x - \frac{\sqrt{s}}{m_{q_{(n)}}}\right) \prod_{i=1}^{n} \delta^4\Big(p_i - xL^{-1}(b) q_i\Big) \ . \tag{3.5.3}$$

Note that $\delta^4(b - q_{(n)}/m_{q_{(n)}}) = \delta^4(p_{(n)} - \sqrt{s}e_0)s^2$ by using Lorentz invariance of the four volume element, $q_{(n)} = (1/x)L(b)p_{(n)}$, and $m_b = 1$. On the other hand, we have $\delta(x - \sqrt{s}/m_{q_{(n)}}) = \delta(1 - b^2)(2/x)$ which follows from the scaling property of the delta distribution and $m_b = \sqrt{s/(x^2 m_{q_{(n)}}^2)}$ as well as the fact that b can only have positive mass. Furthermore, the energy has to be positive since all summands are positive, i.e. $\theta(b^0)$. It is worth to explicitly multiply the density with this Heaviside distribution in

order to keep track of it, without changing the value of the integrand. Furthermore, we can combine the final assignments with $\mathrm{d}R_n$ to

$$\begin{aligned}\mathrm{d}^4q_j\delta\left(q_j^2\right)\delta^4\Big(p_j - xL^{-1}(b)q_j\Big) &= \mathrm{d}^4q_j\delta\left(q_j^2\right)\frac{1}{x^4}\delta^4\Big(q_j - \frac{1}{x}L(b)p_j\Big)\\ &= \frac{1}{x^2}\delta\left(p_j^2\right)\ .\end{aligned} \tag{3.5.4}$$

Finally, we will use that $\prod_i \exp\{-q_i^0\} = \exp\{-\sqrt{s}/xb^0\}$. Plugging all this into Eq. (3.5.3) gives

$$\begin{aligned}\Phi(\{p\}_n) &= \int \mathrm{d}^4b\,\Theta_n(\sqrt{s})\,\frac{1}{(2\pi)^n}\,\underbrace{\frac{1}{x^{2n+1}}\mathrm{e}^{-\frac{b^0\sqrt{s}}{x}}\,\mathrm{d}x}_{=(b^0\sqrt{s})^{-2n}\Gamma[2n]}\,2s^2\,\delta(b^2-1)\theta\left(b^0\right)\\ &= \Theta_n(\sqrt{s})\,\frac{2\Gamma[2n]}{(2\pi)^n s^{n-2}}\int \mathrm{d}b^0\mathrm{d}b^r\,4\pi(b^r)^2\delta\left((b^0)^2-(b^r)^2-1\right)(b^0)^{-2n}\theta\left(b^0\right)\ .\end{aligned} \tag{3.5.5}$$

A subtle but important point is that after integrating out b_r, the integration region of b_0 reduces from $[0,\infty)$ to $[1,\infty)$, which makes the integral convergent. With

$$\int_1^\infty \mathrm{d}b^0\,(b^0)^{-2n}\sqrt{(b^0)^2-1} = \frac{\sqrt{\pi}}{4}\frac{\Gamma[n-1]}{\Gamma[n+\frac{1}{2}]} \tag{3.5.6}$$

and some simplifications between the Gamma functions, we obtain the well known result

$$\Phi(\{p\}_n) = \Theta_n(\sqrt{s})\,\left(\frac{2}{\pi}\right)^{n-1}\frac{\Gamma[n-1]\Gamma[n]}{s^{n-2}} = \Theta_n(\sqrt{s})\,\frac{1}{V_n(\sqrt{s})}\ . \tag{3.5.7}$$

Hence, an integration over $\prod \mathrm{d}^4p_i$ yields one, which verifies that Eq. (3.5.7) is indeed a properly normalized probability density.

3.6. SARGE - Staggered Antenna Radiation GEnerator

The aim is to generate n momenta with the Antenna Pole Structure (APS)

$$\frac{1}{(p_1p_2)(p_2p_3)\cdots(p_{n-1}p_n)(p_np_1)} \tag{3.6.1}$$

and permutations thereof, which is known to be the dominant divergence structure in QCD processes, like the one depicted in Fig. 3.1 due to the propagators of the antennae. The explicit analytic formulae for $n-$gluon amplitudes, which are valid for

certain helicity combinations and in leading color approximation, consist of a sum over permutations of Eq. (3.6.1) times some scalar products of momenta in the numerator [Kui91]. We will repeat and expand on some aspects of the explanation of SARGE [HK00b] with an emphasis of the implementation in Fortran. While most notation is in line with Ref. [HK00b], some has been aligned with this thesis or modified for clarity.

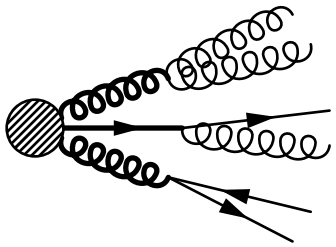


Figure 3.1. APS of a QCD process. The possibly collinear propagators of the antennae are denoted by thick lines and give the dominant divergencies of the matrix element.

3.6.1. Basic Antenna

At first, one has to know how to make a basic antenna which will be the building block of the overall density. We start with two given, arbitrary massless momenta p_1 and p_2 from which a third massless momentum k shall be radiated off. This is motivated by the physical picture of two partons which like to radiate off a third one with a similar momentum. To have the correct infrared and collinear behaviour of the antenna structure, the distribution should be proportional to $[(p_1k)(kp_2)]^{-1}$. By demanding invariance under Lorentz transformations, the basic antenna structure may be defined as

$$\mathrm{d}A(p_1,p_2;k) = \mathrm{d}^4k\,\vartheta(k)\frac{1}{\pi}\frac{(p_1p_2)}{(p_1k)(kp_2)}g\left(\frac{(p_1k)}{(p_1p_2)}\right)g\left(\frac{(kp_2)}{(p_1p_2)}\right) \tag{3.6.2}$$

where $g(\xi)$ serves to regularize singularities as well as to ensure normalization over the whole phase space of k. Therefore, it has to vanish sufficiently fast for both $\xi \to 0$ as well as $\xi \to \infty$. The simplest way to do so, is with a piecewise constant function with one UV cut-off parameter ξ_m whose inverse is used for the infrared:

$$g(\xi) = \mathcal{N}\theta(\xi_m - \xi)\theta(\xi - \xi_m^{-1})\,. \tag{3.6.3}$$

The norm $\mathcal{N}$ as well as the necessity of p_1p_2 in the nominator of Eq. (3.6.2) will quickly become obvious. To find out how to generate k, we evaluate $\int \mathrm{d}A$ by going in the Center of Mass Frame (CMF) of p_1 and p_2 with $L^{-1}(p_1+p_2)$. Then we have

$$p = L^{-1}(p_1+p_2)p_1\ , \quad E = \sqrt{(p_1p_2)/2} \quad \text{and} \quad q = L^{-1}(p_1+p_2)k\ . \tag{3.6.4}$$

Evaluating the first part of Eq. (3.6.2) yields

$$\begin{aligned} \mathrm{d}^4k\vartheta(k) &= \mathrm{d}q^0d^3q\ \delta\left((q^0)^2-\boldsymbol{q}^2\right)\theta\left(q^0\right) \\ &= \frac{q^0}{2}\mathrm{d}q^0\mathrm{d}\cos\theta\ \mathrm{d}\varphi \end{aligned} \tag{3.6.5}$$

Furthermore, we want to set our coordinate system such that $\cos\theta = \boldsymbol{p}\cdot\boldsymbol{q}/(|\boldsymbol{p}|\,|\boldsymbol{q}|) \equiv z$ and $\hat{n}(z,\varphi) = \mathcal{R}_p\frac{\boldsymbol{q}}{|\boldsymbol{q}|}$. Since

$$p_1k = pq = Eq^0(1-z)\ , \quad kp_2 = Eq^0(1+z) \quad \text{and} \quad p_1p_2 = 2E^2\ , \tag{3.6.6}$$

we may write

$$\xi_{1,2} \equiv \frac{p_{1,2}k}{p_1p_2} = \frac{q^0}{2E}(1\mp z) \Leftrightarrow \begin{cases} q^0 &= E(\xi_1+\xi_2) \\ z &= \frac{\xi_2-\xi_1}{\xi_1+\xi_2} \end{cases}\ . \tag{3.6.7}$$

The change from the variables q^0 and z to $\xi_{1,2}$ leads to a Jacobian determinant of p_1p_2/q^0, hence

$$\begin{aligned} (3.6.5) &= \frac{p_1p_2}{2}\mathrm{d}\varphi\mathrm{d}\xi_1\mathrm{d}\xi_2 \quad \text{and} \\ \mathrm{d}A(p_1,p_2;k) &= \frac{\mathrm{d}\varphi}{2\pi}\frac{\mathrm{d}\xi_1 g(\xi_1)}{\xi_1}\frac{\mathrm{d}\xi_2 g(\xi_2)}{\xi_2}\ . \end{aligned} \tag{3.6.8}$$

This sets the normalization condition

$$\int \mathrm{d}\xi\ \frac{1}{\xi}g(\xi) = 1\,, \tag{3.6.9}$$

yielding $\mathcal{N} = (2\log\xi_m)^{-1}$ for Eq. (3.6.3). From Eq. (3.6.8) we can directly read off how to generate the wanted density $\mathrm{d}A$, i.e. draw ϕ uniformly in $[0,2\pi)$ and two ξ's from the density $g(\xi)/\xi$, which follows by integration and inversion, as explained in Section 3.3, yielding:

$$\frac{\theta(\xi_m-\xi)\theta(\xi-\xi_m^{-1})}{2\log\xi_m}\frac{1}{\xi} = \int_0^1 \mathrm{d}x\ \delta\left(\xi - \mathrm{e}^{-\log\xi_m+2\log\xi_m x}\right)\ . \tag{3.6.10}$$

The natural choice for ξ_m which covers the physical phase space, i.e. doesn't cut more than s_0 and keeps the scalar product within the maximally kinematically allowed value $s_m = s - s_0\left(\frac{n(n-1)}{2} - 1\right)$, is

$$\xi_m \equiv \frac{s_m}{s_0} = \frac{s}{s_0} - \frac{(n+1)(n-2)}{2} \,. \tag{3.6.11}$$

As mentioned earlier, Eq. (3.6.3) is just the simplest way to obtain reasonable ξ's and hence there is room for optimization to reduce the number of rejected points which are generated by SARGE, which will be shown in Section 3.8. The basic antenna algorithm can now be formulated as follows

- Given two legs $\{p_1, p_2\}$ in a certain frame, *set* $p = L^{-1}(p_1 + p_2)p_1$ and $E = \sqrt{p_1 p_2/2} = p^0$. Note that this operation is in general not well defined. If p_1 and p_2 are collinear, $m_{p_1+p_2}$ becomes zero and the boost returns unreasonable energies. The resulting momentum from this should not bother us, however, since this set will be rejected due to the cuts anyway.

  ⟨*Set* `p` *and* `E`⟩≡
  ```
  pp = std_boost (-1, p(:,1) + p(:,2), p(:,1))
  E = pp(0)
  ```

- *Draw* two scalar product quotients as random numbers ξ_1 and ξ_2 from $g(\xi)/\xi$ and the azimuthal angle φ of q uniformly in $[0, 2\pi)$.

  ⟨*Draw* `xi` *and* `phi`⟩≡
  ```
  call tao_random_number(ran)
  xi = exp(-lxi + ran(1:2) * 2 * lxi)
  phi = twopi * ran(3)
  ```

- *Construct* q in the CMF of p_1 and p_2 with $z = (\xi_2 - \xi_1)/(\xi_1 + \xi_2)$, absolute value $q^0 = E(\xi_1 + \xi_2)$ and direction $\boldsymbol{q} = q^0 \mathcal{R}_p^{-1} \hat{n}(z, \phi)$. *Boost back as* $k = L(p_1 + p_2)q$ and return it.

  ⟨*Construct* `q` *and boost back as* `k`⟩≡
  ```
  z = (xi(2) - xi(1)) / sum(xi)
  ⟨standard vector n(z,phi)⟩
  pp(1:3) = pp(1:3) / sqrt(dot_product(pp(1:3),pp(1:3)))
  call rotation(-1, pp(1:3), n, q(1:3))
  q(0) = E * sum(xi)
  q(1:3) = q(1:3) * q(0)
  k = std_boost (+1, p(:,1) + p(:,2), q)
  ```

`rotation` is the rotation which takes the `rot_vec` to the 3-axis. This may be achieved by the decomposition $\mathcal{R}_p = B[\phi]A[\theta]$, where A and B are rotations around the 2- and 1-axis. The angles are set by the conditions $(Ap)^1 = 0$ and $(BAp)^2 = 0$, yielding

⟨*Set angles*⟩≡
```
th = atan2(rot_vec(1), rot_vec(3))
s_th = sin(th)
c_th = cos(th)
ph = atan2(rot_vec(2), s_th * rot_vec(1) + c_th * rot_vec(3))
s_ph = sin(ph)
c_ph = cos(ph)
```

The `integer` $s \in \{+1, -1\}$ allows to switch between the rotation and the inverse $R_p^{-1} = A[-\theta]B[-\phi]$. Overall we have

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡
```
subroutine rotation(s, rot_vec, in_vec, res_vec)
  ⟨rotation declarations⟩
  ⟨Set angles⟩
  select case (s)
    case (+1)
      ! Fortran makes columns not rows.
      ! Therefore this looks like a transposition
      AB = reshape([  c_th,  - s_th * s_ph,  s_th * c_ph, &
                      zero,           c_ph,         s_ph, &
                    - s_th,  - c_th * s_ph,  c_th * c_ph], [3, 3])
        res_vec = matmul(AB, in_vec)
    case (-1)
      AB = reshape([         c_th,  zero,         - s_th, &
                   - s_th * s_ph,  c_ph,  - s_ph * c_th, &
                     s_th * c_ph,  s_ph,    c_th * c_ph], [3, 3])
      res_vec = matmul(AB, in_vec)
    case default
      stop 'rotation takes +1 and -1 for s'
  end select
end subroutine rotation
```

The basic antenna is now simply

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡
```
subroutine basic_antenna(p, xi, phi, k)
  ⟨basic antenna declarations⟩
  ⟨Set p and E⟩
  ⟨Construct q and boost back as k⟩
end subroutine basic_antenna
```

The attentive reader has noticed, that we have left out MC part of the algorithm and are giving `xi` and `phi` as variables. This makes the basic antenna a deterministic function which has been cross-checked against the `FORTRAN77` implementation of A. van Hameren et al.. Furthermore, changes in the generation of `xi` don't affect the basic antenna.

3.6.2. Full Antenna

It is straightforward to generate n momenta with the full APS by repeating the above procedure $(n-2)$ times. Let $\mathrm{d}A^i_{j,k} = \mathrm{d}A(q_j, q_k; q_i)$ be the shorthand for the basic antenna in

$$\begin{aligned}\mathrm{d}A^2_{1,n}\mathrm{d}A^3_{2,n}\cdots\mathrm{d}A^{n-1}_{n-2,n} &= \frac{(q_1q_n)}{(q_1q_2)(q_2q_3)\cdots(q_{n-1}q_n)}\frac{g_{n-2}\left(\{q\}\right)}{\pi^{n-2}}\prod_{i=2}^{n-1}\mathrm{d}^4q_i\vartheta(q_i)\\ \text{with}\quad g_{n-2}\left(\{q\}\right) &= g\left(\frac{q_1q_2}{q_1q_n}\right)g\left(\frac{q_2q_n}{q_1q_n}\right)\cdots g\left(\frac{q_{n-2}q_{n-1}}{q_{n-2}q_n}\right)g\left(\frac{q_{n-1}q_n}{q_{n-2}q_n}\right)\ .\end{aligned} \tag{3.6.12}$$

Almost all scalar products in the numerator have cancelled with adjacent basic antenna densities, except q_1q_n. Since this density is by construction invariant under simultaneous Lorentz transformations of all momenta, we can bring them to the lab frame with the same transformations we used for the uniform distribution of RAMBO and maintain the desired density with APS. Before we perform this more formally, we still need the two initial momenta, which we may generate back-to-back and with isotropic three vector orientation. Sadly these cannot serve as initial momenta since they will not be back-to-back after the boost. The corresponding weight can be computed with help of the UAF:

$$w(P) = \int \mathrm{d}^4q_1\vartheta(q_1)\mathrm{d}^4q_n\vartheta(q_n)\,\delta^4(q_1+q_n-P) \tag{3.6.13}$$

By integrating out e.g. q_1 and omitting the other index, we obtain

$$w(P) = \int \mathrm{d}^4q\,\delta\left((q^0)^2-\boldsymbol{q}^2\right)\theta(q^0)\delta\left((q^0-\sqrt{s})^2-\boldsymbol{q}^2\right)\theta(\sqrt{s}-q^0)\ . \tag{3.6.14}$$

The spherical integration is once again trivial and leads to

$$\begin{aligned}w(P) &= 2\pi\int \mathrm{d}q^r\mathrm{d}q^0\,q^r\delta\left(q^0-q^r\right)\delta\left((q^0-\sqrt{s})^2-(q^r)^2\right)\theta(\sqrt{s}-q^0)\\ &= \pi\int \mathrm{d}q^r\,\frac{q^r}{\sqrt{s}}\delta\left(q^r-\frac{\sqrt{s}}{2}\right)\theta(\sqrt{s}-q^r) = \frac{\pi}{2}\ ,\end{aligned} \tag{3.6.15}$$

which is just the same as $V_2(\sqrt{s})$ from Eq. (3.5.7). Obviously one can also use the RAMBO algorithm to generate the first two momenta and obtain the same density. But Eq. (3.6.13) is easier to handle for the subsequent computations and faster in the implementation. Combining the above weight with the RAMBO transformations yields the following density

$$\begin{aligned}\Phi(\{p\}_n) = &\int \mathrm{d}^4q_1\vartheta(q_1)\mathrm{d}^4q_n\vartheta(q_n)\,\delta^4(q_1+q_n-P)\frac{2}{\pi}\,\mathrm{d}A^2_{1,n}\mathrm{d}A^3_{2,n}\cdots\mathrm{d}A^{n-1}_{n-2,n}\\ &\mathrm{d}^4b\,\delta^4\Big(b-q_{(n)}/m_{q_{(n)}}\Big)\mathrm{d}x\,\delta\Big(x-\sqrt{s}/m_{q_{(q)}}\Big)\prod_{i=1}^{n}\delta^4\Big(p_i-xL^{-1}(b)q_i\Big)\\ =&\int\left(\prod_{j=1}^{n}\frac{1}{x^2}\vartheta(p_j)\right)\,\delta^4\left(\frac{1}{x}L(b)(p_1+p_n)-\sqrt{s}e_0\right)\frac{2g_{n-2}(\{q\})}{\pi^{n-1}}\frac{1}{x^{2-2(n-1)}}\\ &\mathrm{APS}\cdot(p_1p_n)^2\,\mathrm{d}^4b\,\delta^4\Big(p_{(n)}-\sqrt{s}e_0\Big)\frac{2s^2}{x}\,\mathrm{d}x\,\delta\Big(1-b^2\Big)\theta\left(b^0\right)\,,\end{aligned} \tag{3.6.16}$$

where we have only used the identities derived in Section 3.5. So far, Eq. (3.6.16) doesn't look quite like the wanted APS due to the additional $(p_1p_n)^2$ in the numerator. However, we can further evaluate to

$$\begin{aligned}\Phi(\{p\}_n) = &\int\Theta_n(\sqrt{s})\,\frac{4g_{n-2}(\{q\})}{\pi^{n-1}x^5}\mathrm{APS}\cdot(p_1p_n)^2\\ &\delta^4\left(b-\frac{L(b)^2(p_1+p_n)}{\sqrt{s}x}\right)\delta\Big(b^2-1\Big)\,\mathrm{d}^4b\,\mathrm{d}x\\ =&\,\Theta_n(\sqrt{s})\frac{1}{(p_1p_2)(p_2p_3)\cdots(p_{n-1}p_n)(p_np_1)}\frac{g_{n-2}(\{q\})s^2}{2\pi^{n-1}}\,.\end{aligned} \tag{3.6.17}$$

Note that the factors of 2 and π are neither consistent with Ref. [HKD00] nor Ref. [HK00b]. The above has been carefully computed and checked to give the same phase space volume as RAMBO. It is of course only valid for $n > 2$ since otherwise no antennae are produced. We should emphasize that Eq. (3.6.16) is a pure combination of properly normed densities and assignments which is why the integral of $\Phi(\{p\}_n)$ over the whole phase space leads to unity and the inverse of Eq. (3.6.17) is the weight associated with a event generated this way. Hereby, $g_{n-2}(\{q\})$ contributes to the weight with $(2\log\xi)^{2n-4}$.

The basic SARGE algorithm is now simply

- *Generate* two massless *momenta* q_1, q_n *back-to-back*

  ⟨*Generate momenta back-to-back*⟩≡

  ```
  call tao_random_number(ran2)
  ```

```
q(0,1) = roots / two
q(0,Nout) = q(0,1)
z = two * ran2(1) - one
phi = twopi * ran2(2)
⟨standard vector n(z,phi)⟩
q(1:3,1) = q(0,1) * n
q(1:3,Nout) = q(0,Nout) * (-n)
```

- Generate $n-2$ momenta q_j with basic antennae $\mathrm{d}A^3_{1,2}\mathrm{d}A^4_{1,3}\cdots\mathrm{d}A^n_{1,n-1}$

  ⟨*Generate basic antennae*⟩≡
  ```
  do j = 1, Nout - 2
    call tao_random_number(ran)
    phi = twopi * ran
    call basic_antenna([q(:,j),q(:,Nout)], xi(:,j), phi, q(:,j+1))
  end do
  ```

- Apply the `RAMBO` transformations, i.e. compute $q_{(n)} = \sum q_j$ and the boost and scaling that brings it to $\sqrt{s}\,e_0$ and apply it to all q_j, resulting in p_j.

Framing this as `subroutine` simply gives

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡
```
subroutine full_antenna(roots, xi, p)
  ⟨Variables of full_antenna⟩
  ⟨Generate momenta back-to-back⟩
  if (Nout > 2) then
    ⟨Generate basic antennae⟩
    ⟨Determine Lorentz boost⟩
    ⟨Boost all⟩
  end if
  p = q
end subroutine full_antenna
```

3.6.3. Permutations

As we have mentioned in the beginning, we do not only need the APS with successive connections but also all permutations thereof. Since we do not include the incoming

momenta in the antenna densities for now, we can achieve the permutations of the momenta with a simple relabeling. This means our density becomes

$$g^{\mathrm{QCD}}(\{p\}_n) = \frac{1}{n!} \sum_{\sigma \in S_n} \Phi(\{p\}_{\sigma(n)}) \,, \tag{3.6.18}$$

where S_n is the symmetric group, i.e. the group of permutations of n objects, which has $n!$ elements. We can represent a permutation σ with an array of size n, where the ith entry j indicates to replace i with j, or with a simple enumeration of the group elements, $i \in \{1, ..n!\}$.

A random permutation of t objects, is a reordering or shuffling of the objects, whereby each out of $t!$ permutations has to occur with equal probability. The algorithm from Ref. [Knu81] for this is very straightforward and reads

⟨*Implementation of* `shuffle`⟩≡
```
do l0 = size(arr), 2, - 1
  call tao_random_number(ran)
  k0 = int(ran * l0) + 1
  ⟨swap arr(k0) and arr(l0)⟩
end do
```

The above chunk is actually polymorphic, i.e. can be used for arbitrary types of `arr` and `tmp`. On the other hand we might also need all permutations for the weight. `next_permute` gives the next permutation in lexicographical ordering [Knu11], meaning that by starting with $[1, 2, \ldots, n]$ successive application of the function gives all permutations, whereby the last is $[n, \ldots, 2, 1]$.

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡
```
pure function next_permute(iarr) result (arr)
  ⟨Variables of next_permute⟩
  arr = iarr
  ⟨find last element arr(k0) smaller than its right neighbor⟩
  ⟨terminate if none found⟩
  ⟨find last element arr(l0) larger than arr(k0)⟩
  ⟨swap arr(k0) and arr(l0)⟩
  ⟨reverse arr after k0⟩
end function next_permute
```

As termination condition, i.e. when no more permutations are possible, `next_permute = 0` has been set. While this is quite efficient, it is worth to save the permutations to a table `table_perms(n,nfac)` as long it is feasible, which also allows to draw a random permutation by drawing a random index $i \in n!$ instead of using `shuffle`. Feasible of

course depends on the environment. For a usual desktop user the upper limit for using a table is probably $n = 11$, leading to 418 MB, while for $n = 12$ we reach 5481 MB, where we have already used only 1 byte integers which can represent numbers up to $2^7 - 1 = 127$. Note that for gluon amplitudes the information associated with the color flows will blow up way before this and we will only have to resort to `shuffle`, when the complete computation is performed without tables for color flows.

Now, we have everything together for the basic SARGE algorithm

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
subroutine sarge0(roots, lxi, Nin, p, weight, perm)
  ⟨Variables of sarge0⟩
  ⟨Initialization of perms and Nout⟩
  ⟨Draw a perm from table_perms⟩
  ⟨Draw xis from the hypercube⟩
  call full_antenna(roots, xi, p)
  weight = sarge0_weight(Nin, perm, roots, lxi)
end subroutine sarge0
```

To wrap everything up, we have the module

⟨`ovm_phasespace.f90`⟩≡

```
#include "macros.h"
module ovm_phasespace
  use kinds
  use ovm_load
  implicit none
  private
  ⟨Declaration of ovm_phasespace procedures⟩
  ⟨Interfaces of ovm_phasespace procedures⟩
  ⟨Variables of ovm_phasespace⟩
contains
  ⟨Implementation of ovm_phasespace procedures⟩
end module ovm_phasespace
```

Note that we use `macros` only to chose between debugging modes as assertive programming with `#if DEBUG` is more reusable than repetitive, brute-force `printf` statements that have to be removed afterwards. If we set `DEBUG` to `0`, we can herewith regain full speed while `1` switches on tests of all implemented assertions.

3.7. Discussion and Performance

At first, we will verify that the generated phase space density is indeed the one of Eq. (3.6.17). For this we sample a fixed number of points and compare the generated probability density for $(p_1 p_2) \cdots (p_n p_1)$ of RAMBO and SARGE. Furthermore, we can check that we obtain an unbiased sampling of the phase space if we attach the corresponding weight to the counts. The results for different cuts and three generated momenta are shown in Fig. 3.2. It has also been checked that the bounds for the smallest and largest value for APS^{-1} are $(s_0/2)^n$ and $(s/(n(n-1)))^n$, respectively, which follow from the imposed cuts and overall available energy. Similar pictures arise for four, five and six momenta.

After the dry tests, we can apply SARGE now to the problem of interest and include the matrix element. The goal is of course to reduce the variance of the integration via importance sampling as in Eq. (3.3.3), which is why the weight function should be kept as close to the integrand as possible. Since we know that, at least for gluon amplitudes, all permutations of the APS occur in the matrix element, the weight function should also consist of all permutations. There are, however, two ways to achieve this. Either we compute the concrete weight of a certain random permutation and multiply it with the integrand or we compute in every point the weight as sum over all permutations. In the statistical mean both methods are equivalent while the former introduces additional variance and the latter additional computations. We observe that the effort for these computations are in fact negligible compared to the gains. Note first, that the average time spend for the complete phase space generation, including helicity, non-accepted events and computation of weights, depends strongly on the cuts and is about the same for $\tau = 0.1$ as and roughly an order of magnitude smaller for $\tau = 0.01$ than the time for the evaluation of the matrix elements, whereby we have sampled helicities and summed over all colors. From this fraction the amount needed for the sum of inverse scalar products is hard to measure at all, since the scalar products have been already computed to check for the cuts. On the other hand, using the full weight can reduce the number of needed accepted events to achieve a certain error in the integration by factors of about ~ 8 for fully connected final states like $q\bar{q} \to 4g$. As long as all color flows are computed, we can therefore highly recommend the use of the full weight. In Fig. 3.3 we show the drastically improved convergence behaviour of the basic SARGE algorithm compared to RAMBO. We can see a clean $1/\sqrt{N_{\mathrm{acc}}}$ convergence without the bumps that occur when a flat phase space generator like RAMBO comes close to the singularities.

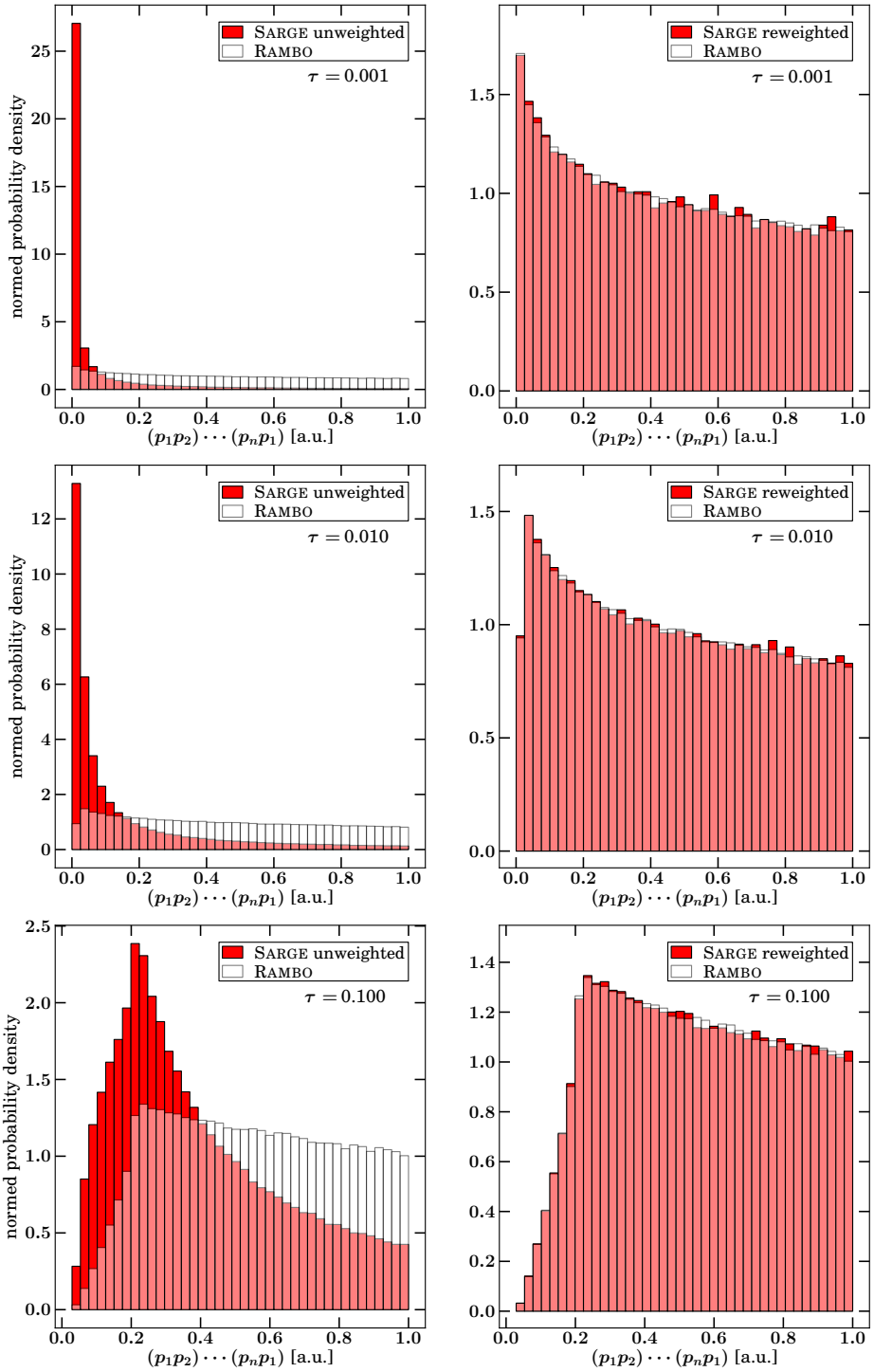


Figure 3.2. The generated phase space density of RAMBO and SARGE for various values of the cut-off parameter τ. As desired, we are preferably sampling points with small scalar products. The histograms on the right side result from the same points but here the counts of SARGE have been weighted according to Eq. (3.6.17). Note that SARGE only appears to be stronger fluctuating, especially in the top right, after reweighting due to the resulting lower counts in the tail of the distribution.

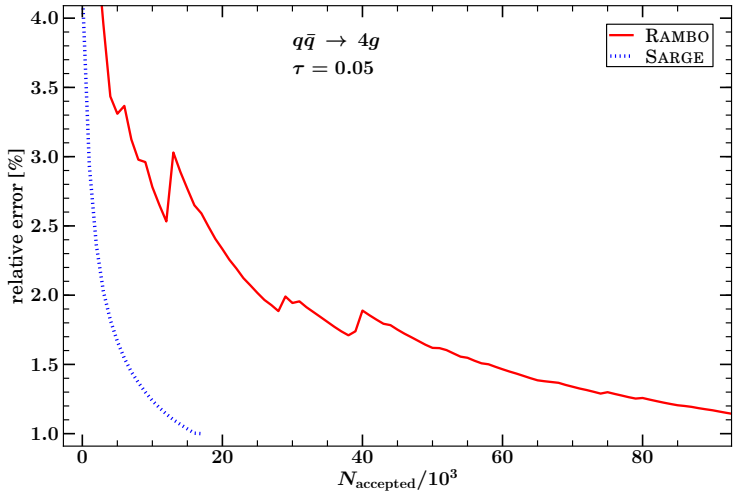


Figure 3.3. The relative error of both momentum generators as function of the accepted momenta, which is a good measure for the overall time as discussed in the text. The convergence of SARGE compared to RAMBO is obviously way smoother and has a smaller variance due to the importance sampling leading to earlier termination. RAMBO shows peaks when it hits a singularity.

We notice that the phase space density Eq. (3.6.17) is invariant under cyclic permutations. This means that the number of distinct channels is not $n!$. In fact if we say that two permutations are equivalent when one can be written as cyclic permutation of the other, the group structure of S_n reduces to the factor group S_n/Z_n, whereby Z_n is the cyclic group with n elements. This implies that the true number of distinct channels is the order of the factor group which is $|S_n| \,/\, |Z_n| = (n-1)!$, analogously to the number of distinct color flows in an n gluon amplitude. Furthermore, we see that the different channels have substantial overlap, which is why even the use of merely one channel can still significantly reduce the variance. Overall, SARGE represents an excellent mapping of QCD and QCD associated processes that allows to integrate even small cuts and high multiplicities in a reasonable amount of time. Hereby, the extensions of the next section are quite essential to fully map all divergencies and maintain high acceptance rates, which is why a more detailed analysis with different processes will follow at the end of this chapter.

3.8. SARGE Extensions

The above has shown the beauty and simplicity of the SARGE algorithm, which solves the problem of generating n massless momenta with the desired phase space density.

Now, we will extend this idea and increase thereby the efficiency. Section 3.8.1 will improve the performance by reducing the number of rejected points while not changing the accepted. The second one in Section 3.8.2 will improve the convergence since the accepted points will now also include the APS associated with incoming momenta.

3.8.1. Reducing Rejected Points

To reduce the number of rejected points, we should discuss if the implemented realization of $g(\xi)$ is really efficient enough. One quickly sees that we are restricting only the ratios of scalar products which are involved in the generation of the QCD antennae, i.e.

$$\frac{p^{i-1}p^i}{p^{i-1}p^n} \quad \text{and} \quad \frac{p^ip^n}{p^{i-1}p^n} \,. \tag{3.8.1}$$

It makes, however, more sense to demand this for *all ratios* that can be formed in the generation of the antennae

$$\xi_m^{-1} < \frac{p^ip^j}{p^kp^l} < \xi_m \,. \tag{3.8.2}$$

By going to variables $x_{yz}^{ij} = \log \frac{p^ip^j}{p^yp^z}$, we can create all scalar products via differences $x_{kl}^{ij} = x_{yz}^{ij} - x_{yz}^{kl}$, whereby we have to demand that

$$\left|x_{yz}^{ij}\right| < 1 \quad \text{and} \quad \left|x_{yz}^{ij} - x_{yz}^{kl}\right| < 1 \,. \tag{3.8.3}$$

By dropping the fix point $_{yz}$ and relabelling (ij) as they occur in the generation of the antennae $i \in \{1, \ldots, 2n-4\}$, we can replace the assignment of Eq. (3.6.10), $\xi = \xi_m^{-1+2x}$, by

$$\xi = \mathrm{e}^{\log \xi_m (x_i - x_j)} \,. \tag{3.8.4}$$

This notation allows to realize that the restrictions of Eq. (3.8.3), define an $m = 2n-4$ dimensional polytope P in m dimensional space whereby the different x's are now different coordinates. In fact, if we use Eq. (3.8.4) with the mere restriction that $\boldsymbol{x}$ are from the hypercube, $\boldsymbol{x} \in [-1,1]^m$, we reobtain the original sampling. The efficiency of the original sampling compared to one which respects the mutually constraints of Eq. (3.8.3) is therefore given by the ratio of the volume of the defined polytope and the hypercube $V_m(P)/2^m$. For better imagination and illustration, we show the polytope in two and three dimensions in Fig. 3.4. We can immediately conjecture

that the ratio of volumes will decrease with increasing dimension, which will be shown below. Furthermore, we see that the mutual condition is only relevant if x_i and x_j have opposite signs.

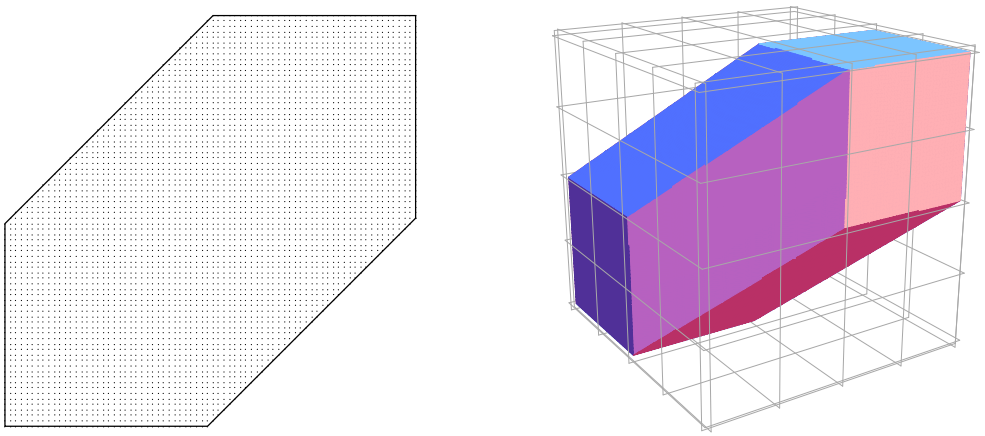


Figure 3.4. The convex hull of the polytope defined by $|x_k| < 1$ and $|x_k - x_l| < 1$ $\forall k, l \in \{1, \ldots, m\}$ for $m = 2, 3$.

In Ref. [HK00a], an efficient algorithm for generating $\boldsymbol{x}$ inside the polytope with unit efficiency is given for arbitrary dimension. To obtain it, one simply integrates the volume of the polytope analytically and replaces the integration with a sampling. We want to compute

$$V_m(P) = \int_{-1}^{1} \mathrm{d}x_1 \cdots \mathrm{d}x_m \prod_{\substack{i,j=1 \\ i \neq j}}^{m} \theta\Big(1 - |x_i - x_j|\Big) \,. \tag{3.8.5}$$

At first, one should divide the integration variables in $m - k$ positive and k negative ones and set $y_k = -x_k$, allowing to write $m(m-1)/2$ constraints as one. The number of possibilities to have k negative variables in m is given by the binomial such that we have by relabeling

$$\begin{aligned} V_m(P) &= \sum_{k=0}^{m} \binom{m}{k} V_{k,m} \\ V_{k,m} &= \int_0^1 \mathrm{d}y_1 \cdots \mathrm{d}y_k \mathrm{d}x_{k+1} \cdots \mathrm{d}x_m \, \theta\left(1 - \max_i x_i - \max_j y_j\right) \,. \end{aligned} \tag{3.8.6}$$

We can furthermore say that one of the y's and x's is largest, which we call y_1 and x_m, respectively. The number of possibilities for this is $\binom{k}{1} = k$ and $m - k$ and the integration over the other variables each simply yields the upper bound which is y_1

and x_m. Overall, we have

$$\begin{aligned} V_{m,k}(P) &= k(m-k)\int_0^1 \mathrm{d}y_1\, y_1^{k-1}\int_0^{1-y_1}\mathrm{d}x_m\, x_m^{m-k-1} \\ &= k\int_0^1 \mathrm{d}y_1 y_1^{k-1}(1-y_1)^{m-k} \;= \frac{k!(m-k)!}{m!}\,, \end{aligned} \tag{3.8.7}$$

which is just the inverse of the binomial, and hence

$$V_m(P) = m+1\,. \tag{3.8.8}$$

Since we have now the volume of the polytope, we can compute how bad our efficiency used to be, which is shown in Tab. 3.1. Furthermore, we can translate the calculation to an algorithm via the UAF:

Table 3.1. The efficiency of drawing within the hypercube compared to the polytope, $V_m(P)/2^m$, for $m = 2n-4$ and n generated momenta.

n	3	4	5	6	7	8	9	10
efficiency [%]	75.0	31.3	10.9	3.52	1.07	0.317	0.0916	0.0259

- Draw k in $[0,m]$. If $k=0$, simply assign $x_i = \mathtt{x}\ \forall i \in \{1,\ldots,n\}$, where $\mathtt{x}$ is a random number in $[0,1)$, redrawn for every i. Analogously, if $k=m$, set $x_i = -\mathtt{x}$.

- If $0<k<m$, generate y_1 with density $y_1^{k-1}(1-y_1)^{m-k}$ between 0 and 1 and x_m with density x_m^{m-k-1} between 0 and $1-y_1$. The corresponding mappings are [HK00a]

$$v_1 = -\log\left\{\prod_{i=1}^{k} x\right\}, \qquad v_2 = -\log\left\{\prod_{j=1}^{m-k+1} x\right\}, \qquad y_1 = \frac{v_1}{v_1+v_2}$$
$$\text{and} \quad x_m = (1-y_1)x^{1/(m-k)} \tag{3.8.9}$$

- Finally, draw the remaining x's and y's within their bounds and perform a random permutation of the whole set to get rid of the ordering, which we have introduced for the calculation,

$$\begin{aligned} x_1 &= -y_1\,, \qquad x_i = -y_1 x\,, \qquad i = 2,\ldots,k \\ & \qquad\quad x_i = \ x_m x\,, \qquad i = k+1,\ldots,m-1\,. \end{aligned} \tag{3.8.10}$$

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
subroutine polytope(x)
  ⟨Variables of polytope⟩
  ⟨Create permutation labels⟩
  ⟨Draw k and all x if k is 0 or m⟩
  ⟨Generate y1 with density y_1^{k-1}(1-y_1)^{m-k}⟩
  ⟨Draw remaining x within their bounds⟩
end subroutine polytope
```

Of course, we have to account for the reduced volume also in the density, which becomes

$$g_{n-2}^{P}(\{q\}) = \frac{2^{2n-4}}{2n-3} g_{n-2}(\{q\}) \prod_{\substack{i,j=1 \\ i\neq j}}^{2n-4} \theta\Big(1 - |x_i - x_j|\Big) \,. \tag{3.8.11}$$

The only thing that changes in the SARGE algorithm apart from the weight is the generation of the `xi`'s

⟨*Draw* `xi`*s from the polytope*⟩≡

```
call polytope(x(1:))
x(0) = zero
do i = 2, Nout-1
  xi(1,i-1) = exp((x(2*i-3)-x(2*i-4))*lxi)
  xi(2,i-1) = exp((x(2*i-2)-x(2*i-4))*lxi)
end do
```

We have verified empirically that the increase in efficiency according to Tab. 3.1 is indeed reflected by the same amount in the number of accepted momenta.

3.8.2. Including Incoming Momenta

For the incoming momenta p_0 and $\tilde{p}_0$, we cannot use the basic antennae since they have to be back-to-back and should lie fixed on an axis. The latter problem can be easily solved by rotating the whole system of generated momenta for each phase space point. So we basically just need to generate two momenta back-to-back with the desired density $\propto [(p_0 p_1)(p_n \tilde{p}_0)]^{-1}$ which completes the full APS. It turns out that in order to avoid singularities in the integration, it is useful to introduce a small $\delta > 0$ such that the scalar product of massless momenta reads [HK00b]

$$(pq)_\delta = p^0 q^0 (1 + \delta - \hat{p}\hat{q}) \,. \tag{3.8.12}$$

Obviously, this breaks Lorentz invariance. However, we will generate the incoming momenta after the remaining momenta have been brought to speed, i.e. boosted and scaled such that their sum is equal to $\sqrt{s}e_0$. This means that no further Lorentz boosts will be involved and the only invariance which is still important is rotation invariance, which is given by Eq. (3.8.12). From the point of view of importance sampling, the density

$$\varphi(p_1,p_2) = \mathcal{N}(p_1,p_2)\frac{\vartheta(k)\delta(k^0-1)}{(kp_1)_\delta(p_2\tilde{k})_\delta}\,, \quad \text{with} \quad \int \mathrm{d}^4k\,\varphi(p_1,p_2) = 1\,, \tag{3.8.13}$$

will be a very good approximation of the desired density as long as δ is small enough. The given p_1 and p_2 are arbitrary momenta of the full antenna and k^0 is simply for convenience set to one as scaling it leads to an overall factor that can be absorbed in the norm $\mathcal{N}$. To see how we can generate this, we start to evaluate

$$\begin{aligned}\frac{1}{\mathcal{N}(p_1,p_2)}\int \mathrm{d}^4k\,\varphi(p_1,p_2) &= \int \mathrm{d}^3k\,\frac{\delta(k^2)}{(kp_1)_\delta(p_2\tilde{k})_\delta}\\ &= \int \mathrm{d}\Omega\,\frac{1}{2}\left[p_1^0(1-\hat{\boldsymbol{p}}_1\hat{\boldsymbol{k}}+\delta)p_2^0(1+\hat{\boldsymbol{p}}_2\hat{\boldsymbol{k}}+\delta)\right]^{-1}\,. \end{aligned}\tag{3.8.14}$$

To avoid the dependence of the integrand on two angles, we can transform the product into a sum with the Feynman trick, $1/(A\cdot B) = \int_0^1 \mathrm{d}x\,[xA+(1-x)B]^{-2}$, yielding

$$\begin{aligned}(3.8.14) &= \frac{1}{2p_1^0p_2^0}\int \mathrm{d}\Omega\int_0^1 \mathrm{d}x\,\left[1+\delta-\boldsymbol{p}_x\hat{\boldsymbol{k}}\right]^{-2} \quad \text{where} \quad \boldsymbol{p}_x = x\hat{\boldsymbol{p}}_1+(x-1)\hat{\boldsymbol{p}}_2\\ &= \frac{\pi}{p_1^0p_2^0}\int_0^1 \mathrm{d}x\int_{-1}^1 \mathrm{d}z\,\left[1+\delta-z\,|\boldsymbol{p}_x|\right]^{-2} \quad \text{setting} \quad z=\cos\left(\angle(\boldsymbol{p}_x,\hat{\boldsymbol{k}})\right)\,. \end{aligned}\tag{3.8.15}$$

This is a good place to pause and think how δ is related to the overall cut-off τ. In fact, so far δ was not necessary at all and also this integral is finite if we take the cuts into account. Obviously, the divergent region is around $z=1$ and $|\boldsymbol{p}_x|=1$. $z=1$ means that $\hat{k}$ is close to the superposition of $\hat{\boldsymbol{p}}_1$ and $-\hat{\boldsymbol{p}}_2$ while $|\boldsymbol{p}_x|=1$ indicates that $\boldsymbol{p}_x=\hat{\boldsymbol{p}}_1$ or $\boldsymbol{p}_x=-\hat{\boldsymbol{p}}_2$ or that $(\hat{\boldsymbol{p}}_1\parallel-\hat{\boldsymbol{p}}_2)$. This is, however, bounded since

$$(p_i+k)^2 > s_0 \quad\Leftrightarrow\quad p_i^0\sqrt{s}(1-\hat{\boldsymbol{p}}_i\hat{\boldsymbol{k}}) > s_0 \tag{3.8.16}$$

$$\Rightarrow \boldsymbol{p}_x\hat{\boldsymbol{k}} < 1-\frac{s_0}{\sqrt{s}}\left(\frac{x}{p_1^0}+\frac{1-x}{p_2^0}\right) \equiv z_m(x)\,. \tag{3.8.17}$$

Also the energies are bounded by Eq. (3.8.16), $p^0_{i,\min} = s_0/(2\sqrt{s})$, and by the available energy, $p^0_{i,\max}=\sqrt{s}$, yielding $1/2 < z_m < 1-s_0/s$. Note, that these are only estimates

and z_m is actually further constraint[2]. In principle, we can use $z_m(x)$ as bound for the z integration, leading to

$$\int_{-z_m}^{z_m} \mathrm{d}z \, \frac{1}{(1 - z\,|\boldsymbol{p}_x|)^2} = \frac{2z_m}{1 - z_m^2 \boldsymbol{p}_x^2} \,. \tag{3.8.18}$$

But this doesn't help any further since the ensuing integration over x gives no handable expression, which is needed to quickly compute the weight. We therefore acknowledge the gain in changing the lower boundary a bit such that

$$\int_{-1-\delta/|\boldsymbol{p}_x|}^{1-\delta/|\boldsymbol{p}_x|} \mathrm{d}z \, \frac{1}{(1 - z\,|\boldsymbol{p}_x|)^2} = \int_{-1}^{1} \mathrm{d}z \, \frac{1}{(1 + \delta - z\,|\boldsymbol{p}_x|)^2} = \frac{2}{(1+\delta)^2 - \boldsymbol{p}_x^2} \,, \tag{3.8.19}$$

which only depends on x over $\boldsymbol{p}_x$ and will give a closed expression later on. While we are not too concerned about the extension of the lower boundary, we might still learn something about the relationship between δ and τ by examining the singularity. More specifically, we can ask how to choose δ such that we are integrating exactly to the boundary given by the cut, i.e. $z_m = 1 - \delta/\,|\boldsymbol{p}_x|$ or equivalently

$$\begin{aligned} \frac{s_0}{\sqrt{s}} \left(\frac{x}{p_1^0} + \frac{1-x}{p_2^0} \right) &= \frac{\delta}{\sqrt{1 - 2(1 + \hat{\boldsymbol{p}}_1 \hat{\boldsymbol{p}}_2) x(1-x)}} \\ \Leftrightarrow \frac{\delta}{\tau} = \frac{2}{n(n-1)} \left(\frac{x}{p_1^0/\sqrt{s}} + \frac{1-x}{p_2^0/\sqrt{s}} \right) &\sqrt{1 - 2(1 + \hat{\boldsymbol{p}}_1 \hat{\boldsymbol{p}}_2) x(1-x)} \,. \end{aligned} \tag{3.8.20}$$

Obviously, δ depends non-trivially on τ and the combination of p_1 and p_2. At the edge values, $x = 1$ and $x = 0$, we have, by defining a mean energy as $\langle\sqrt{s}\rangle = \sqrt{s}/n$,

$$\delta = \tau \frac{2}{n-1} \, \frac{\langle\sqrt{s}\rangle}{p_{1,2}^0} \equiv \delta_{1,2} \,, \tag{3.8.21}$$

whereas in the middle, for $x = 1/2$,

$$\delta = \delta_1 \left(\frac{1 + p_1^0/p_2^0}{2} \right) \sqrt{1 - \frac{1}{2}\Big(1 + \hat{\boldsymbol{p}}_1 \hat{\boldsymbol{p}}_2\Big)} \,. \tag{3.8.22}$$

Examining Eqs. (3.8.20) to (3.8.22), we find that if p_1 and p_2 have roughly the mean energy $p_{1,2}^0 = \langle\sqrt{s}\rangle$ then $\delta = 2\tau/(n-1)$ integrates the singularity up to the cut, as long as $\hat{\boldsymbol{p}}_1 \hat{\boldsymbol{p}}_2$ is not too small as it can drive the square root from 1 to 0. Of course, also this scalar product is cut meaning that δ can only become zero if τ is zero. This fits our expectation that for large τ, δ can be large as well while still representing the density in the corresponding phase space. Since we can compute the exact dependence of the density on δ, not the final result but the variance will depend

[2] E.g. we assumed for $p_{i,\min}^0$ that $\hat{\boldsymbol{p}}_i \hat{\boldsymbol{k}} = -1$ which would conflict with the cut between p_i and $\tilde{k}$.

on it and we can use $\delta = 2\tau/(n-1)$ as a very good rule of thumb for a reasonable density which is close to the integrand.

As we have explained the reasoning behind δ and how it is related to the bounded phase space, we resume now with the combination of Eqs. (3.8.15) and (3.8.19)

$$\begin{aligned}\frac{1}{\mathcal{N}(p_1,p_2)}\int \mathrm{d}^4k\,\varphi(p_1,p_2) &= 2\pi\int_0^1 \mathrm{d}x\,\left[p_1^0p_2^0(2\delta+\delta^2)-2p_1\tilde{p}_2(x^2-x)\right]^{-1}\\ &= -\frac{\pi}{p_1\tilde{p}_2}\int \mathrm{d}x\,\frac{1}{(x-x_+)(x-x_-)}\,,\end{aligned}$$

where $x_\pm$ are solutions of $(1+\delta)^2 = \boldsymbol{p}_x^2 \Leftrightarrow x_\pm = \frac{1}{2}\pm\sqrt{\frac{1}{4}+\frac{p_1^0p_2^0(2\delta+\delta^2)}{2p_1\tilde{p}_2}}$,

$$= \frac{2\pi}{p_1\tilde{p}_2}\frac{\log\left|\frac{x_+}{x_-}\right|}{(x_+-x_-)} = \frac{1}{\mathcal{N}(p_1,p_2)}\,. \tag{3.8.23}$$

Finally, we should rotate the system to have fixed incoming momenta. As noted above, if we use $k^0 = \sqrt{s}/2$, the norm scales as $\mathcal{N} \to (\sqrt{s}/2)\mathcal{N}$. Using momenta i, j of the full antenna $\Phi(\{p\}_n)$ for the generation, we obtain a density

$$\begin{aligned}D(\{p\}_n,i,j) &= \int \mathrm{d}^{4n}q\,\mathrm{d}^4k\,\Phi(\{q\}_n)\,\varphi(q_i,q_j)\prod_{i=1}^n\delta^4(p_i-\mathcal{R}_kq_i) \qquad (3.8.24)\\ &= \Phi(\{p\}_n)\,\frac{\sqrt{s}}{2}\mathcal{N}(p_i,p_j)\int \mathrm{d}^4k\,\vartheta(k)\delta(k^0-1)\frac{1}{(p_i\mathcal{R}_kk)_\delta(p_j\mathcal{R}_k\tilde{k})_\delta}\,,\end{aligned}$$

where we have used that rotations are orthogonal transformation, which neither change the volume element nor $\Phi(\{q\}_n)$. The last line can now be evaluated analogously to the computation of the norm but with a trivial z integration since $\mathcal{R}_kk = k^0(e_0+e_3)$ and thusly

$$D(\{p\}_n,i,j) = \Phi(\{p\}_n)\,\frac{1}{(p_ip_0)_\delta(p_j\tilde{p}_0)_\delta}\frac{p_i\tilde{p}_j(x_+-x_-)}{\log\left|\frac{x_+}{x_-}\right|\left(\frac{2}{\sqrt{s}}\right)^2}\,. \tag{3.8.25}$$

The scaling factor of $\sqrt{s}/2$ has been neglected in Ref. [HK00b] but is important to obtain the correct weight. The above calculation reads as an algorithm

- Given a pair $\{q_i,q_j\}$, compute $x_\pm$

  ⟨*Compute* `x_pm`⟩≡

```
qiqjt = dot(q(:,i),[q(0,j),-q(1:3,j)])
rt = sqrt(one/four + q(0,i)*q(0,j)*(two*delta+delta**2) / (two*qiqjt))
```

```
x_p = one / two + rt
x_m = one / two - rt
```

- Generate $x \in [0,1]$ according to a density

$$-\frac{1}{(x-x_+)(x-x_-)} = \int_{z_0}^{z_1} \mathrm{d}z\, \delta\left(x - \frac{x_+ \mathrm{e}^{x_+ z} - x_- \mathrm{e}^{x_- z}}{\mathrm{e}^{x_+ z} - \mathrm{e}^{x_- z}}\right) , \tag{3.8.26}$$

where $x(z_1) = 1,\ x(z_0) = 0$ and $z = (\log\{x - x_+\} - \log\{x - x_-\})/(x_- - x_+)$. Note that since this is a unnormalized density, $z_1 - z_0 = 2\log\left|\frac{x_+}{x_-}\right|/(x_+ - x_-) \neq 1$. With x, set $\boldsymbol{p}_x = x\hat{\boldsymbol{p}}_i + (x-1)\hat{\boldsymbol{p}}_j$. To maintain the signs, we need complex z's but in the end all imaginary parts are zero.

⟨*Generate* `x` *and set* `p_x`⟩≡
```
call tao_random_number(ran)
z0 = log(complex(x_p / x_m, zero))
z1 = log(complex((x_p-one) / (x_m-one), zero))
yy = real(exp(z0 + (z1 - z0) * ran))
x = (x_m * yy - x_p) / (yy - one)
p_x = x * q(1:3,i) / sqrt(dot_product(q(1:3,i), q(1:3,i)))
p_x = p_x + (x-one) * q(1:3,j) / sqrt(dot_product(q(1:3,j), q(1:3,j)))
```

- Generate $\varphi \in [0, 2\pi)$ uniformly and $z \in [-1,1]$ with a density

$$\frac{1}{(1+\delta-|\boldsymbol{p}_x|\,z)^2} = \int_{s_0}^{s_1} \mathrm{d}y\, \delta\left(z - \left(-\frac{1}{s\boldsymbol{p}_x^2} + \frac{1+\delta}{|\boldsymbol{p}_x|}\right)\right) , \tag{3.8.27}$$

with the boundary values $s_{1,0} = 1/(|\boldsymbol{p}_x|\,(1+\delta \mp |\boldsymbol{p}_x|))$.

⟨*Generate* `phi` *and* `z`⟩≡
```
call tao_random_number(ran)
phi = twopi * ran
abs_px = sqrt(dot_product(p_x,p_x))
call tao_random_number(ran)
s0 = one / (one+delta+abs_px)
s1 = one / (one+delta-abs_px)
s = s0 + (s1-s0) * ran
z = (one + delta - one / s) / abs_px
```

- Compute $\boldsymbol{k} = \mathcal{R}_{p_x}^{-1}\hat{n}(z,\varphi)$, rotate all momenta with $\mathcal{R}_k$ and return them

⟨*Compute* `k` *and rotate all*⟩≡
```
⟨standard vector n(z,phi)⟩
p_x = p_x / abs_px
```

```
      call rotation(-1, p_x, n, k)
      do l=1, size(q, dim=2)
        call rotation(+1, k, q(1:3,l), pp(1:3,l))
        pp(0,l) = q(0,l)
      end do
```

⟨*Implementation of* ovm_phasespace *procedures*⟩+≡

```
  subroutine include_incoming(q, i, j, delta, pp, weight, wght_only)
    ⟨Variables of include_incoming⟩
    ⟨Compute x_pm⟩
    ⟨Generate x and set p_x⟩
    ⟨Generate phi and z⟩
    if (.not. present(wght_only)) then
      ⟨Compute k and rotate all⟩
    end if
    weight = real(-z1+z0) / (two*(x_p - x_m)*(qiqjt))
  end subroutine include_incoming
```

The question remains where to insert this density. While earlier we could just permute the whole set and relabel the momenta, now we have to take care of the permutations involving incoming momenta ourselves. Note that $D(\{p\}_n, i, j) = D(\{p\}_n, j, i)$ if we substitute $\boldsymbol{k} \to -\boldsymbol{k}$ in the integration. Despite that, it turns out to be highly advantageous to explicitely symmetrize the density with respect to i, j, i.e. using $\frac{1}{2}D(i,j) + \frac{1}{2}D(j,i)$, which can be achieved by randomly chosing between swapping i and j or leaving them. This results in errors which are more than a magnitude smaller. We can understand this by comparing our density, e.g. with the formula of Parke and Taylor for the tree-level Maximally Helicity Violating (MHV) amplitudes in n-gluon scattering [PT86]

$$\begin{aligned}\left|\mathcal{M}(+,\ldots,+,i^-,+,\ldots,+,j^-,+,\ldots,+)\right|^2 &= c(p_i p_j)^4 \\ \sum_{\sigma\in S_n}\Big[(p_{\sigma(1)}p_{\sigma(2)})\cdots(p_{\sigma(n-1)}p_{\sigma(n)})(p_{\sigma(n)}p_{\sigma(1)})\Big]^{-1}&,\end{aligned} \tag{3.8.28}$$

where c is some constant and subleading terms in color are neglected. Due to the sum over all permutations, the matrix element is symmetrized between any exchanges of momenta and so should our density. The comparably poor performance, if this is not applied, results thusly from missing divergencies in the weight function compared to the matrix element that can be catched by using both combinations of i, j.

With this in mind, the remaining permutations of the set reduce to two cases. Either p_0 and $\tilde{p}_0$ are next to each other, i.e. connected with an implied $(p_0\tilde{p}_0) = s/2$ that we

don't have to sample since it never becomes singular,

$$-1-2-\cdots-i-0-\tilde{0}-(i+1)-\cdots-n- \tag{3.8.29}$$

or they are not and we need two insertion points

$$-1-2-\cdots-i-0-(i+1)-\cdots-j-\tilde{0}-(j+1)-\cdots-n- \ . \tag{3.8.30}$$

The problem is that we have to break up the earlier connection between $i-(i+1)$ and $j-(j+1)$, meaning we need p_ip_{i+1}'s in the nominator, if we want an exact APS with $n+2$ scalar products in the denominator as the full gluon integrand has it. This problem can be solved statistically, i.e. by choosing (i,j) proportional to $(p_ip_{i+1})(p_jp_{j+1})$ with the downside that we can't just choose a certain channel, according to a certain color flow or diagram, out of $(n+1)!$ which corresponds to a permutation of $n+2$ neglecting cyclic permutations. Furthermore, in the split case we are only creating connections of type $i-0$ *or* $0-(i+1)$ though we actually need both, another point which can be solved in the statistical mean. After stating the problems, we now present the solutions, implying $(i \mod n)$ for each index i [HK00b]:

One insertion: Draw i according to the density

$$\frac{p_ip_{i+1}}{\Sigma_1(\{p\}_n)} \quad \text{with} \quad \Sigma_1(\{p\}_n)=\sum_{i=1}^{n}(p_ip_{i+1}) \tag{3.8.31}$$

and set $j=i+1$. This cancels as desired the connection between i and $i+1$ and replaces it with a much softer factor Σ_1. The total density is then

$$B_1(\{p\}_n)=\frac{1}{\Sigma_1(\{p\}_n)}\sum_{i=1}^{n}(p_ip_{i+1})D(\{p\}_n,i,i+1)\ . \tag{3.8.32}$$

Two insertions: Draw two points i and j according to

$$\frac{(p_ip_{i+1})(p_jp_{j+1})}{\Sigma_2(\{p\}_n)} \quad \text{with} \quad \Sigma_2(\{p\}_n)=\sum_{i\neq j}^{n}(p_ip_{i+1})(p_jp_{j+1})\ . \tag{3.8.33}$$

Choose for both spots wether to connect forwardly or backwardly, i.e.

$$(k,l)\in\{ij\}_+=\Big\{(i,j),\ (i,j+1),\ (i+1,j),\ (i+1,j+1)\Big\}\ . \tag{3.8.34}$$

This allows in the mean to get all connections if we allow a sum of scalar products

in the nominator, since

$$\frac{1}{p_i p_0} + \frac{1}{p_{i+1} p_0} = \frac{p_{i+1} p_0 + p_i p_o}{(p_i p_0)(p_0 p_{i+1})} \,, \tag{3.8.35}$$

and if we take out the bias out of the four possibilities. This means, we have to choose k, l according to $\mathcal{N}(p_k, p_l)^{-1} / \sum_{(q,r)\in\{ij\}_+} \mathcal{N}(p_q, p_r)^{-1}$ such that the generated density is

$$\begin{aligned} B_2(\{p\}_n) = \Phi(\{p\}_n) \cdot \sum_{i\neq j}^{n} \frac{(p_i p_{i+1})(p_j p_{j+1})}{\Sigma_2(\{p\}_n)} \cdot \frac{\sum_{(k,l)\in\{ij\}_+} (p_k p_0)_\delta (\tilde{p}_0 p_l)_\delta}{\sum_{(q,r)\in\{ij\}_+} \mathcal{N}(p_q, p_r)^{-1} / 2\pi} \\ \cdot \frac{1}{(p_i p_0)_\delta (p_0 p_{i+1})_\delta (p_j \tilde{p}_0)_\delta (\tilde{p}_0 p_{j+1})_\delta} \,, \end{aligned} \tag{3.8.36}$$

just as desired.

So far all densities for choosing i, j were fixed by the APS we want to build. But it remains free to choose from the two possibilities of how to insert p_0 and $\tilde{p}_0$, analogously to the different permutation channels and can possibly be tackled with the adaptive multichannel method [KP94]. A reasonable choice for constant weights yields the density

$$G^{\mathrm{QCD}}(\{p\}_n) = \frac{1}{n!} \sum_{\sigma\in S_n} \Phi\left(\{p\}_{\sigma(n)}\right) \frac{(p_0\tilde{p}_0)\Sigma_1 \cdot B_1 + \Sigma_2 \cdot B_2}{(p_0\tilde{p}_0)\Sigma_1 + \Sigma_2}\left(\{p\}_{\sigma(n)}\right) \,. \tag{3.8.37}$$

Even with the power of literate programming the corresponding subroutine looks a bit complicated, which is of course reflected in higher computational costs for the phase space generation.

⟨*Implementation of* `ovm_phasespace` *procedures*⟩+≡

```
subroutine sarge2(roots, lxi, Nin, p, weight, perm, mode)
  ⟨Variables of sarge2⟩
  ⟨Initialization of perms, Nout and delta⟩
  ⟨Draw a perm from table_perms⟩
  ⟨Draw xis from the polytope⟩
  ⟨Generate full antenna for final states and update pdotp⟩
  if (Nout > 2) then
    ⟨Compute adjacent scalar products to choose an insertion place⟩
    ⟨Choose between B1 and B2 according to their weights⟩
    ⟨If channel B1, draw i and set j=i+1⟩
    if (⟨Channel B2 or B12⟩) then
      ⟨Compute weights in all four channels⟩
      if (ch12i .eq. 2) then
        ⟨Randomly choose to connect forwardly or backwardly⟩
```

```
      end if
    end if
    ⟨Randomly swap i and j or not⟩
    ⟨Include incoming with i and j and update pdotp⟩
    ⟨Compute weight and permute final states⟩
  else
    weight = sarge0_weight(Nin, perm, roots, lxi)
  end if
end subroutine sarge2
```

Finally, we discuss now the benefits of `sarge2`. In order to estimate the effects of choosing an insertion place with a probability corresponding to its scalar product with the next momentum, we also use constant weights (CW) as comparison. Furthermore, we compare the channel B1 with B2 and the combination B12. The results for some processes are shown in Tab. 3.2. We define the acceptance rate $a = N_{\text{ac}}/(N_{\text{ac}} + N_{\text{rej}})$, whereby these numbers relate to the cuts and not a reweighting procedure. The efficiency on the other hand is $\epsilon = \langle\sigma\rangle / \sigma_{\max}$, such that $\epsilon \cdot N_{\text{ac}}$ would be the number of *unweighted* events. Since the calculations have been run until a certain estimated error has been achieved, one should look especially for low N_{ac} as ϵ is numerically not so stable and depends strongly on whether the true maximum has been found in the respective run. Highly connective initial states like gluons or mirrors of final states as $qq' \to qq'g$ and $q\bar{q} \to q\bar{q}g$ benefit greatly from the inclusion of the initial momenta and converge more than a magnitude faster. B1 tends to perform better than B2 for three and four particles in the final state. We can see that using all permutations in terms of B12 is never worse than the slower channel and very comparable with the faster one. As it depends on which types of permutations are present in the matrix element, the relative weights of B1 and B2 should be computed in a warm up phase. $q\bar{q} \to q'\bar{q}'ng$ are the only types of processes, where the inclusion of the incoming momenta is no major improvement. This is not surprising as for $n = 1$ only two, due to initial state radiation, of the six possible scalar products between incoming and outgoing momenta are present in the matrix element at all. Here it is worth to keep the original phase space density by using constant weights without cancellations. One should also note that for these processes the number of needed points to integrate are already quite low for S0 compared to the other processes, so there is not so much room for improvement. It has been checked that the qualitative features of Tab. 3.2 are also apparent in various other processes and for other cuts.

Table 3.2. The number of needed accepted phase space points until the integration has reached the respective estimated error alongside the acceptance rate, efficiency and times for matrix element evaluation and phase space generation. The lower N_{acc} and the higher ϵ is, the closer is our density to the integrand and the faster the integration finishes. Compared is SARGE without incoming momenta S0, in the channels B1 and B2 and the combination according to Eq. (3.8.37) each with the weights according to Eqs. (3.8.31) and (3.8.33) and with constant weights (CW). The numbers are obtained by avering over 10 seeds, sampling over helicity and summing over colors with the fixed parameters $\tau = 0.01$ and $\sqrt{s} = 1000\,\text{GeV}$. As usual, the statistical error is denoted in parantheses in units of the last digit. Empty parentheses correspond to errors below the shown digits.

$gg \to ggg$ rel. err. 1%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	1010(1)	34.6(5)	92.9(10)	70.7(9)	251(12)	38.9(9)	166(23)
a[%]	85.2()	50.8()	56.2(1)	52.5()	52.5()	51.1()	54.4()
$\epsilon/10^{-4}$	7.31(27)	103(12)	38.4(25)	60.1(16)	6.69(76)	70.5(97)	13.5(22)
t^{ME}[µs]	91.6(26)	89.6(43)	92.4(27)	99.4(12)	86.8(33)	97.7(38)	88.6(36)
t^{PS}[µs]	4.93(14)	18.7(12)	17.5(6)	31.5(5)	26.2(11)	32.0(11)	26.4(11)

$gg \to gggg$ rel. err. 2%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	476(18)	31.9(28)	86.1(141)	55.3(45)	81.7(53)	28.2(28)	63.8(64)
a[%]	44.4()	30.6()	32.6()	30.3()	28.4()	30.5(1)	30.5()
$\epsilon/10^{-4}$	1.81(18)	14.9(18)	6.54(109)	8.15(89)	6.09(98)	21.6(28)	8.30(104)
t^{ME}[µs]	476(1)	477(2)	479(2)	491(7)	484(7)	493(12)	481(3)
t^{PS}[µs]	8.39(7)	20.7(6)	19.5(3)	28.4(6)	29.8(5)	28.8(8)	28.8(4)

$qq' \to qq'g$ rel. err. 1%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	6640(4)	239(5)	620(5)	513(6)	1750(10)	256(5)	1010(4)
a[%]	85.2()	50.8()	56.2()	52.5()	52.5()	51.1()	54.4()
$\epsilon/10^{-4}$	1.12(4)	14.0(23)	6.10(19)	6.98(88)	1.06(14)	12.1(12)	1.61(12)
t^{ME}[µs]	19.7(6)	19.7(10)	20.9(9)	20.2(9)	19.2(7)	20.2(10)	19.1(10)
t^{PS}[µs]	4.58(16)	17.6(9)	16.6(8)	25.6(12)	24.6(9)	27.0(13)	24.1(12)

Table 3.2 (cont.) Overall we see an immense reduction in most cases in the number of points that are needed to integrate a process to the desired precision by including the incoming momenta in the APS. As expected, constant weights worsen here the importance sampling since they have an additional divergence that is not included in the matrix element. B1 tends to perform better than B2 for three and four particles in the final state. The combination of all permutations in terms of B12 is never worse than the slower channel and comparable with the faster one.
$q\bar{q} \to q'\bar{q}'ng$ are the only types of processes, where the inclusion of the incoming momenta is no major improvement. Here, constant weights are preferable. The time for the phase space generation per accepted phase space point roughly triples compared to S0, because of the cuts that give lower acceptance rates and the increased computational effort. They are, however, usually dominated by the times for the evaluation of the matrix elements, which are of course constant for all modes.

$q\bar{q} \to q\bar{q}g$ rel. err. 1%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	6430(7)	239(5)	614(9)	523(23)	1760(9)	252(3)	1120(7)
a[%]	85.2()	50.7()	56.2()	52.5()	52.5()	51.1()	54.4()
$\epsilon/10^{-4}$	1.23(4)	11.9(16)	5.86(43)	6.84(92)	0.991(99)	13.7(13)	1.60(23)
t^{ME}[µs]	22.7(9)	22.1(11)	24.1(11)	21.8(11)	23.1(11)	22.9(11)	22.3(11)
t^{PS}[µs]	4.60(19)	16.6(7)	16.4(8)	24.3(12)	25.7(12)	26.1(14)	24.4(12)

$q\bar{q} \to q'\bar{q}'g$ rel. err. 1%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	90.9(17)	211(9)	68.7(8)	58.6(6)	56.2(3)	182(16)	69.4(8)
a[%]	85.3()	50.8()	56.2()	52.5(1)	52.5()	51.1()	54.3()
$\epsilon/10^{-4}$	50.1(30)	9.63(122)	59.9(31)	68.5(44)	83.8(44)	13.0(21)	47.6(37)
t^{ME}[µs]	10.5(1)	10.9(3)	11.0(3)	11.0(4)	11.3(2)	11.0(2)	11.5(2)
t^{PS}[µs]	2.90(6)	9.30(15)	8.67(14)	12.6(3)	12.5(4)	12.5(2)	12.0(3)

$q\bar{q} \to q'\bar{q}'ggg$ rel. err. 2%

	S0	B1	B1_CW	B2	B2_CW	B12	B12_CW
$N_{\text{ac}}/10^3$	65.7(93)	129(15)	57.2(41)	53.1(110)	38.1(24)	86.8(195)	37.7(28)
a[%]	15.5()	11.8()	12.0()	11.1(1)	10.4()	11.7()	11.3()
$\epsilon/10^{-4}$	3.93(47)	1.09(17)	3.09(33)	4.16(88)	4.04(55)	2.66(50)	5.76(102)
t^{ME}[µs]	232(4)	224(1)	229(3)	224(1)	227(3)	227(2)	226(2)
t^{PS}[µs]	43.2(9)	83.5(6)	81.2(9)	108(1)	118(2)	106()	109(1)

4. Helicity Summation and Spin Correlations

The purpose of this chapter is twofold. At first, different possibilities to carry out the helicity sum are discussed for the purpose of speeding up the calculation of cross sections with many legs. Hereby, we distinguish between methods which use a fixed set of helicities and more general parametrizations with superpositions of helicities. This will furthermore pave the way for the generation of events with spins in a certain direction in their rest frame. Such a feature is usually not implemented in automated MC event generators and is especially important if one wants to study spin correlations in the decays of massive particles like the τ lepton or t quark.

Note, that for the τ lepton, there exists a well established tool, namely TAUOLA, which can decay τ's in HEPMC event records. However, the spin correlations are implemented for standard processes mediated by Z/γ by using spin density matrices that are calculated by hand with some approximations [DN12]. Further limitations are the requirement, that the τ is on-shell, and the algorithm, which is based on interpreting a spin weighting factor as a probability whether an event should be accepted or rotated. Another recent program aiming to overcome these problems, based on MADGRAPH, is TAUDECAY [Hag+12]. Our approach differs from these as we are able to explicitly construct the density matrix that corresponds to a spin in a certain direction for each external particle, as will be shown below, and are not depending on MADGRAPH. If the full phase space integration is too expensive, we can still set the τ on-shell and use factorization to generate events while maintaining full spin correlations.

4.1. Fixed Helicity Sampling

To improve the overall performance of the MC integration various helicity techniques can be employed. The basic idea is to include the helicity sum in the MC evaluation of the integration

$$\sum_{h=1}^{N_{\text{hel}}} \int \mathcal{D}p \; |\mathcal{M}_h|^2 = \frac{N_{\text{hel}}}{N} \sum_i |\mathcal{M}_{h_i}|^2 (p_i) \; , \tag{4.1.1}$$

whereby $|\mathcal{M}_h|^2$ should be understood as flavor- and color-summed, if necessary. For illustration, think of a toy problem where $|\mathcal{M}_h|^2$ is roughly the same for all h. By

computing all helicity combinations for each phase space point, N_{hel} times too many points are used for the same information. If instead only a random helicity combination h_i is calculated, N_{hel} times more independent phase space points can be used while still having the same number of matrix element evaluations. In general, the additional variance, which is zero in our toy problem, has to be reduced, in order to maintain the benefits of improved sampling of the phase space.

4.1.1. Complete Sum

For comparison the sum over all helicities gets computed in every phase point. Note, that this is the de facto standard approach to compute helicity sums, not only in many tree- but also loop-level automated calculations [HP99]. We will show that the sum is almost always inferior to the MC methods if it is implemented in a Single Instruction Multiple Data (SIMD) approach, i.e. the same matrix element is evaluated for different values of the polarization of the external wave functions. When different chiralities are treated as different particles with appropriate Feynman rules as in the Weyl-van-der-Waerden formalism [Dit98], helicity MC seems to become preferably only for very large multiplicities like more than 11 external particles [GH08]. This is due to the possible reuse of 1POWs between amplitudes where only some helicities are flipped.

4.1.2. Discrete Importance Sampling

If the calculation is restricted to a fixed set of helicities, the most elaborate method is discrete MC with importance sampling. Hereby, the necessary weights w_h can be generated in a warm up phase with N_W sampling points from amplitudes $|\tilde{\mathcal{M}}_h|^2$. Results of this phase are discarded before the final phase in order to guaranty that the result is unbiased. In other words

$$\sum_h |\mathcal{M}_h|^2 = \sum_h w_h \frac{|\mathcal{M}_h|^2}{w_h} = \left\langle \frac{|\mathcal{M}_h|^2}{w_h} \right\rangle_w \quad \text{with} \quad w_h = \frac{\left|\tilde{\mathcal{M}}_h\right|^2}{\sum_j \left|\tilde{\mathcal{M}}_j\right|^2}\,, \tag{4.1.2}$$

where the deterministic sum has been replaced in the last step by the stochastic sampling $\langle \circ \rangle_w$ according to the probabilities w_h. This includes the case of equal weights or uniform probability distribution where $1/w_h = 1/w = N_{\text{hel}}$. The scaling of $|\mathcal{M}_h|^2$ with $1/w_h$ leads to a smaller variance and hence over Eq. (3.3.2) to a smaller error of the MC integration. Furthermore, it is a natural way to exclude

helicity combinations that have a vanishing matrix element since the probability for these vanishes as well. This means, that we do not need some heuristic approach to exclude such combinations, which is currently seen in WHIZARD or MADGRAPH, and that we can gradually lower the probability of sampling these instead of simply turning them on and off.

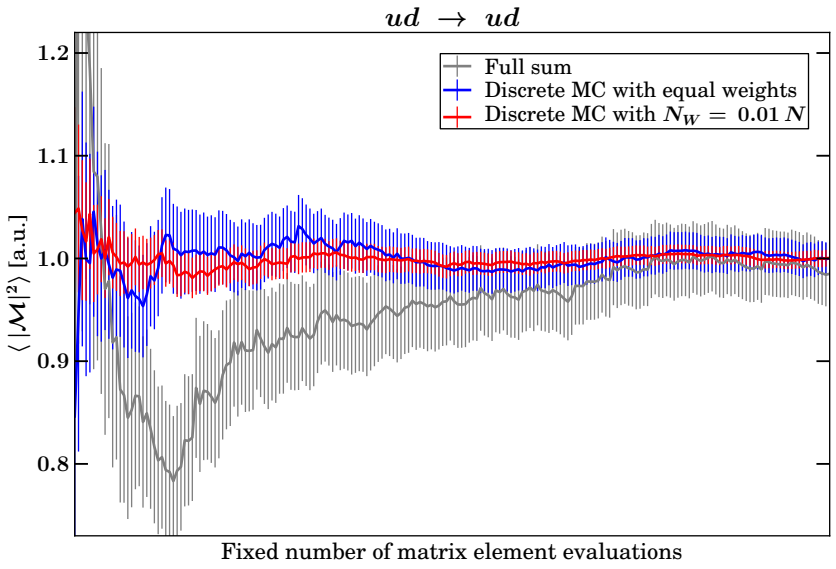


Figure 4.1. The convergence of the cross section with HMC and helicity sum for $2 \to 2$ scattering, integrated with RAMBO. For a fixed number of matrix element evaluations, helicity MC clearly profits from the additional independent momentum phase space points. Moreover, one can see that a small investment in a warm up phase to generate weights (N_W) can significantly reduce the variance opposed to uniform probabilities.

As illustrative example, the convergence rate for $ud \to ud$ is shown in Fig. 4.1. It is obvious that here for a fixed number of matrix element evaluations, helicity MC is superior to the classical full sum. If this is also reflected in the overall time, which is needed to compute a process to a requested error, will be analyzed later on. Although the discrete MC with importance sampling clearly shows the smaller error bands, it should be stressed that one cannot draw conclusions from the convergence rate of a single run. Hence, we will employ statistics and take into account multiple processes with several independent seeds in Section 4.4.

4.2. Superpositions of Helicities

While the above method can easily be implemented on the top level of the computation, we will now relax this restriction and allow for arbitrary superpositions of

helicities. This has to be implemented on the lowest level, namely the construction of the external wave functions. In the amplitude computation by the OVM, it is no problem to use linear combinations of external wave functions. For the sake of simplicity, we will stick to QCD which implies that $N_{\text{hel}} = 2^{N_{\text{prt}}}$ and only spin 1/2 spinors and massless spin 1 vector bosons are involved.

4.2.1. Continuous Helicity Sampling

Continuous helicity sampling replaces the sum over discrete spin configurations with an integral over superpositions of helicities. E.g. for a massless vector-boson field ϵ^μ, this can be achieved by using a superposition of the two physical polarization vectors

$$\epsilon^\mu_\theta(p) = \mathrm{e}^{\mathrm{i}\theta}\epsilon^\mu_+(p) + \mathrm{e}^{-\mathrm{i}\theta}\epsilon^\mu_-(p)\ . \tag{4.2.1}$$

Note, that this is a real polarization since $\epsilon^*_+ = \epsilon_-$. Furthermore, we can see that $(\epsilon_\theta/\sqrt{2})^2 = -1$ and $k_\mu\epsilon^\mu_\theta = 0$. But, one should be careful not to use it blindly as a set of polarization vectors, because $\epsilon^*_{\theta\,\mu}\epsilon_{\theta'}{}^\mu/2 = -\cos(\theta-\theta') \neq 0$ in general. To establish the connection to spin-summed amplitudes, we remember the helpful role of projectors, i.e. polarization sums, in analytical calculations of matrix elements

$$\Pi(p) = \sum_s |p,s\rangle\langle p,s| \tag{4.2.2}$$

or in our example

$$\Pi^{\mu\nu}(p) = \sum_{\lambda=\pm}\epsilon^\mu_\lambda(p)\epsilon^\nu_\lambda(p)^*\ . \tag{4.2.3}$$

Of course, we cannot use the explicit, analytical representation of $\Pi^{\mu\nu}(p)$ in terms of $g^{\mu\nu}$ and p^μ in a numerical setup like O'Mega. But, following the idea of Ref. [DKP98], we may replace the sum by an integral over the states of Eq. (4.2.1)

$$\frac{1}{\pi}\int_0^\pi \mathrm{d}\theta\,\epsilon^\mu_\theta(p)\epsilon^\nu_\theta(p)^* = \sum_{\lambda=\pm}\epsilon^\mu_\lambda(p)\epsilon^\nu_\lambda(p)^*\ , \tag{4.2.4}$$

which can be verified trivially by inserting the definition of ϵ^μ_θ. Suppose now, that we have calculated the matrix element $\mathcal{M}_\lambda$ for the transition of an initial state $|A\rangle$ to some $|B,\epsilon_\lambda\rangle$, where $|B\rangle$ may be any free, $n-1$ multi-particle state in a $2\to n$

scattering. For the spin sum of the vector boson follows

$$\sum_{\lambda} |\mathcal{M}_{\lambda}|^2 = \sum_{\lambda} \left\langle A \left| \mathcal{T} \right| B, \epsilon_{\lambda} \right\rangle \left\langle B, \epsilon_{\lambda} \left| \mathcal{T}^{\dagger} \right| A \right\rangle \tag{4.2.5}$$

$$= \frac{1}{\pi} \int_0^{\pi} \mathrm{d}\theta \, \left\langle A \left| \mathcal{T} \right| B, \epsilon_{\theta} \right\rangle \left\langle B, \epsilon_{\theta} \left| \mathcal{T}^{\dagger} \right| A \right\rangle \, , \tag{4.2.6}$$

where $\mathcal{T}$ is the transition matrix, connected to the scattering matrix S via $\mathrm{i}(\mathbb{1} - S) = (2\pi)^4 \delta\left(p_{\mathrm{out}} - p_{\mathrm{in}}\right) \mathcal{T}$.[1] The generalization of this to all $n+2$ helicities and spinors u, v is obvious and leads to an $n+2$ dimensional integration instead of a sum over the 2^{n+2} possible spin configurations. So far no approximation is involved and it is not really intuitive why an integral should be computationally less expensive than a sum over two elements. But, using a MC sampling, only one random phase is used per independent momentum phase points. Since the MC error does not depend on the dimension, merely the price of additional variance and calculation of oscillating terms with mean zero has to be paid. To verify the significant speed up by an order of magnitude in convergence, which has been reported in Ref. [Oor12], we will test this parametrization thoroughly in Section 4.4.

4.2.2. General Helicity MC Formalism

The above is merely one example of how one can parametrize a sampling of the helicity sum. We will now formulate the necessary and sufficient conditions that have to be fulfilled by a sampling, which can be either deterministic or stochastic. Given a particle u with two helicities, we can decompose $\Pi = u\bar{u}$ into four linear independent terms

$$\begin{aligned} \Pi = \quad & f(\boldsymbol{\theta}) u_+ \bar{u}_+ + g(\boldsymbol{\theta}) u_+ \bar{u}_- \\ & + h(\boldsymbol{\theta}) u_- \bar{u}_+ + j(\boldsymbol{\theta}) u_- \bar{u}_- \, , \end{aligned} \tag{4.2.7}$$

where $\boldsymbol{\theta}$ can be any real or complex $n-$dimensional vector. The only condition that must be imposed to obtain asymptotically the result of the traditional helicity sum is

$$\begin{aligned} \langle f(\boldsymbol{\theta}) \rangle_{\boldsymbol{\theta}} &= \langle j(\boldsymbol{\theta}) \rangle_{\boldsymbol{\theta}} = 1 \\ \langle g(\boldsymbol{\theta}) \rangle_{\boldsymbol{\theta}} &= \langle h(\boldsymbol{\theta}) \rangle_{\boldsymbol{\theta}} = 0 \, . \end{aligned} \tag{4.2.8}$$

[1] In color flow basis, we are actually computing $\sum_{\lambda} \sum_{n=1}^{N_{\mathrm{cfactors}}} \mathrm{cf}_n \mathcal{M}_{\lambda}^{i_n} \mathcal{M}_{\lambda}^{j_n *}$. However, this does not violate the validity of Eq. (4.2.6) when the same θ phases are used for all colors and especially for $\mathcal{M}^{i_n}$ and $\mathcal{M}^{i_n *}$.

To explore this formulation, we will begin with two possible and orthogonal parametrizations for $n = 1$. In the following section, these will be shown to be special cases of an $n = 2$ dimensional parametrization of the unit sphere of the spin density matrix, if u is a massive spin 1/2 spinor.

At first, we will recover the *exponential* parametrization, which has been introduced in Section 4.2.1. With the analogue of Eq. (4.2.1) for u, immediately follows

$$\begin{aligned}\Pi = \quad & 1 \quad u_+\bar{u}_+ + \mathrm{e}^{2\mathrm{i}\theta}\, u_+\bar{u}_- \\ & + \mathrm{e}^{-2\mathrm{i}\theta}\, u_-\bar{u}_+ + 1 \quad u_-\bar{u}_- \ ,\end{aligned} \tag{4.2.9}$$

which fulfills Eq. (4.2.8) if we take a sampling in which the mean of the mixing terms vanishes. Sampling the integral

$$\langle \circ \rangle_\theta = \frac{1}{\pi}\int_0^\pi \mathrm{d}\theta \ \circ \quad \to \quad \sum_{\theta_i \in [0,\pi]} \circ \ , \tag{4.2.10}$$

whereby the sum should indicate a MC sampling, is as valid as choosing randomly between two phases

$$\langle \circ \rangle_\theta = \frac{1}{2}\sum_{\theta = 0, \frac{\pi}{2}} \circ \quad \to \quad \sum_{\theta_i \in \{0, \frac{\pi}{2}\}} \circ \ . \tag{4.2.11}$$

In each case, we compute $u\bar{u} = (u_+\bar{u}_+ + u_-\bar{u}_- + \text{osz.})$ with a single matrix element evaluation. However, the sampling does affect the convergence of the mixing terms. A drawback of this parametrization is that it is by construction not suitable for the generation of events with a certain helicity.

To obtain weights for the different helicities, a *trigonometrical* parametrization can be used

$$u = \cos\theta\ u_+ + \sin\theta\ u_- \tag{4.2.12}$$

leading to

$$\begin{aligned}\Pi = \quad & \left(\cos\theta\right)^2 \quad u_+\bar{u}_+ + \sin\theta\ \cos\theta\, u_+\bar{u}_- \\ & + \sin\theta\ \cos\theta\, u_-\bar{u}_+ + \left(\sin\theta\right)^2 \quad u_-\bar{u}_- \ .\end{aligned} \tag{4.2.13}$$

The corresponding samplings, which fulfill Eq. (4.2.8), are e.g.

$$\langle \circ \rangle_\theta = \frac{1}{\pi/2} \int_0^\pi \mathrm{d}\theta \circ \quad \to \quad 2 \sum_{\theta_i \in [0,\pi]} \circ \tag{4.2.14}$$

or as discrete version

$$\langle \circ \rangle_\theta = \sum_{\theta=0,\frac{\pi}{2}} \circ \qquad \to \quad 2 \sum_{\theta_i \in \{0,\frac{\pi}{2}\}} \circ \ . \tag{4.2.15}$$

Eq. (4.2.15) corresponds to randomly selecting a helicity or in other words, the sampling of Section 4.1.2 with equal weights. In helicity conserving processes, we can expect a rather large variance of this discrete sampling since many helicity combinations give a vanishing matrix element. The natural choice to avoid this, is to use $\theta \in \{\frac{\pi}{4}, \frac{3}{4}\pi\}$ instead. This is equivalent to the discrete sampling of the exponential parametrization but the computation time may differ since it is calculated in another way.

4.3. Spin Density Matrices in the Rest Frame

It turns out that the ad hoc parametrizations that were found earlier are just lines on the unit sphere of the spin density matrix of a spin 1/2 particle. As known from basic quantum mechanics [Sch07b], the density matrix ρ for a spin in the $\boldsymbol{\alpha}$ direction, may be written with the Pauli matrices σ_i as

$$\rho(\boldsymbol{\alpha}) = \frac{1}{2}\left(\mathbb{1} + \boldsymbol{\alpha}\boldsymbol{\sigma}\right) \quad \text{with} \quad \boldsymbol{\sigma} = (\sigma_x,\ \sigma_y,\ \sigma_z)\ . \tag{4.3.1}$$

This can be verified, by computing

$$\langle \boldsymbol{\sigma} \rangle = \mathrm{Tr}\left[\rho\boldsymbol{\sigma}\right] = \mathrm{Tr}\left[\frac{1}{2}(\mathbb{1} + \boldsymbol{\alpha}\boldsymbol{\sigma})\boldsymbol{\sigma}\right] = \boldsymbol{\alpha}\ . \tag{4.3.2}$$

Especially in terms of quantum information, $\boldsymbol{\alpha}$ is also known as Bloch vector. If ρ should describe a pure state, it has to be idempotent, i.e. $\rho^2 = \rho$, which requires $|\boldsymbol{\alpha}| = 1$ since $\sigma_i^2 = 1$. Geometrically, this is related to the fact that a pure state may not be expressed as convex combination of other states. Obviously, these states lie on the boundary of the domain of $\rho(\boldsymbol{\alpha})$ while fully mixed states are in the center $\boldsymbol{\alpha} = \mathbf{0}$.

Using the canonical spherical coordinates for radius one

$$\boldsymbol{\alpha} = (\sin\theta\ \cos\varphi,\ \sin\theta\ \sin\varphi,\ \cos\theta)\ , \tag{4.3.3}$$

we obtain with help of Euler's formula

$$\rho(\theta,\ \varphi) = \begin{pmatrix} \cos^2\left(\frac{\theta}{2}\right) & \frac{1}{2}e^{-\mathrm{i}\varphi}\sin(\theta) \\ \frac{1}{2}e^{\mathrm{i}\varphi}\sin(\theta) & \sin^2\left(\frac{\theta}{2}\right) \end{pmatrix}\ , \tag{4.3.4}$$

which looks rather familiar if compared with Eq. (4.2.9) and Eq. (4.2.13). This matrix is of course written in a basis in which S_z is diagonal $\langle\uparrow| = (1,\,0)$. $\rho(\alpha) = |\alpha\rangle\langle\alpha|$ should be understood as dyadic product and $\rho(\boldsymbol{e}_z) = |\uparrow\rangle\langle\uparrow|$ can be checked in Eq. (4.3.4) for $\theta = 0$. So, by sampling θ and φ with spinors with momentum p, we are actually setting up a density matrix $\rho(\boldsymbol{\alpha})$ with $\rho(\boldsymbol{e}_z) = |u_+\rangle\langle u_+|$. How this object transforms under Lorentz transformations, will be shown in Section 4.5. Reformulating Eq. (4.2.5) in these terms, the helicity sum becomes a trace over the density matrix $(\rho_z + \rho_{-z})$,

$$\sum_\lambda |\mathcal{M}_\lambda|^2 = \mathrm{Tr}\left[\left\langle A\right|\mathcal{T}\Big(|B\rangle\langle B|\otimes(\rho_z+\rho_{-z})\Big)\mathcal{T}^\dagger\left|A\right\rangle\right] \tag{4.3.5}$$

or if $|A\rangle$ and $|B\rangle$ also consist of spin 1/2 particles, which are summed over,

$$\begin{aligned} \frac{1}{2^{N_\mathrm{in}}}\sum_{\lambda_1,\ldots\lambda_{n+2}} |\mathcal{M}_{\lambda_1,\ldots\lambda_{n+2}}|^2 &= 2^{N_\mathrm{out}}\,\mathrm{Tr}\left[\rho_\mathrm{in}\,\mathcal{T}\,\rho_\mathrm{out}\,\mathcal{T}^\dagger\right] \\ \text{with}\quad N_\mathrm{in} = 2 \quad\text{and}\quad \rho_\mathrm{in} &= \frac{1}{4}\mathbb{1}_A = \frac{(\rho_z+\rho_{-z})^{\otimes 2}}{4} \end{aligned} \tag{4.3.6}$$

and $\rho_\mathrm{out} = \frac{1}{2^{N_\mathrm{out}}}\mathbb{1}_{B'}$, whereby A and B' are the full incoming and outgoing states. Note that the trace goes over two elements in Eq. (4.3.5) and over 2^{n+2} elements in Eq. (4.3.6).

The original motivation of this chapter, or more generally in jet physics if one is not interested in spin correlations, is to speed up the calculation of the $\mathbb{1}$. As noted earlier, this is the fully mixed state consisting of $\uparrow$ and $\downarrow$. However, we cannot obtain this $\mathbb{1}$ directly since our physical states in the Hilbert space are pure states that lie on the unit sphere. The first parametrization, which was introduced in Section 4.2.1, is $2\rho(\pi/2, -2\varphi)$ and the continuous integral over φ is just the line integral along the equator while the discrete sampling of this at $\varphi = 0, \pi/2$ is ρ_x and ρ_{-x}. The second parametrization shown in Section 4.2.2 in terms of sin and cos is orthogonal to the first, connecting north and south pole of the unit sphere and is equivalent to $\rho(2\theta, 0)$.

Viewed in this light, there is no good argument why either parametrization should be superior, which will be confirmed by numerical data later on.

In order to use this parametrization for calculations of the helicity sum, we should now consider to fulfill Eq. (4.2.8) with Eq. (4.3.4). Possible normalizations are

$$\langle \circ \rangle_{\theta,\varphi} = \frac{1}{\pi^2} \int_0^\pi \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\varphi \circ \qquad \to \quad 2 \sum_{\substack{\theta_i \in [0,\pi] \\ \varphi_i \in [0,2\pi]}} \circ \tag{4.3.7}$$

or

$$\langle \circ \rangle_{\theta,\varphi} = \frac{1}{2\pi} \int_{-1}^1 \mathrm{d}\cos\theta \int_0^{2\pi} \mathrm{d}\varphi \circ = \frac{1}{2\pi} \int \mathrm{d}\Omega \circ \qquad \to \quad 2 \sum_{\substack{\cos\theta_i \in [-1,1] \\ \varphi_i \in [0,2\pi]}} \circ \,. \tag{4.3.8}$$

Note that although the probability density for θ is either unity or $\sin\theta$, the normalization and mean remain the same. As in the case of discrete samplings, different samplings can still yield the same mean information. However, the uniform sampling of $\cos\theta$ can be regarded as more natural, since it yields a uniform sampling of the sphere without any prior to the momentum direction. Despite the fact that we are analyzing massive spin 1/2 particles in this section, it is of course possible to use analogue parametrizations for massless particles. E.g. for gluons

$$\epsilon_{\theta,\varphi}(\boldsymbol{p}) = \mathrm{e}^{-\mathrm{i}\varphi/2} \cos\left(\frac{\theta}{2}\right) \epsilon_+(\boldsymbol{p}) + \mathrm{e}^{+\mathrm{i}\varphi/2} \sin\left(\frac{\theta}{2}\right) \epsilon_-(\boldsymbol{p}) \,. \tag{4.3.9}$$

The interpretation of a spin in a certain direction is in this case not meaningful, since we cannot go in the rest frame of a massless particle. Nevertheless, we use it as *spherical* parametrization in the context of speed testing.

Note, that the identification of the real direction of the spin in three dimensional space with $\boldsymbol{\alpha}$ is only possible for spin 1/2 particles due to the isomorphism between $SU(2)$ and $SO(3)/\mathbb{Z}_2$. For massive spin 1 particles, like W and Z, we could describe $\rho(\boldsymbol{\beta})$ with Gell-Mann matrices λ^a, $a = 1, \ldots 8$ as basis for 3x3 matrices whereby the polarization direction is not so easily identified with $\boldsymbol{\beta}$. The generation of events with massive vector bosons in terms of density matrices is of practical interest since it allows to factorize production and decay while keeping full spin correlations. Further elaboration for spin greater 1/2 is therefore required.

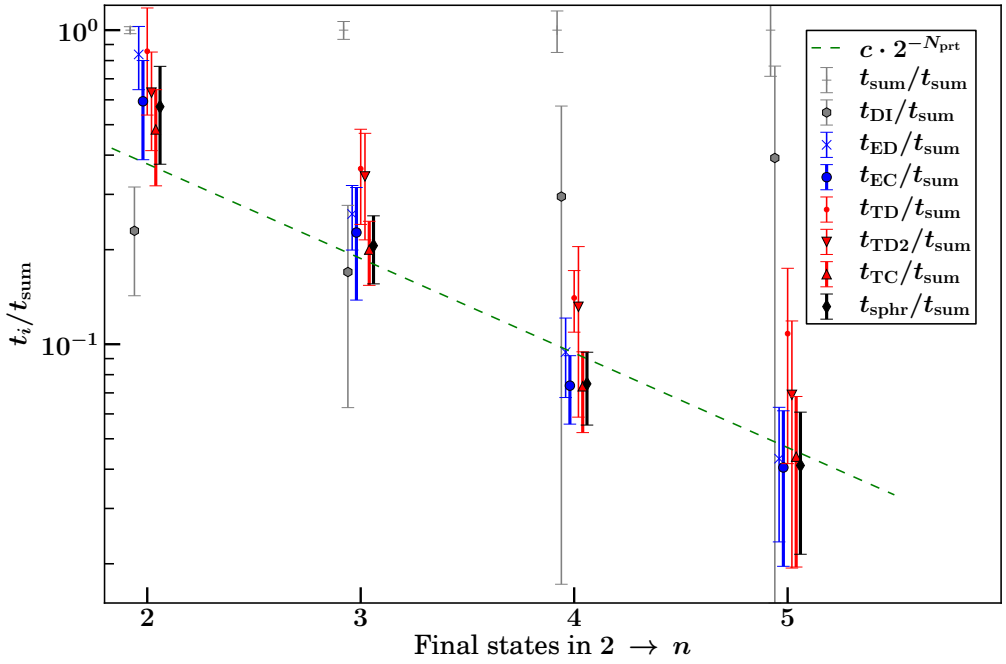


Figure 4.2. Mean execution times for various $2 \to 2, 3, 4, 5$ processes, listed in detail in Tab. B.1, until the estimated error has reached 0.2 %, 1 %, 3 % and 3 %, each averaged over 15 different seeds. The different modes are encoded as follows: DI = *discrete* helicity MC with *importance* sampling, {ED, EC} = *exponential* parametrization with *discrete* and *continuous* sampling, respectively, {TD, TD2, TC} = *trigonometrical* parametrization with $\theta \in \{0, \pi/2\}$, $\theta \in \{1/4\,\pi, 3/4\,\pi\}$, $\theta \in [0, \pi]$ and sphr = *spherical* parametrization as described in the text. For improved visibility, different methods are shifted around integer n. The whole batch has been computed with $N_W = 1000$, which is obviously insufficient to generate good weights for $n > 3$. Continuous samplings are denoted by a thicker symbol and tend to be faster than the discrete ones.

4.4. Discussion and Performance

All parametrizations mentioned in this chapter have been implemented in the OVM and tested in detail. The mean computation times for various processes are listed in Tab. B.1. Results obtained with different methods agree with the full sum within their respective error. Furthermore, the standard deviation between different runs with the same method is approximately the desired error for $2 \to 2$ and $2 \to 3$ processes, thereby confirming the reliability of the error estimation which is done according to Appendix A. For $2 \to 4$ and $2 \to 5$ the RAMBO algorithm can lead to apparent convergence because the variance gets smaller and smaller until the algorithm hits another peak. If the termination criterion, i.e. $\delta I/I < 3\,\%$, is fulfilled before the next peak is reached, this leads to a wrong result. By taking the mean over 15 runs, we still obtain agreement for the different methods, but with larger errors, which is also

reflected in the larger errors of the time measurement.

The full sum matrix elements for $2 \to 2$ processes have also been checked to agree with the analytic expressions for $\sum |\mathcal{M}|^2$ from the literature [ESW96] in 12 digits in every phase space point, as expected when using double precision. This also confirms the correct implementation of the virtual machine and the normalization of the color flow basis. Hereby, one should keep in mind, that g in the color flow basis corresponds to $g'/\sqrt{2}$ where g' is the input parameter, taken from other measurements, for the strong coupling constant [Kil+12].

In Fig. 4.2, we only show the mean execution times averaged over all processes. Additionally, a line is shown that corresponds to the speed up that is possible, if the variance is not changed by helicity MC at all. The computation time t is roughly proportional to the number of evaluations. For unchanged variance, we have to compute N_{hel} times more points in the full sum, hence $t_{\text{HMC}}/t_{\text{sum}} \propto 2^{-N_{\text{prt}}}$. We can see that most methods in Fig. 4.2 roughly possess this behaviour. Scaling being better or worse than this can be interpreted as reduction or increase, respectively, in variance of the integrand due to the superpositions of helicities.

Furthermore, continuous methods seem to be superior to the discrete samplings when no importance sampling is employed to reduce the variance. By using importance sampling, which is very straightforward in the discrete case, it can be highly competitive. However, the number of points for the weighting phase N_W should be fine tuned or adaptively computed. This has not been done here, resulting in bad performance for $n > 3$. All three parametrizations tested here, namely exponential, trigonometrical and spherical, perform equally well and it remains a matter of choice which is used. Of course, if one is interested in generating events with spin or helicity information, spherical and trigonometrical parametrizations are the natural options.

Overall, we can see that helicity MC keeps improving over the full sum as the number of external legs grows. Hence, it is highly recommended for the calculation of multi-particle processes. Of course, one has to keep in mind that the overall variance is now the squared sum of the variance in the momentum phase space $\sigma(f_{\text{PS}})$ and the helicity $\sigma(f_{\text{H}})$: $\sigma(f)^2 = \sigma(f_{\text{PS}})^2 + \sigma(f_{\text{H}})^2$. This increase in variance can be minimized by using discrete importance sampling or a convenient parametrization in which it is already very small. To generate the weights for importance sampling, only a small fraction of total computation time $\lesssim 1\,\%$ of N has to be invested. Note, that this investment can also be used to generate weights for the other discrete variables that may be included in the MC integration, namely color and flavor. However, for a fully automated MC integrator, it is not known beforehand what $1\,\%$ of N is. In order

to still benefit from importance sampling, one could use an empiric formula that determines N_W depending on N_{prt} or keep accumulating weights until their variance is under control. It remains to be tested if adaptive variance reduction techniques for the θ phases like VEGAS [Lep78] yield a further improvement in convergence or if it is better to reserve these for the already difficult phase space integration to avoid a proliferation of integration dimensions. To ensure robustness and convergence, an adaptive change of the discrete weights during runtime has not been implemented.

4.5. Lorentz Transformations of Density Matrices

We conclude this chapter with some additional thoughts about density matrices. Though we have firm knowledge how to interpret spin density matrices at rest, in a relativistic setting this intuition may fail. We will therefore deduce the behaviour of the density matrix under Lorentz transformations strictly from the transformation of the wave functions. Note, that a consideration of the reduced spin density matrix, i.e. where the momentum has been integrated out, has been shown to have no covariant meaning [PST02], due to the correlations between spin and momentum. Despite being an interesting topic by its own, these relations are of course crucial if one wants to relate spin directions in different reference frames. As it will become evident below, the specification of a spin direction without noting the reference frame is useless. An interesting, future application of this transformation behavior might be the reconstruction of the spin of intermediate particles in cascade decays due to the spin correlations between the decay products.

Let us first concentrate on the proper, orthochronous Lorentz transformations $U(\Lambda, 0) \equiv U(\Lambda), \Lambda \in L_+^\uparrow$ without translations. As the full density matrix of all particles is merely the direct product of single particle density matrices, it is sufficient to regard the transformation of single particle states. Each momentum p^μ can be written as $L(p)k$, where $L(p)$ is a pure boost, i.e. without rotations, and $k^\mu = (m, \mathbf{0})$, in the case of positive definite mass, is the rest frame momentum. Note, that although in the rest frame there is no distinguished direction, it inherits the orientation of the coordinate system from which it is boosted back. State vectors $|p; s, m\rangle$ can be defined via standard momentum states $|k; s, m\rangle$ as

$$|p; s, m\rangle = U(L(p)) \, |k; s, m\rangle \ . \tag{4.5.1}$$

Now, the transformation of this state vector is

$$U(\Lambda)\,|p;s,m\rangle = U(\Lambda)U(L(p))\,|k;s,m\rangle \tag{4.5.2}$$

using the group property and setting $p_\Lambda = \Lambda p$

$$= U(L(p_\Lambda))U\big(L^{-1}(p_\Lambda)\Lambda L(p)\big)\,|k;s,m\rangle\ . \tag{4.5.3}$$

Hereby, we can define the Wigner rotation $W(\Lambda,p) = L^{-1}(p_\Lambda)\Lambda L(p)$ that boosts a particle from rest to p, applies Λ and brings it back to rest. Let us explain the reasoning for this three step round trip on the mass hyperboloid, $p^2 = p_\Lambda^2 = k^2 = m^2$. It is clear, that since $Wk = k$, it has to be a pure rotation $\mathrm{diag}(1,R)$. Such a transformation of the state in the rest frame under a rotation $R \in SO(3)$ can be written as [Sch07a]

$$U(R)\,|k;s,m\rangle = \sum_{m'} D^{(s)}_{m'm}(R)\,|k;s,m'\rangle\ , \tag{4.5.4}$$

where $D^{(s)}(R)$ are the $(2s+1)$ dimensional representations of the rotation group for spin s. Plugging the pieces together, we obtain

$$\begin{aligned} U(\Lambda)\,|p;s,m\rangle &= U(L(p_\Lambda))U\big(W(\Lambda,p)\big)\,|k;s,m\rangle \\ &= \sum_{m'} D^{(s)}_{m'm}(W(\Lambda,p))\,|p_\Lambda;s,m'\rangle\ . \end{aligned} \tag{4.5.5}$$

So, by inserting a $\mathbb{1}$ in Eq. (4.5.3) and using the rest frame, we are able to translate the action of a Lorentz transformation Λ on a state with p into a rotation of the states with p_Λ. Note, that the explicit use of the rest frame indicates obstacles for massless particles. However, here the group that leaves k^μ invariant, which is in general called *little group*, are the rotations in 2D around the direction of motion of the particle, $k^\mu = (1,0,0,1)$, $SO(2)$ as well as translations $T(3)$, i.e. $ISO(2) \equiv E_2$ [AM02]. In fact, E_2 is a semi-direct product of the two abelian subgroups $T(3)$ and $SO(2)$. By computing the corresponding Lie algebra of E_2, one finds that two of the generators, which are combinations of rotations and translations, commute with each other and can therefore simultaneously be diagonalized [Wei05]. The problem with these states is that if one set with non-zero eigenvalues exists, a whole continuum of states must exist since the generators can be continuously rotated onto each other. Since we don't see massless particles with a continuous degree of freedom for fixed momenta in nature, physical states are defined with zero eigenvalues with respect to the mentioned operators. What remains is the third generator J_3, which is the

angular momentum in the direction of motion for our chosen standard vector k^μ, also called *helicity*. The translations of E_2 also play an important role in realizing that no four-vector field can be constructed with creation and annihilation operators of a particle of mass zero and helicity ± 1 if not additionally gauge invariance is demanded [Wei05].

After this small digression, we return now to massive spin 1/2 particles and make the above a bit more explicit. Consider p^μ along the z axis with rapidity η: $p^\mu = m(\cosh\eta, 0, 0, \sinh\eta)$. A further boost in the z axis would lead to an identity matrix for W. Thus, without loss of generality, we make a boost in the x direction with rapidity ω. By calculating the boost matrices, one can obtain $W(\Lambda, p)$ for this case. It acts on a spatial vector $z^\mu = (0, 0, 0, 1)$ as [AM02]

$$(Wz)^\mu = \begin{pmatrix} 0 \\ \sinh\eta\sinh\omega/(1+\cosh\eta\cosh\omega) \\ 0 \\ (\cosh\omega+\cosh\eta)/(1+\cosh\eta\cosh\omega) \end{pmatrix} \tag{4.5.6}$$

defining the Wigner angle

$$\tan\Omega_p = \frac{\sinh\eta\sinh\omega}{\cosh\eta+\cosh\omega} \tag{4.5.7}$$

with $\Omega_p < \theta$ for $0 \leq \eta, \omega < \infty$, where $\theta = \measuredangle(\boldsymbol{p}_\lambda, \boldsymbol{p})$ and $\tan\theta = \frac{\sinh\omega}{\tanh\eta}$. So, a boost in the x direction does not only rotate $\boldsymbol{p}$ around the y axis but also the spin state in the same sense but with lesser magnitude.

For spin 1/2 particles, the D matrices of Eq. (4.5.5) take the elegant form

$$D^{(1/2)}_{m'm}(W(\Lambda,p)) = \exp\left\{-\mathrm{i}\,\boldsymbol{\Omega}_p\frac{\boldsymbol{\sigma}}{2}\right\} \equiv \cos\frac{|\boldsymbol{\Omega}_p|}{2} - \mathrm{i}\,\frac{\boldsymbol{\Omega}_p}{|\boldsymbol{\Omega}_p|}\boldsymbol{\sigma}\sin\frac{|\boldsymbol{\Omega}_p|}{2}\,, \tag{4.5.8}$$

which simplifies for the considered rotation around y to

$$\begin{pmatrix} u(p,+\frac{1}{2})' \\ u(p,-\frac{1}{2})' \end{pmatrix} = \begin{pmatrix} \cos\frac{\Omega_p}{2} & \sin\frac{\Omega_p}{2} \\ -\sin\frac{\Omega_p}{2} & \cos\frac{\Omega_p}{2} \end{pmatrix} \begin{pmatrix} u(p,+\frac{1}{2}) \\ u(p,-\frac{1}{2}) \end{pmatrix} \equiv R \begin{pmatrix} u_+(p) \\ u_-(p) \end{pmatrix} \tag{4.5.9}$$

Note, that due to Eq. (4.5.5), Eq. (4.5.8) has to be transposed to write it in matrix notation. Furthermore, this is no violation of Wigner's statement [Wig39] that the Lorentz group has no true unitary representation in a finite number of dimensions. We are merely using a momentum dependent, local unitary rotation of the spin components.

With Eq. (4.5.9), the transformation of ρ is

$$\begin{aligned}\rho'(p_\Lambda, \boldsymbol{\alpha}) &= \sum_{ij} c_{ij}\; u'_i(p_\Lambda)\bar{u}'_j(p_\Lambda) \\ &= \sum_{ijkm} R^{\mathrm{T}}_{ki} c_{ij} R_{jm}\; u_k \bar{u}_m \\ &= \sum_{ijkm} \frac{1}{2}\left(\delta_{ij} + \boldsymbol{\alpha}\underbrace{R^{\mathrm{T}}_{ki}\boldsymbol{\sigma}_{ij}R_{jm}}_{\boldsymbol{\sigma}'_{km}}\right) u_k \bar{u}_m \,. \end{aligned} \tag{4.5.10}$$

This transformation of $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma}'$ can also be performed by rotating $\boldsymbol{\alpha}$. More generally, it is an easy exercise of matrix multiplication to show that

$$\boldsymbol{\sigma}\boldsymbol{\alpha}' \equiv \boldsymbol{\sigma}\left(\mathrm{e}^{-\mathrm{i}\,\boldsymbol{\theta}\cdot\boldsymbol{J}}\boldsymbol{\alpha}\right) = \boldsymbol{\alpha}\left(\mathrm{e}^{-\mathrm{i}\,\boldsymbol{\theta}\cdot\frac{\boldsymbol{\sigma}}{2}}\;\boldsymbol{\sigma}\;\mathrm{e}^{\mathrm{i}\,\boldsymbol{\theta}\cdot\frac{\boldsymbol{\sigma}}{2}}\right) \equiv \boldsymbol{\alpha}\boldsymbol{\sigma}' \,, \tag{4.5.11}$$

whereby $\boldsymbol{\theta} \in \mathbb{R}^3$ are the rotation angles and $\boldsymbol{J}$ and $\boldsymbol{\sigma}/2$ are the generators of $SO(3)$ and $SU(2)$ rotations, respectively, and is commonly known as the $SO(3)$-$SU(2)$ correspondence. This means that the Wigner rotation of the spin states induced by a Lorentz boost perpendicular to the momentum can be pulled back to a rotation of the three vector of the density matrix. Furthermore, we can note that $\rho(p, \boldsymbol{\alpha}) = U(L(p))\rho(k, \boldsymbol{\alpha})U^\dagger(L(p)) = \rho(k, \boldsymbol{\alpha})$ as the boost parallel to the momentum direction that brings p to the rest frame vector k is independent of the spin state.

Finally, we should also note the transformation behaviour under rotations and translations. Rotations are just the simplification of the above whereby now we have a truly global, unitary rotation of the spin components. Translations are given by the unitary operator $U(\mathbb{1}, a) = \exp\{\mathrm{i}a_\mu P^\mu\}$ leading to two phases with opposite signs for ρ'. Therefore, ρ is invariant under translations.

5. Conclusion

In this thesis, we have touched various aspects of the computation of multi-jet cross sections. Hereby, we aimed to complement the excellent abilities of the event generator Whizard to compute cross sections involving some heavy particles by an approach which is especially suited for many light particles, as it occurs in Quantum Chromodynamics (QCD). We have shown that a high-performance Virtual Machine (VM) in `Fortran95` is a very powerful alternative to the traditional approach of compiling the matrix element of every process separately and yields many benefits. Firstly, we eliminate the, eventually time consuming, necessity to compile completely. This allows to compute amplitudes with eight external gluons with O'Mega, which fails to compile due to lacking memory on a common desktop environment. The VM is also handy to check a lot of processes quickly. Even the computation time of the matrix element per phase space point has been reduced for complex amplitudes compared to the compiled code with an improved memory layout. The VM also allows to parallelize the computation easily and effectively. Exploiting the recursive nature of the computation, we have identified hereby the necessary synchronization points. These are represented in the bytecode, HepBC, in an abstract way independent of the implementation of the interpreter. With the scaling with multiple cores, we have demonstrated that 75 to 95 % of the whole computation is parallelizable, which paves the way to a potential implementation on a `GPU`. Additionally, the HepBC is a platform independent format, which can be reused on clusters or environments where a full O'Mega installation would be tedious. The HepBC could also be extended to other event generators, which involve the computation of a matrix element, as it is a fairly simple and general concept. Furthermore, we have shown that the reduction of four vertices with auxiliary fields reduces the number of vertices in a *recursively* computed amplitude for 11 or more external particles while it is higher for less particles. Since the computation time for three and four vertices is not the same, this does not necessarily reflect a break-even point at which it is worth to use auxiliary fields. It rather has to be tested at which point O'Mega can profit from the slower growth in the number of vertices.

Since a flat momentum phase space generator, i.e. Rambo, causes high variance in the integration of matrix elements, we have implemented Sarge, which samples points with a density close to the leading divergencies of QCD. For this, we have recalculated the explicit weights, the inverse of the density, of the basic Sarge algorithm as

well as its extension including initial momenta analytically using the Unitary Algorithm Formalism (UAF) and find that it differs in global factors of 2, π and center-of-mass energy $\sqrt{s}$ from the published results [HK00b; HKD00]. Though they are not important for the event generation of fixed $\sqrt{s}$ processes, only our formulae give the correct result for the phase space volume and hence the absolute value of the cross section. We should mention that the `FORTRAN77` implementation of A. van Hameren et al. uses the correct weight. Overall, we have verified that SARGE is indeed a very good mapping of the soft and collinear divergencies occurring in QCD and a major improvement over RAMBO. The reduction of the volume for the random variables representing quotients of scalar products from the hypercube to a smaller polytope increases the acceptance rate to generate a valid phase space point by an order of magnitude for five outgoing momenta and grows like $\sim 2^{2n}$ for n external particles. Including initial momenta in the phase space density requires a modified algorithm, whereby the variance of the integration depends greatly on the way how the permutations are performed and the weight is computed. We have presented an algorithm, strongly inspired by Ref. [HK00b], which performs very well and reduces the number of points needed to integrate an amplitude to a given precision by more than an order of magnitude for most processes compared to the basic version of SARGE.

In the last chapter, we have compared different types of possible ways to perform the helicity sum. The inclusion of the helicity degrees of freedom as additional dimension in the Monte Carlo (MC) integration has shown to be very fruitful. Hereby, we have obtained the lowest overall variance when the warm up phase for the discrete importance sampling was long enough. With unit weight, however, superpositions of helicities, i.e. continuous phases, seem to result in a lower variance compared to the discrete samplings. We have generalized the known parametrization from the literature [DKP98] and given the conditions that must be fulfilled to obtain the result of the usual, deterministic helicity sum. Furthermore, we can identify this parametrization for massive spin 1/2 particles as integration path along the equator of the Bloch sphere of the external particle. Each point on the Bloch sphere corresponds hereby to a spin in a certain direction, measured in the rest frame of each particle. Finally, we considered the transformation of spin density matrices under Lorentz transformations. Though helicity MC is way more efficient than the full sum in our setup, we note that the treatment of spinors in the Weyl-van-der-Werden formalism and regarding different chiralities as different wave functions removes redundancies between amplitudes where only some helicities are flipped in a recursive computation. Apparently, this seems to be preferable to simplest helicity sampling for up to 11 external particles [GH08] and would be a possible extension to O'MEGA. Overall, we can

conjecture that the phase space integration will be further improved when adaptive importance sampling methods like VEGAS will be used for the random variables of SARGE as well as the helicity phases. As it is advisable to generate the grids in a warm up phase and then integrate with constant grids to ensure convergence, this warm up phase can also be used to chose the dominant permutation channels for SARGE. It is also open to explore whether discrete importance sampling of helicities or VEGAS grids for continuous helicity phases are superior.

Zusammenfassung

In dieser Arbeit wurden verschiedene Aspekte der Berechnung von Multi-Jet Wirkungsquerschnitten behandelt. Hierbei haben wir versucht die hervorragenden Fähigkeiten des Eventgenerators WHIZARD Wirkungsquerschnitte, bei denen einige schwere Teilchen beteiligt sind, zu berechnen um einen Ansatz zu erweitern, der besonders gut für viele leichte Teilchen geeignet ist, wie es in der Quantenchromodynamik (QCD) auftritt. Wir haben gezeigt, dass eine leistungsfähige Virtual Machine (VM) in `Fortran95` eine sehr mächtige Alternative zur traditionellen Methode ist, bei der das Matrixelement jedes Prozesses seperat kompiliert wird, und sich viele Vorteile ergeben. Die eventuell zeitaufwändige Notwendigkeit zu Kompilieren wird hiermit komplett entfernt. Dies erlaubt Amplitude mit acht externen Gluonen mit O'MEGA zu berechnen, was auf einer üblichen Desktopumgebung aufgrund von fehlendem Arbeitsspeicher nicht kompilierbar ist. Die VM ist auch praktisch um viele Prozesse schnell zu überprüfen. Selbst die Zeit zur Berechnung des Matrixelementes pro Phasenraumpunkt wurde mit einem verbesserten Memorylayout für komplexe Amplituden verbessert. Desweiteren erlaubt die VM die Berechnung einfach und effektiv zu parallelisieren. Wir haben die hierbei notwendigen Synchronisierungspunkte identifiziert indem wir die rekursive Natur der Rechnung ausgenutzt haben. Diese Punkte sind im Bytecode, HEPBC, auf abstrakte Weise unabhängig von der Implementierung des Interpreters repräsentiert. Anhand der Skalierung mit mehreren Kernen haben wir demonstriert, dass 75 bis 95 % der gesamten Rechnung parallelisierbar sind, was den Weg für eine mögliche `GPU` Implementierung ebnet. Ferner ist HEPBC ein plattformunabhängiges Format, welches auf Clustern oder in Umgebungen, in denen eine volle O'MEGA Installation mühsam ist, wiederverwendet werden kann. Der HEPBC könnte auch für andere Eventgeneratoren, bei denen ein Matrixelement berechnet wird, erweitert werden, da es ein recht einfaches und allgemeines Konzept ist. Desweiteren haben wir gezeigt, dass die Reduktion von Vierervertices mit Hilfsfeldern die Anzahl der Vertices in einer *rekursiv* berechneten Amplitude für elf oder mehr externe Teilchen reduziert, während sie für weniger Teilchen höher ist. Da die Berechnungszeit für Dreier- und Vierervertices nicht identisch ist, stellt dies nicht unbedingt einen Break-Even Punkt dar, ab welchem es sich lohnt Hilfsfelder zu verwenden. Stattdessen muss getestet werden ab welchem Punkt O'MEGA von dem langsameren Wachstum der Anzahl an Vertices profitieren kann.

Da ein flacher Impulsphasenraumgenerator, d.h. RAMBO, für eine hohe Varianz in

der Integration der Matrixelemente sorgt, haben wir SARGE implementiert, welcher Punkte mit einer Dichte erzeugt, die nahe an den führenden Divergenzen von QCD liegt. Dafür haben wir die expliziten Gewichte, also das Inverse der Dichte, sowohl für den grundlegenden als auch den erweiterten, welcher die einlaufenden Impulse mitberücksichtigt, SARGE-Algorithmus analytisch mit Hilfe des Unitary Algorithm Formalism (UAF) berechnet und erhalten ein Ergebnis, welches in globalen Faktoren von 2, π und Schwerpunktsenergie $\sqrt{s}$ von den veröffentlichten Resultaten abweicht [HK00b; HKD00]. Obwohl dies für die Erzeugung von Events bei festem $\sqrt{s}$ irrelevant ist, ergeben nur unsere Formeln das korrekte Phasenraumvolumen und somit absoluten Wirkungsquerschnitt. Wir sollten erwähnen, dass die `FORTRAN77` Implementierung von A. van Hameren et al. das korrekte Gewicht verwendet. Insgesamt haben wir verifiziert, dass SARGE in der Tat eine sehr gute Abbildung der weichen und kollinearen Divergenzen, die in QCD auftreten, darstellt und eine enorme Verbesserung gegenüber RAMBO ist. Die Reduktion des Volumens für Zufallsvariablen, welche Quotienten von Skalarprodukten darstellen, von einem Hyperkubus zu einem kleineren Polytop erhöht die Akzeptanzrate zur Erzeugung eines gültigen Phasenraumpunkts um eine Größenordnung für fünf ausgehende Impulse und wächst wie 2^{2n} für n externe Teilchen. Die Berücksichtigung der einlaufenden Impulse in der Phasenraumdichte erfordert einen modifizierten Algorithmus, wobei die Varianz der Integration in hohem Maße davon abhängt auf welche Art die Permutationen durchgeführt werden und wie das Gewicht berechnet wird. Wir haben einen Algorithmus präsentiert, deutlich inspiriert von Ref. [HK00b], welcher sehr gut funktioniert und die Anzahl an Punkte, die notwendig sind um eine Amplitude bis zu einer gegebenen Präzision zu integrieren, für die meisten Prozesse um mehr als eine Größenordnung reduziert im Vergleich zur grundlegenden Version von SARGE.

Im letzten Kapitel haben wir verschiedene Arten von möglichen Wegen die Helizitätssumme durchzuführen verglichen. Die Hinzunahme der Helizitätsfreiheitsgrade in der Monte Carlo (MC) Integration als weitere Dimension hat sich als sehr erfolgreich herausgestellt. Hierbei haben wir insgesamt die niedrigste Varianz erhalten, wenn die Aufwärmphase für das diskrete Importance Sampling lang genug war. Bei konstanten Gewichten scheinen Superpositionen von Helizitäten, also kontinuierliche Phasen, allerdings zu niedrigerer Varianz im Vergleich zu diskreten Phasen zu führen. Wir haben die bekannte Parametrisierung der Literatur [DKP98] verallgemeinert und die Bedingungen angegeben, die erfüllt werden müssen, um das Result der gewöhnlichen, deterministischen Helizitätssumme zu erhalten. Desweiteren können wir diese Parametrisierung für massive Spin 1/2 Teilchen als Integrationspfad entlang des Äquators der Blochsphäre des äußeren Teilchens identifizieren. Jeder Punkt auf der

Blochsphäre entspricht hierbei einem Spin in eine gewisse Richtung, gemessen im Ruhesystem des jeweiligen Teilchens. Schließlich haben wir die Transformation von Spindichtematrizen unter Lorentztransformationen betrachtet. Obwohl Helizitäts-MC in unserem Fall viel effizienter als die volle Summe ist, merken wir an, dass die Behandlung von Spinoren im Weyl-van-der-Werden Formalismus und von verschiedenen Chiralitäten als verschiedene Wellenfunktionen Redundanzen zwischen Amplituden, in denen sich nur einige Helizitäten ändern, in einer rekursiven Berechnung entfernt. Dies scheint dem einfachsten Helizitäts-MC für bis zu elf externe Teilchen überlegen zu sein [GH08] und wäre eine mögliche Erweiterung für O'Mega. Insgesamt können wir vermuten, dass die Phasenraumintegration sich weiter verbessern wird, wenn adaptive Importance Sampling Methoden wie Vegas für die Zufallsvariablen von Sarge und den Helizitätsphasen verwendet werden. Da es ratsam ist, die Gitter in einer Aufwärmphase zu generieren und anschließend mit festen Gittern zu integrieren, um Konvergenz zu garantieren, kann diese Aufwärmphase auch dazu verwendet werden, um die dominanten Permutationskanäle für Sarge zu wählen. Es bleibt ebenfalls offen, ob für die Helizitätsphasen diskretes Importance Sampling oder Vegas Gitter überlegen sind.

A. Recursive mean and variance

The definition of the sample mean $\bar{x}_n$ and unbiased sample variance σ_n^2 from n points $\{x_i\}_{i=1,\dots n}$ is

$$\begin{aligned} \bar{x}_n &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \sigma_n^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2 \,. \end{aligned} \tag{A.0.1}$$

For the recursive mean follows straightforwardly

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n-1} x_i + \frac{x_n}{n} = \frac{n-1}{n}\bar{x}_{n-1} + \frac{x_n}{n} \tag{A.0.2}$$

or as implemented

$$\texttt{mean\_new} = \texttt{mean} + \frac{\texttt{x} - \texttt{mean}}{\texttt{n}} \,. \tag{A.0.3}$$

For the variance, inserting a zero allows to compare with the old mean

$$\begin{aligned} \sigma_n^2 &= \frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - \bar{x}_{n-1}) + (\bar{x}_{n-1} - \bar{x}_n)\right)^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n-1}(x_i - \bar{x}_{n-1})^2 + \frac{1}{n-1}\Big\{(x_n - \bar{x}_{n-1})^2 + 2(\bar{x}_{n-1} - \bar{x}_n) \\ &\qquad\qquad \sum_{i=1}^{n}(x_i - \bar{x}_{n-1}) + n(\bar{x}_{n-1} - \bar{x}_n)^2\Big\} \\ &= \frac{n-2}{n-1}\sigma_{n-1}^2 + \frac{1}{n-1}\Big\{\big(n(\bar{x}_n - \bar{x}_{n-1})\big)^2 - (2n-n)(\bar{x}_n - \bar{x}_{n-1})^2\Big\} \,, \end{aligned} \tag{A.0.4}$$

where we have used Eq. (A.0.2) in the last step. Finally, we obtain

$$\sigma_n^2 = \Big(1 - \frac{1}{n-1}\Big)\sigma_{n-1}^2 + n(\bar{x}_n - \bar{x}_{n-1})^2 \tag{A.0.5}$$

or in the language of Eq. (A.0.3)

$$\texttt{varsq} = \texttt{varsq} - \frac{\texttt{varsq}}{\texttt{n}-\texttt{1}} + \texttt{n}\,(\texttt{mean\_new} - \texttt{mean})^2 \,. \tag{A.0.6}$$

B. Helicity speed up tables

Table B.1. Execution times for various $2 \to 2, 3$ processes until the estimated error has reached 0.2 % and 1 % averaged over 15 seeds. RAMBO was used as momentum generator with $\sqrt{s} = 100\,\text{GeV}$ and relatively generous cuts, i.e. $\tau = 0.1$, such that the spiky convergence of the algorithm is reduced. The different modes are encoded as follows: s = full sum, DI = *discrete* HMC with *importance* sampling, {ED, EC} = *exponential* parametrization with *discrete* and *continuous* sampling, respectively, {TD, TD2, TC} = *trigonometrical* parametrization with $\theta \in \{0, \pi/2\}$, $\theta \in \{1/4\,\pi, 3/4\,\pi\}$, $\theta \in [0, \pi]$ and sphr = *spherical* parametrization as described in the text. The average errors follow from Gaussian error propagation whilst taking into account the statistical spread over different processes.

process	t_s/t_s	t_DI/t_s	t_ED/t_s	t_EC/t_s	t_TD/t_s	t_TD2/t_s	t_TC/t_s	t_sphr/t_s
$qq \to qq$	1.00(11)	0.34(6)	1.09(9)	0.79(9)	1.34(12)	0.95(8)	0.60(8)	0.65(6)
$qq' \to qq'$	1.00(7)	0.19(3)	0.97(8)	0.70(7)	1.16(7)	0.81(6)	0.55(5)	0.59(1)
$q\bar{q} \to q'\bar{q}'$	1.00(4)	0.36(4)	0.90(5)	0.90(7)	0.73(4)	0.74(5)	0.75(4)	0.95(3)
$q\bar{q} \to q\bar{q}$	1.00(9)	0.17(2)	0.86(7)	0.63(3)	1.01(7)	0.75(5)	0.51(6)	0.56(4)
$qg \to qg$	1.00(6)	0.10()	0.70(7)	0.33(4)	0.47(2)	0.33(1)	0.22()	0.27()
$q\bar{q} \to gg$	1.00(1)	0.24(1)	0.46(1)	0.47(1)	0.41(1)	0.42(1)	0.42(1)	0.57(6)
$gg \to gg$	1.00(6)	0.21(2)	0.86(4)	0.34(2)	0.87(3)	0.42(3)	0.33(3)	0.41(2)
average	1.00(3)	0.23(9)	0.84(19)	0.59(21)	0.86(32)	0.63(22)	0.48(16)	0.57(20)

process	t_s/t_s	t_DI/t_s	t_ED/t_s	t_EC/t_s	t_TD/t_s	t_TD2/t_s	t_TC/t_s	t_sphr/t_s
$qq \to qqg$	1.00(13)	0.10(1)	0.23(3)	0.26(5)	0.28(4)	0.40(5)	0.20(3)	0.19(5)
$qq' \to qq'g$	1.00(15)	0.09(1)	0.33(4)	0.39(5)	0.43(4)	0.54(8)	0.28(4)	0.27(3)
$q\bar{q} \to q\bar{q}g$	1.00(18)	0.08(1)	0.22(3)	0.22(4)	0.28(3)	0.36(5)	0.21(4)	0.20(3)
$qg \to qgg$	1.00(18)	0.12(2)	0.33(3)	0.22(3)	0.32(3)	0.37(5)	0.16(4)	0.15(2)
$q\bar{q} \to ggg$	1.00(12)	0.26(2)	0.17(2)	0.12(1)	0.60(6)	0.16(2)	0.21(3)	0.26(4)
$gg \to ggg$	1.00(18)	0.36(7)	0.28(2)	0.15(2)	0.27(3)	0.22(4)	0.15(2)	0.15(2)
average	1.00(7)	0.17(11)	0.26(6)	0.23(9)	0.36(12)	0.34(13)	0.20(5)	0.21(5)

Table B.1 (cont.) Execution times for various $2 \to 4, 5$ processes until the estimated error has reached 3 % averaged over 15 seeds. The discrete importance sampling fails to produce good weights for the full gluonic amplitudes as it was only run with a fixed number of warm-up runs. Over one magnitude in computing time can be saved by using HMC for five external particles.

process	t_s/t_s	t_{DI}/t_s	t_{ED}/t_s	t_{EC}/t_s	t_{TD}/t_s	t_{TD2}/t_s	t_{TC}/t_s	t_{sphr}/t_s
$qq' \to qq'gg$	1.00(59)	0.06(1)	0.07(3)	0.09(5)	0.14(5)	0.17(12)	0.09(6)	0.07(4)
$q\bar{q} \to q'\bar{q}'gg$	1.00(2)	0.31(1)	0.09()	0.08()	0.15(4)	0.07()	0.08()	0.09(1)
$q\bar{q} \to q\bar{q}gg$	1.00(43)	0.07(3)	0.08(6)	0.07(3)	0.14(5)	0.21(16)	0.08(6)	0.07(4)
$qg \to qggg$	1.00(43)	0.15(7)	0.14(5)	0.09(4)	0.18(8)	0.12(6)	0.07(5)	0.09(6)
$q\bar{q} \to gggg$	1.00(2)	0.34(2)	0.08()	0.07()	0.10(2)	0.07()	0.07(1)	0.08()
$gg \to gggg$	1.00(32)	0.84(48)	0.11(2)	0.05(2)	0.14(5)	0.15(23)	0.05(2)	0.05(1)
average	1.00(15)	0.30(28)	0.09(3)	0.07(2)	0.14(3)	0.13(7)	0.07(2)	0.07(2)

process	t_s/t_s	t_{DI}/t_s	t_{ED}/t_s	t_{EC}/t_s	t_{TD}/t_s	t_{TD2}/t_s	t_{TC}/t_s	t_{sphr}/t_s
$qg \to qq'\bar{q}'gg$	1.00(67)	0.17(5)	0.06(7)	0.05(7)	0.22(21)	0.04(3)	0.06(11)	0.05(8)
$qq' \to qq'q''\bar{q}''g$	1.00(163)	0.04(2)	0.02(2)	0.03(2)	0.09(5)	0.05(5)	0.03(4)	0.02(2)
$qq \to qqggg$	1.00(49)	0.15(7)	0.04(3)	0.04(5)	0.08(4)	0.11(15)	0.07(8)	0.05(3)
$qq' \to qq'ggg$	1.00(86)	0.15(7)	0.03(2)	0.06(10)	0.08(3)	0.11(10)	0.05(6)	0.03(2)
$q\bar{q} \to q\bar{q}ggg$	1.00(43)	0.14(8)	0.06(4)	0.03(2)	0.17(21)	0.14(12)	0.05(3)	0.06(7)
$q\bar{q} \to q'\bar{q}'ggg$	1.00(17)	0.53(10)	0.04()	0.03()	0.09(2)	0.03()	0.04(1)	0.04()
$q\bar{q} \to gggg$	1.00(2)	0.66(12)	0.04(1)	0.03()	0.05(1)	0.03()	0.03()	0.04(1)
$gg \to gggg$	1.00(99)	1.28(121)	0.07(4)	0.03(3)	0.09(4)	0.05(9)	0.02(1)	0.04(4)
average	1.00(29)	0.39(42)	0.04(2)	0.04(2)	0.11(7)	0.07(5)	0.04(2)	0.04(2)

Acknowledgements

It is a pleasant task to thank all those, who have helped me to write this thesis. First of all, I want to thank Prof. Dr. Thorsten Ohl for giving me the opportunity to work on an interesting topic, the various discussions and the useful comments about the manuscript. I also enjoyed the sometimes really enlightening discussions in our small group about my problems as well as those of others. Overall, the atmosphere at the department *Theoretische Physik II* was very pleasant and encouraging. Furthermore, I want to thank André van Hameren for providing me with his modified SARGE code, which allowed me to find a bug in my implementation. Special thanks go to my girl friend for spell checking, taking care of me, when I forget to do so, and so much more. Moreover, I want to thank the FOKUS group for great physics discussions and countless distractions throughout my studies. Last but not least I want to thank my family for supporting me although they are not so sure what exactly I am doing.

This might also be the place to thank for all the open source software that we often take for granted though it is the result of mostly unpaid, hard work of great programmers who try to find the best solution. Without their efforts this thesis would certainly not have been possible in this form. The following should be seen as incomplete list of most of the software that was involved in the creation of this work and is available for free over the internet.

- **LaTeX**: Beautiful typesetting while automating numbering and referencing.
- **Matplotlib**: Scriptable, high-quality vector plots with LaTeX support.
- **Python**: The easy language makes solving day-to-day tasks fun.
- **OCaml**: Incredible power to do things one could hardly imagine.
- **gfortran**: Free `Fortran` compiler with pretty good error messages.
- **FeynMP**: Amazing Feynman diagrams almost out-of-the-box and LaTeX support.
- **Noweb**: Simple, literate programming that will hopefully be used more someday.
- **VIM**: Editor and IDE for all languages actively developed since 1991.
- **Linux**: Open and powerful OS without inbuilt supervision.

Bibliography

[Abe+08] S. Abe, T. Ebihara, S. Enomoto, K. Furuno, et al. Precision Measurement of Neutrino Oscillation Parameters with KamLAND. In *Physical Review Letters*, **100**:22 (June 2008), p. 221803. arXiv:0801.4589v3

[AM02] P. Alsing and G. Milburn. Lorentz invariance of entanglement. In *Quantum Information & Computation*, **2**:6 (2002), pp. 487–512. arXiv:quant-ph/0203051v1 [quant-ph]

[AHM11] J. Alwall, M. Herquet, and F. Maltoni. MadGraph 5: going beyond. In *Journal of High Energy Physics*, 1 (2011), p. 128. arXiv:1106.0522v1

[Amd67] G. Amdahl. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. In *AFIPS Conference Proceedings*, **30**: (1967), pp. 483–485

[An+12] F. P. An, J. Z. Bai, A. B. Balantekin, H. R. Band, et al. Observation of Electron-Antineutrino Disappearance at Daya Bay. In *Physical Review Letters*, **108**:17 (Apr. 2012), p. 171803. arXiv:arXiv:1203.1669v2

[Aoy+07] T. Aoyama, M. Hayakawa, T. Kinoshita, and M. Nio. Revised value of the eighth-order QED contribution to the anomalous magnetic moment of the electron. In *Physical Review D*, **77**:5 (Dec. 2007), p. 54. arXiv:0712.2607

[Ark+02] N. Arkani-Hamed, A. Cohen, E. Katz, and A. Nelson. The Littlest Higgs. In *Journal of High Energy Physics*, **2002**:07 (July 2002), pp. 034–034. arXiv:0206021v2 [arXiv:hep-ph]

[ATL13] C. ATLAS. Evidence for the spin-0 nature of the Higgs boson using ATLAS data. In *Physics Letters B*, **726**:1-3 (Oct. 2013), pp. 120–144. arXiv:1307.1432v1

[ATL12] C. ATLAS. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. In *Physics Letters B*, **716**:1 (Sept. 2012), pp. 1–29. arXiv:1207.7214v2

[BO12] F. Bach and T. Ohl. Anomalous top couplings at hadron colliders revisited. In *Physical Review D*, **86**:11 (Dec. 2012), p. 114026. arXiv:1209.4564

[Bäh+08] M. Bähr, S. Gieseke, M. A. Gigg, D. Grellscheid, et al. Herwig++ physics and manual. In *The European Physical Journal C*, **58**:4 (Nov. 2008), pp. 639–707. arXiv:arXiv:0803.0883v3

[BBS91] K. Begeman, A. Broeils, and R. Sanders. Extended rotation curves of spiral galaxies - Dark haloes and modified dynamics. In *Monthly Notices of the Royal Astronomical Society*, **249**: (1991), pp. 523–537

[Ber+12] J. Beringer, J.-F. Arguin, R. M. Barnett, K. Copic, et al. Review of Particle Physics. In *Physical Review D*, **86**:1 (July 2012), p. 010001

[BDJ01] M. Böhm, A. Denner, and H. Joos. *Gauge Theories of the Strong and Electroweak Interaction*. Vieweg+Teubner Verlag, 2001

[Boo+04] E. Boos, V. Bunichev, M. Dubinin, L. Dudko, et al. CompHEP 4.4—automatic computations from Lagrangians to events. In *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **534**:1-2 (Nov. 2004), pp. 250–259. arXiv:hep-ph/0403113v1 [hep-ph]

[BW86] W. Buchmüller and D. Wyler. Effective lagrangian analysis of new interactions and flavour conservation. In *Nuclear Physics B*, **268**:3-4 (May 1986), pp. 621–653

[Cho13] B. Chokoufe Nejad. *https://github.com/bijancn/ovm: OVM implementation in Fortran*. 2013

[CD09] N. Christensen and C. Duhr. FeynRules – Feynman rules made easy. In *Computer Physics Communications*, **180**:9 (Sept. 2009), pp. 1614–1641. arXiv:arXiv:0806.4194v1

[Clo+06] D. Clowe, M. Bradač, A. H. Gonzalez, M. Markevitch, et al. A Direct Empirical Proof of the Existence of Dark Matter. In *The Astrophysical Journal*, **648**:2 (Sept. 2006), pp. L109–L113. arXiv:0608407v1 [arXiv:astro-ph]

[CMS12] C. CMS. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. In *Physics Letters B*, **716**:1 (Sept. 2012), pp. 30–61. arXiv:1207.7235v2

[DN12] N. Davidsona and G. Nanavae. Universal Interface of TAUOLA Technical and Physics Documentation. In *Comput.Phys.Commun.*, **183**:February (Feb. 2012), pp. 821–843. arXiv:1002.0543v1

[Dit98] S. Dittmaier. Weyl–van der Waerden formalism for helicity amplitudes of massive particles. In *Physical Review D*, May (1998). arXiv:9805445v1 [arXiv:hep-ph]

[DKP02] P. Draggiotis, R. Kleiss, and C. Papadopoulos. Multi-jet production in hadron collisions. In *The European Physical Journal C - Particles and Fields*, **24**:3 (July 2002), pp. 447–458. arXiv:0202201v1 [hep-ph]

[DKP98] P. Draggiotis, R. Kleiss, and C. Papadopoulos. On the computation of multigluon amplitudes. In *Physics Letters B*, October (1998), pp. 157–164. arXiv:hep-ph/9807207 [hep-ph]

[EGK08] R. Ellis, W. Giele, and Z. Kunszt. A numerical unitarity formalism for evaluating one-loop amplitudes. In *Journal of High Energy Physics*, **2008**:03 (Mar. 2008), pp. 003–003. arXiv:0708.2398

[ESW96] R. Ellis, W. Stirling, and B. Webber. *QCD and Collider Physics*. Cambridge University Press, 1996

[EB08] L. Evans and P. Bryant. LHC Machine. In *Journal of Instrumentation*, **3**:08 (Aug. 2008), S08001–S08001

[Gla61] S. L. Glashow. Partial-symmetries of weak interactions. In *Nuclear Physics*, **22**:4 (Feb. 1961), pp. 579–588

[GH08] T. Gleisberg and S. Höche. Comix, a new matrix element generator. In *Journal of High Energy Physics*, **2008**:12 (Dec. 2008), p. 039. arXiv:0808.3674v2

[Gle+09] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, et al. Event generation with SHERPA 1.1. In *Journal of High Energy Physics*, **2009**:02 (Feb. 2009), pp. 007–007. arXiv:0811.4622

[Hag+11] K. Hagiwara, R. Liao, A. Martin, D. Nomura, and T. Teubner. $(g-2)_\mu$ and $\alpha(M_Z^2)$ re-evaluated using new precise data. In *Journal of Physics G: Nuclear and Particle Physics*, **38**:8 (Aug. 2011), p. 085003. arXiv:1105.3149v2

[Hag+12] K. Hagiwara, T. Li, K. Mawatari, and J. Nakamura. TauDecay: a library to simulate polarized tau decays via FeynRules and MadGraph5. In (Dec. 2012). arXiv:1212.6247

[HP99] T. Hahn and M. Pérez-Victoria. Automated one-loop calculations in four and D dimensions. In *Computer Physics Communications*, **118**:2-3 (May 1999), pp. 153–165. arXiv:hep-ph/9807565 [hep-ph]

[HK00a] A. van Hameren and R. Kleiss. A fast algorithm for generating a uniform distribution inside a high-dimensional polytope. In *Computer Physics Communications*, **133**:1 (Dec. 2000), pp. 1–5. arXiv:physics/0003078v3 [physics]

[HKD00] A. van Hameren, R. Kleiss, and P. D. Draggiotis. SARGE: An algorithm for generating QCD-antennas. In *Physics Letters B*, **483**:1-3 (June 2000), pp. 124–130. arXiv:0004047v2 [arXiv:hep-ph]

[HK00b] A. van Hameren and R. Kleiss. Generating QCD antennas. In *The European Physical Journal C*, **17**:4 (2000), pp. 611–621. arXiv:0008068v1 [arXiv:hep-ph]

[HP02] A. van Hameren and C. Papadopoulos. A hierarchical phase space generator for QCD antenna structures. In *The European Physical Journal C*, (Apr. 2002), p. 21. arXiv:0204055 [hep-ph]

[Her02] M. Hermanns. *Parallel Programming in Fortran 95 using OpenMP*. Tech. rep. Madrid: School of Aeronautical Engineering, 2002

[KOR11] W. Kilian, T. Ohl, and J. Reuter. WHIZARD—simulating multi-particle processes at LHC and ILC. In *The European Physical Journal C*, **71**:9 (Sept. 2011), p. 1742. arXiv:0708.4233v2

[Kil+12] W. Kilian, T. Ohl, J. Reuter, and C. Speckner. QCD in the color-flow representation. In *Journal of High Energy Physics*, **2012**:10 (Oct. 2012), p. 22. arXiv:1206.3700v2

[Kle00] R. Kleiss. "Phase Space Monte Carlo". In *Particle Production Spanning MeV and TeV Energies (NATO Science Series C: Mathematical and Physical Sciences Vol. 554)*. Ed. by W. Kittel, P. Mulders, and O. Scholten. Springer, 2000, p. 431

[KP94] R. Kleiss and R. Pittau. Weight optimization in multichannel Monte Carlo. In *Computer Physics Communications*, **83**:2-3 (Dec. 1994), pp. 141–146. arXiv:9405257v2 [arXiv:hep-ph]

[KSE86] R. Kleiss, W. Stirling, and S. Ellis. A new Monte Carlo treatment of multiparticle phase space at high energies. In *Computer Physics Communications*, **40**: (1986), pp. 359–373

[Knu81] D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms (2nd Edition)*. 2nd. Addison-Wesley Pub (Sd), 1981, p. 704

[Knu11] D. E. Knuth. *The Art of Computer Programming, Volume 4: Combinatorial Algorithms*. 2011, p. 883

[Kui91] J. Kuijf. *Multiparton production at hadron colliders*. PhD thesis. University of Leiden, 1991

[LSZ55] H. Lehmann, K. Symanzik, and W. Zimmermann. Zur Formulierung quantisierter Feldtheorien. In *Il Nuovo Cimento*, **1**:1 (Jan. 1955), pp. 205–225

[Lep78] G. Lepage. A new algorithm for adaptive multidimensional integration. In *Journal of Computational Physics*, **27**:2 (1978), pp. 192–203

[MOR01] M. Moretti, T. Ohl, and J. Reuter. O'Mega: An Optimizing Matrix Element Generator. In *IKDA-2001-06, LC-TOOL-2001-040*, (Feb. 2001), p. 29. arXiv:hep-ph/0102195 [hep-ph]

[Ohl99a] T. Ohl. Electroweak Gauge Bosons at Future Electron-Positron Colliders. In *arXiv preprint hep-ph/9911437*, June (1999). arXiv:hep-ph/9911437v1 [hep-ph]

[Ohl99b] T. Ohl. Vegas revisited: Adaptive Monte Carlo integration beyond factorization. In *Computer Physics Communications*, **120**:1 (July 1999), pp. 13–19. arXiv:9806432v1 [hep-ph]

[Ohl+12] T. Ohl, J. Reuter, W. Kilian, C. Speckner, and C. Schwinn. *O'Mega: Optimal Monte-Carlo Event Generation Amplitudes - Manual*. 2012

[Oor12] G. van den Oord. *Recursion, Monte Carlo and vector boson scattering at Hadron Colliders*. PhD thesis. Radboud Universiteit Nijmegen, 2012

[PT86] S. Parke and T. Taylor. Amplitude for n-Gluon Scattering. In *Physical Review Letters*, **56**:23 (June 1986), pp. 2459–2460

[PST02] A. Peres, P. Scudo, and D. Terno. Quantum Entropy and Special Relativity. In *Physical Review Letters*, **88**:23 (May 2002), p. 230402. arXiv:quant-ph/0203033 [quant-ph]

[Pla13] C. Planck. Planck 2013 results. I. Overview of products and scientific results. In *arXiv preprint*, (2013). arXiv:arXiv:1303.5062v1

[Ram94] N. Ramsey. Literate programming simplified. In *IEEE Software*, **11**:5 (Sept. 1994), pp. 97–105

[Sal68] A. Salam. *Weak and Electromagnetic Interactions*. Elementary. Nobel Symposium 8, Aspenasgarden, 1968, p. 367

[Sch07a] F. Scheck. *Theoretische Physik 4: Quantisierte Felder*. Springer, 2007

[Sch07b] F. Schwabl. *Quantenmechanik (QM I): Eine Einführung (Springer-Lehrbuch) (German Edition)*. Springer, 2007

[SMS08] T. Sjöstrand, S. Mrenna, and P. Skands. A brief introduction to PYTHIA 8.1. In *Computer Physics Communications*, **178**:11 (June 2008), pp. 852–867. arXiv:arXiv:0710.3820v1

[Sta13] F. Staub. SARAH 3.2: Dirac gauginos, UFO output, and more. In *Computer Physics Communications*, **184**:7 (July 2013), pp. 1792–1809. arXiv:arXiv:1207.0906v3

[Wei67] S. Weinberg. A Model of Leptons. In *Physical Review Letters*, **19**:21 (Nov. 1967), pp. 1264–1266

[Wei05] S. Weinberg. *The Quantum Theory of Fields, Volume 1: Foundations*. Cambridge University Press, 2005

[Wig39] E. Wigner. On unitary representations of the inhomogeneous Lorentz group. In *The Annals of Mathematics*, **40**: (1939), p. 149

[Wu57] C. S. Wu. Experimental Test of Parity Conservation in Beta Decay. In *Physical Review*, **105**:4 (Feb. 1957), pp. 1413–1415

Declaration

Hereby I declare the single handed composition of this thesis. Furthermore, I confirm that no other sources have been used than those specified in the thesis itself. This thesis has not been submitted as part of another examination process neither in identical nor in similar form.

January 13, 2014