UCLouvain

FACULTE DES SCIENCES

Université catholique de Louvain
Faculty of Science
School of Physics

Efficient Amplitude Computation

Optimizing matrix elements in MadGraph5_aMC@NLO

Author : Jorge Emiliano NAVARRO MORALES

Supervisors : Prof. Fabio MALTONI – Olivier MATTELAER

Reader : Gwenhaël DEWASSEIGE

Academic year 2024-2025

Dissertation presented in view of obtaining the academic degree of Master [120] in physical sciences, in-depth

Contents

Introduction

`MadGraph5_aMC@NLO` is a Monte-Carlo simulation tool capable of generating tree-level Feynman diagrams from a given Lagrangian theory and output Fortran files that compute squared matrix elements simulating high-energy collider scattering events [1]. These squared matrix elements are integrated by adapted Monte-Carlo methods in order to simulate collider observables such as cross-sections. These way, the squared matrix elements are the building bricks needed to probe through the phase space of final state particles. Optimizing the matrix element generation algorithms, while maintaining them adapted to the Monte-Carlo integrations, is a worth-taking challenge. For conciseness, in this work, we will refer to the software simply as `MadGraph`.

The goal of this thesis is to study the methods ans specific routines that `MadGraph` uses to generate matrix elements for LHC-like simulations, and then to propose and implement concrete optimizations to reduce the required number of kernels. We will particularly target cases that could help diminish computations of large multiplicities of final-state particles, by tackling down redundancies, and exploiting symmetries.

Computations at Leading Order (LO), even if they represent the most basic approximations, already consume significant CPU resources, and we wish to motivate our work as a hint into future implementations into GPU parallelization. We do not aim at having optimizations that reduce computation time, but rather that reduce the amount of needed code.

Three approaches will be explored:

- A Flavour optimization: for processes differing only by external particle spinor flavours
- A C-symmetry inspired optimization: for processes related by an exchange in the fermionic current.
- A Cross-symmetry optimization: which allows to exchange initial and final state particles.

The first two of these optimizations will be implemented in the standalone Fortran output of `MadGraph`, as we only wish to give proof-of-concept. Modifying the Python metacode responsible for diagram generation would be the next step.

Chapter 1 presents the theoretical background for collider simulations and motivates the proposed optimizations. Chapter 2 provides the conventions and technical foundations, specifically reviewing `HELAS` routines [9] for helicity spinor and polarization vector wavefunctions. We will also review how the colour algebra is symbolically treated, and coloured ordered amplitudes. In Chapter 3, we implement the proposed optimizations and validate them by numerical comparison against original MadGraph results. During this chapter, we derive and prove an interesting colour algebra result, allowing to implement reversing QCD flow algorithms.

Chapter 1

Theory Fundamentals

No physical theory has been as successful, in its prediction power and accuracy, as quantum field theory. This theory remains the baseline for description of quantum particles interactions at high energy scales such as in events generated by accelerators based experiments.

The present chapter summarizes some of the relevant aspects of quantum field theory and collider physics that will remain essential to understand the present master's thesis work. Our main objective is not to give a complete review of all formal aspects and methods, but to gain insight on the physics behind `MadGraph`'s LHC-like event . Understanding `MadGraph`'s methods will allow to motivate this work and to contextualize any optimizations contributions to the algorithm.

Our approach will follow closely a series a lectures on `MadGraph` given at Iwate Collider School 2025 [**?**]. Technical aspects on Feynmann diagram generation algorithm and squared matrix element computation will be exposed thoroughly in Chapter 2.

We will dedicate the latter part of this chapter to state some relevant precautions needed when implementing optimizations while using Monte-Carlo methods for integrating cross-sections. We will analyze the particular case of implementing optimizations through Crossing-symmetries.

1.1 Simulation of collider events

Simulations play an essential role in particle physics research. As successful as the Standard Model (SM) has been, it does not allow to have a complete description of all observed phenomena nor it allows to answer all fundamental questions. Among its limitations are the absence of a dark matter candidate [3], the lack of neutrino masses [6], the discrepancies between the predicted value of the muon anomalous magnetic momentum [2] and the insufficient explanation for baryogenesis [11]. Collider experiments permit to probe for new physic by producing large amounts of data arising from highly energetic interactions. This data can be interpreted as physical observables which in turn can be validated or challenged through the means of simulated physics. These simulations can

be based on both the SM or larger theories, commonly denominated Beyond Standard Model (BSM) theories, in order to interpret experimental results and identify potential signs of new patterns and phenomena.

Ultimately, the aim of any collider event simulator is to produce observables such as cross-sections or differential cross-sections that can be compared to experimental data.

As stated in [1] the most efficient, reliable and compact way to encode information from any local field theory is to write its Lagrangian and determine its interactions from textbook rules Feynman vertices. `MadGraph` follows this recipe, as it is a highly versatile algorithm that simulates several observables from Lagrangian descriptions of models. Given a model Lagrangian, `MadGraph` makes use of Monte-Carlo physics to simulate observables as if those were produced at the LHC. This way we can effectively associate data to any given local theory. It must be stressed that the other way around is equally as important, if not more, to be able to associate a theory from experimental data, but that is a discussion beyond the scope of the present work.

1.1.1 Monte-Carlo physics: multi-scale and LHC-master formula

Collisions at LHC generally involve two proton beams, with estimated energies of up to 7 TeV each, allowing for highly energetic phenomena to occur. As protons are bound states, they do not interact themselves, but it is rather the partons inside their internal structures that do so. As such, the energy of a given interaction will not be the total energy of the accelerated hadrons but fractions endowed to the interacting partons. The probability of a given parton to interact at a certain fraction of the total energy is computed using Parton Distribution Functions (PDF)[1] at a given arbitrary scale.

Monte-Carlo simulations, which require a multiscale description, as sketched in Figure 1.1:

a.) High-Q^2, hard scattering: at the highest energy level, at Tev scales, there are deep inelastic interactions governed by perturbative quantum field theories. Matrix squared elements can be computed from first-principle descriptions, according to the considered process. Any presence of BSM interactions shall appear at this level.

b.) Moderate-Q^2, parton showers: as quarks and gluons are radiated from the High-Q^2 scatterings, they are subject to QCD interactions that make them cascade into other partons at around upper GeV scales interactions. First-principle descriptions models can still be used to describe this phase as it remains perturbative. Energy becomes too low to consider any sensible BSM features at this level.

[1] PDF's are not probabilities in a strict sense, but distributions extracted from global experimental fits.

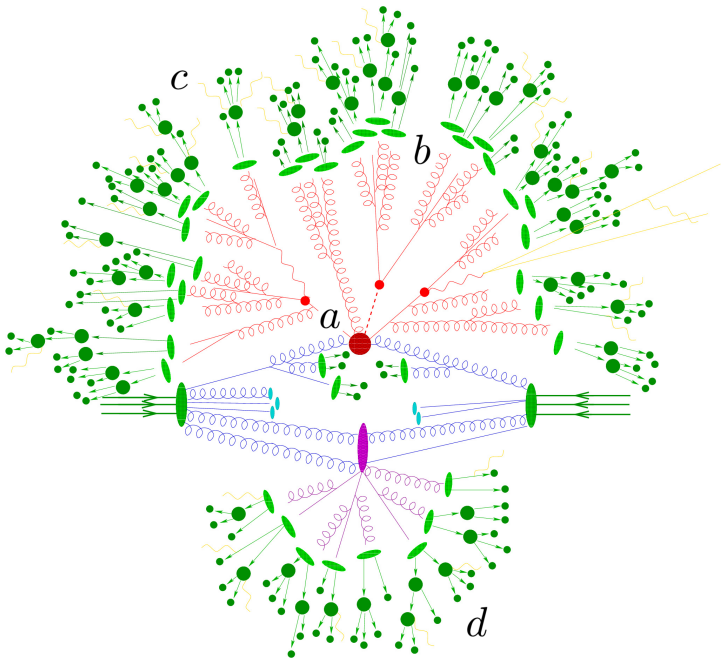


Figure 1.1: Sketch of a hadron collision described by a Monte Carlo event generator. a. hard scattering b. parton shower c. hadronization and d. the underlying event. (Source: Image courtesy of Frank Krauss)

c.) Low-Q^2, hadronization: at lower GeV or MeV scales non-pertubative phenomena govern the interactions, and confinement forces any QCD-coloured particles to hadronize. These are the particles that live long enough to be observed experimentally and produce signals. We also find at this level all underlying events, sometimes at lower GeV or MeV scales. There are events such as interactions coming from the beams, other softer parton scatterings occurrings, or even multiparton scatterings. This simulates a contextual signal that will be always be present in experimental data.

It must be stressed that `MadGraph` operates only at a.) the highest energy level to compute squared matrix elements. Parts b.) and c.) are usually dealt within parton-shower algorithms such as `Pythia8`.

In this context, the cross-section for a $2 \to X$ parton-level process can be computed using:

$$\sigma_{pp\to X} = \sum_{a,b} \int dx_1 dx_2 d\Phi \; f_a(x_1, \mu_F) f_b(x_2, \mu_F) \times \hat{\sigma}_{ab\to X}(\hat{s}, \mu_F, \mu_R) \; . \tag{1.1}$$

The integral is performed over the whole phase space Φ of final states and over parton momentum fractions x_1 and x_2. f_a and f_b are the PDF's for the initial partons a and b, which depend upon a chosen factorization scale μ_F. Any process whose Q^2 is less energetic than this μ_F scale is factorized into the PDF's. Lastly, $\hat{\sigma}_{ab\to X}$ is the partonic cross section, which depends upon the chosen model's renormalization scale

μ_R. For LO events, such as the ones we will study during this thesis, we consider that $\hat{\sigma}_{ab\to X} = |\mathcal{M}(\alpha_s(\mu_R))|^2$ and we usually take $\mu_R = Q^2$.

The integral in Equation 1.1 is evaluated numerically by making use of Monte-Carlo integration methods (which give their name to this family of physics simulations). These are based on estimating the integral through sampling randomly the integrand value. A full description on Monte-Carlo methods is beyond the scope of this work, but some common usage in high energy physics can be found in [13]. It must be stressed that these methods require to be able to compute a large amounts of times the square matrix element $|\mathcal{M}|^2$, each for every probed point in the phase space. Plainly put, Monte-Carlo physics require to compute $|\mathcal{M}|^2$ for any given model (masses, coupling constants, etc.) and at several different kinematic configurations (energy, momenta, helicities and polarizations of the incoming and outcoming particles, etc).

The important takeaway from Equation 1.1 is that it presents a way to make predictions at LHC even though the physics responsible for describing the wavefunction of the proton is non-perturbative, by factorizing these non-perturbative effects into the PDF functions and assuming that factorization holds. Given methods to model PDF's, parton showers and hadronization, it becomes possible to simply change the squared matrix computation to be able to generate well-grounded Monte-Carlo simulations for different theoretical models.

It is also worth mentioning that the dependence on a renormalization scale μ_R is an artefact of the perturbative regime to which we compute squared matrix elements. If we had an integrable theory, and wouldn't need to rely on these approximations, any dependence upon μ_R would disappear.

1.1.2 Helicity amplitudes

The focus on the remainder of this work will be on the computation of the squared matrix element at the deep inelastic scattering level. `MadGraph` works as a Python written meta-code, by generating all (non-zero amplitude) tree-level Feynman diagrams for a specified process, and by outputing code necessary to evaluate numerically the squared matrix element at given phase space points. During this work, we will not delve into deeper than just Leading Order (LO) computing terms, even if current state methods allow to do so.

The squared matrix element is computed from helicity amplitudes methods. As explained in [4], the usual textbook methods for computing an unpolarized cross section involve squaring the scattering amplitude at the beginning, then summing analytically over the spins of external states, and transforming the result into an expression that only involves momentum invariants (Mandelstam variables) and masses. This approach is not always feasible, for processes with $2 \to N$ usually grow as $N!$ number of Feynman diagrams, which in turn would require for $(N!)^2$ products to obtain $|\mathcal{M}|^2$. Helicity

amplitude methods solve this issue by first computing the matrix element for each Feynman diagram and each helicity combination. From there, once their complex value is known, multiply them by their complex conjugate to obtain the desired squared matrix element. This process scales in a much more convenient $N!2^N$ way. Also we will consider that all initial state incoming particles are unpolarized, as they usually are in LHC experiments.

Therefore, for a given unpolarized process with n number of external particles and m number of Feynman diagrams, the square matrix element is

$$|\mathcal{M}(p_1, \dots, p_n)|^2 = \frac{1}{N_{\text{colours}}} \sum_{\textbf{colours}} \frac{1}{N_{\text{helicities}}} \sum_{\text{helicities}} \left| \sum_{i}^{m} \mathcal{M}_i(p_1, h_1 \ ; \dots ; \ p_n, h_n) \right|^2 \tag{1.2}$$

where each p_j is the four-momenta and each h_j is the helicity or polarization of the j-th particle, $\mathcal{M}_i$ is called the helicity amplitude or helicity matrix element for the i-th Feynman diagram, and the left sum is performed over all permutations of possible helicities/polarizations of every external particles. As we consider the incoming particles to be unpolarized, we also average over the total number of helicities permutations $N_{\text{helicities}}$ of the initial state. When dealing with QCD interactions, a similar average must be done to render the process colourless, summing over all SU(3) colours and dividing over a factor of permutations of initial state particles N_{colours}. The squared matrix result is analytic, but it is worth noting that when computing cross-sections with expressions such as (1.2) some Monte-Carlo methods will not sum over all helicities, but rather randomly compute valid permutations of helicities/polarizations.

During the rest of this work, we will usually make reference to helicity amplitudes when talking about matrix elements. As such, the average over helicities will mostly be implied unless otherwise stated.

1.1.3 Single-Diagram-Enhancement

An important remark on the Monte-Carlo method employed by `MadGraph` is that it has been optimized to probe the phase space in order to find significative but very narrow contributions to the overall integral from (1.1). It is known a priori that the squared helicity amplitude of a single Feynman diagram $|\mathcal{M}_i|^2$ has peak-like structures at either resonances or near on-shell propagators. We can therefore evaluate the phase space integral as

$$I = \int d\Phi |\mathcal{M}|^2 = \sum_{i=1}^{m} \int d\Phi |\mathcal{M}|^2 \frac{|\mathcal{M}_i|^2}{\sum_{j=1}^{m} |\mathcal{M}_j|^2}. \tag{1.3}$$

The advantage of this reformulation holds in the fact that, in the classical limit where Feynman diagrams contributions interfere very few, or in phase space regimes where

there are low interferences, we can consider Monte-Carlo optimizations that equate

$$\frac{|\mathcal{M}|^2}{\sum_{j=1}^{m} |\mathcal{M}_j|^2} \approx 1\,, \tag{1.4}$$

and therefore allow the integral I to be

$$I \approx \sum_{i=1}^{m} \int d\Phi |\mathcal{M}_i|^2 \ . \tag{1.5}$$

at appropriate probed phase spaces regions.

This technique is called Single-Diagram-Enhanced (SDE) multi-channel integration and was introduced for the first time in [7]. We call each term in this sum of integrals an integration channel. SDE presents several advantages and has been thoroughly validated by benchmark tests, but has as an explicit requirement that every single Feynman diagram contribution must be computed and stored.

1.2 Motivations and proposed optimizations

Performing large amounts of numerical computations for Monte-Carlo integration is no trivial feat. This is all the more important for processes with large amounts of final state particles containing jets. For example, back in 2011, [1] showed that processes such as $pp \to W^+jjjjj$, simulating a proton-proton collision whose hard scattering final state are a W^+ boson and five jets, could take almost 12 hours to be computed at a 128-core computer cluster with Intel Xeon 2.50GHz CPUs. Of course, we have since had several improvements, both on the hardware and software aspects of simulating these types of collisions. From now on, we are adopting `MadGraph`'s multiparticle labels nomenclature, where $p = j$, protons and jets, stand for $g/u/\tilde{u}, c/\tilde{c}/d/\tilde{d}/s/\tilde{s}$ gluons and quarks and where $l^\pm$ leptons, stand for $= e^\pm/\mu^\pm$ electrons and muons.

Recent optimizations such as [8] have tackled down speeding up the computation of LO processes by recycling helicity amplitudes during the evaluation of squared matrix elements, and providing new multi-channel handling of the SDE integration.

In CERN's HL-LHC computing review [5], it is stated that one of the software modernization challenges is to achieve better data parallelization such that matrix element functions are computed repeatedly at many phase space points. A currently explored option consists in using GPU kernels to parallelize computations [12]. Nonetheless, porting `MadGraph`'s algorithm to GPU adapted optimizations is a challenge by itself, as it usually requires to diminish the amount of functions and to ameliorate compilation times.

The proposed optimizations for this master's thesis work is to reduce the number of kernels needed to compute multiprocess events, while maintaining the same amount

of integration channels for SDE integration. We expect that having less code would allow for potential future integrations with GPU. We hope that the proposed optimizations could be employed using SIMD (single instruction multiple data) instructions. To attain this result, we wish to give some proof-of-concept that the squared matrix elements for the following processes can be computed within the same matrix algorithm file:

- Multiprocesses with multiparticle labels such as $l^{\pm}$. For example, the process $e^+e^- \to l^+l^-$, which encompasses as subdiagrams both $e^+e^- \to e^+e^-$ and $e^+e^- \to \mu^+\mu^-$ (shown in detail in Chapter 3, Figure 3.1). This is grounded on the fact that Feynman diagrams for the process $e^+e^- \to e^+e^-$, at a given approximations, already compute the $e^+e^- \to \mu^+\mu^-$ matrix element. We will refer to this proposed optimization as "Flavour optimization", and review later how this idea can be extended locally.
- Multiprocesses involving a subprocess with fermion currents and their corresponding antifermion currents. For instance, $ae^+ \to ae^+$ and $ae^- \to ae^-$ or even $gu \to gu$ and $g\tilde{u} \to g\tilde{u}$ (shown in detail in Chapter 3, in Figures 3.7 and 3.8). This is inspired by the Lagrangian discrete C-symmetry, known to leave invariant QED and QCD interactions. We will call this accordingly "C-symmetry inspired optimization".
- Cross-symmetries, for processes such as $e^+e^- \to \mu^+\mu^-$ and $e^-\mu^- \to e^-\mu^-$ which can be related to each other by taking analytical continuations of their matrix elements. Implementing this optimization would be the most powerful one in terms of reduced amounts of kernels, since quite a few processes could be "crossed" into each other.

For multiprocess events, a single subdirectory is created for each subprocess. There, the squared matrix element is computed algorithmically following Equation 1.2. `MadGraph` automatically sorts into a same subprocess directory any subprocesses with matching spin, colour and mass of external particles. For example, processes whose sole distinction is an up-quark u instead of a down-quark d, which in the SM high-energy approximation are both massless. To illustrate the proposed optimizations further, consider the process $pp \to jj$, set with QED=0, meaning that only QCD interactions are taken into account. This multiprocess produces a total of 12 directories summarized in the first column of Table 1.1. Then, using first the Flavour optimization and then the C-symmetry inspired, we can reduce them for a total of 5 necessary kernels, as shown on the third column. Implementing Cross-symmetries would even leave us at just 3 required kernels.

To implement this reduction in the amount of needed directories, we will delve into the squared matrix element computation routines in Chapter 2 and then see how these routines shall be modified in Chapter 3.

Default kernels	Flavour optimized	C-inspired optimized	Cross-symmetry optimized
$gg \to gg$	$gg \to gg$	$gg \to gg$	$gg \to gg$
$gg \to u\tilde{u}$	$gg \to u\tilde{u}$	$gg \to u\tilde{u}$	$gu \to gu$
$u\tilde{u} \to gg$	$u\tilde{u} \to gg$	$u\tilde{u} \to gg$	
$gu \to gu$	$gu \to gu$	$gu \to gu$	
$g\tilde{u} \to g\tilde{u}$	$g\tilde{u} \to g\tilde{u}$		
$u\tilde{c} \to u\tilde{c}$			
$uc \to uc$	$uu \to uu$	$uu \to uu$	$uu \to uu$
$uu \to uu$			
$u\tilde{u} \to u\tilde{u}$	$u\tilde{u} \to u\tilde{u}$		
$u\tilde{u} \to c\tilde{c}$			
$\tilde{u}\tilde{c} \to \tilde{u}\tilde{c}$	$\tilde{u}\tilde{u} \to \tilde{u}\tilde{u}$		
$\tilde{u}\tilde{u} \to \tilde{u}\tilde{u}$			

Table 1.1: List of kernels created by `MadGraph` for the $pp \to jj$ process and reduced list of necessary kernels after applying successively each proposed optimization. These are estimates based on the condition that all proposed optimizations work correctly

1.2.1 Crossing-symmetry

As we're interested into optimizing matrix element generations and making use of fewer functions, an interesting case with solid theoretical background could be to exploit Crossing symmetries as to be able to move any initial state particle into final state particles and vice versa.

Crossing symmetries are made possible thanks to Lorentz invariance, four-momenta conservation and the mathematical hypothesis that squared matrix elements expressions have complex analytical continuations.

To see this, consider any process with n number of incoming and m number of outcoming particles, at the center of mass frame, and assume all particles are on-shell with positive energy. Then, by conservation laws

$$\sum_{i=1}^{n-1} p_i + p_n - \sum_{j=1}^{m} p'_j = 0\,, \tag{1.6}$$

where the p_i are the four-momenta of initial state particles, and p'_i are those for final state particles. Heuristically, this conservation law allows for

$$\sum_{i=1}^{n-1} p_i - (-p_n) - \sum_{j=1}^{m} p_j = 0\,, \tag{1.7}$$

replacing the n-th incoming particle with an outcoming one with a four-momenta $p'_n =$

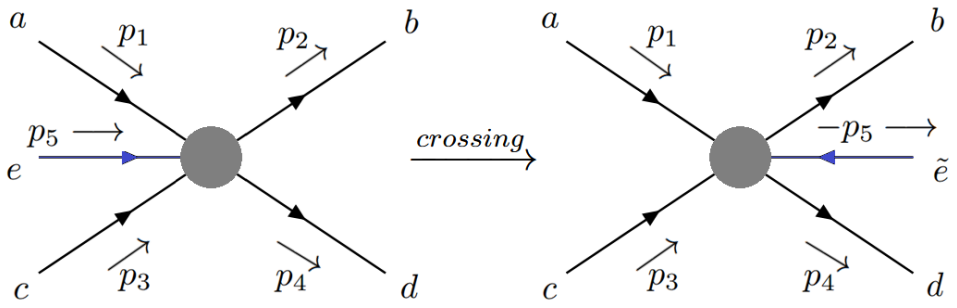


Figure 1.2: Crossing symmetry applied to a single incoming line.

$-p_n$. This is equivalent to considering a process with same $n-1$ initial state particles and $m+1$ final ones, but where the incoming n-th particle has being replaced by an outcoming **anti**particle.

Let's call $\mathcal{M}_a$ the matrix element for the original process and $\mathcal{M}_b$ the one after the $n-$th particle has "crossed". At the matrix element level, we can perform this transformation by defining an analytical function F over complexified momenta such that:

$$\mathcal{M}_a(\dots, \phi(p) \to \dots) = F(\dots, p\,; \dots) \tag{1.8}$$

$$\xrightarrow{crossing} \quad \mathcal{M}_b(\dots \to \dots, \bar{\phi}(k)) = F(\dots\,; \dots, k) \tag{1.9}$$

where $k = -p$. See that since both p and k are on-shell, that implies that either k^0 or p^0 is negative, rendering one of the former expressions unphysical. However, under the assumption that F admits an analytic continuation to complex values of momenta, the relation between $\mathcal{M}_a$ and $\mathcal{M}_b$ is still valid. A more complete proof by construction of the LSZ reduction formula can be found in [10]. Figure 1.2 illustrates this with a simple example.

Having computed the matrix element for a process such as $a + b \to c + d$, we wish to be able to recycle this result in order to compute, in the same single function, other crossed processes such as for instance $a + \tilde{c} \to \tilde{b} + d$ or $\tilde{c} + \tilde{d} \to \tilde{a} + \tilde{b}$ amongst others. Doing so would allow for instance to regroup the $gg \to u\tilde{u}$ with $gu \to gu$ in Table 1.1.

Unfortunately, it isn't as easy to implement crossing symmetries for our optimization purposes as crossing symmetries mix integration channels corresponding to Feynman diagram's helicity amplitudes $\mathcal{M}_i$. To see this, consider the two QED processes $e^+e^- \to \mu^+\mu^-$ and $e^-\mu^- \to e^-\mu^-$, which are related by crossing symmetries, as is illustrated in Figure 1.3.

The matrix element for $e^+e^- \to \mu^+\mu^-$ is an s-channel while $e^+e^- \to \mu^+\mu^-$ is a t-channel.

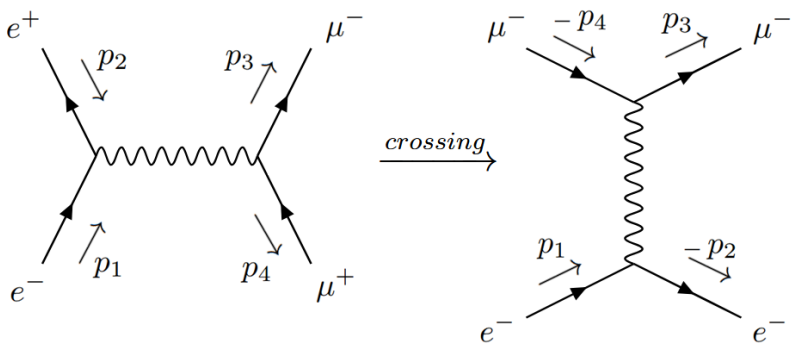


Figure 1.3: Example of Crossing-symmetry, passing from an s-channel to a t-channel.

Peaks of these channels will differ, making any possible optimization based on cross symmetries not easely translatable to the same integration channel.

We have nonetheless shown numerically that it is possible to change the squared matrix element algorithm and obtain crossed symmetries. It requires taking a little bit of caution, as `MadGraph` by default computes the energy of any on-shell spinor wavefunction with four-momenta p as simply $|(p^0)|^2$, which would be unfit for negative values of $(p^0)^2$.

Chapter 2

Matrix element computation

The present chapter has two main objectives. On the one hand it will set the necessary conventions to consider spinor and vector fields as well as their interactions following HELAS paper[9]. On the other hand, this chapter will provide a general overview of the fundamentals behind MadGraph's squared matrix element computation following the works in [1], and [?] presenting the general philosophy behind the wavefunctions routines and the color algebra module. In order to do so, we will go through a couple of textbook examples in detail; a QED process of photon-electron scattering $\gamma e^- \to \gamma e^-$ and its QCD analogue, gluon-quark scattering $gu \to gu$. Both were chosen specifically to illustrate the helicity computation methods and colour algebra module.

Chapter 3 will make use of these descriptions to implement the motivated optimizations from Chapter 1 to MadGraph's algorithm, thus justifying the long technical descriptions that follow.

2.1 HELAS and Conventions

We have introduced MadGraph as a tool that automates the computation of tree-level (LO) scattering squared matrix elements with the aim of ultimately performing the sum from Equation 1.2. MadGraph Python meta-code can generate amplitudes at tree-level for any Lagrangian based model (renormalizable or effective) implemented in FeynRules via the UFO interface [1].

The resulting squared matrix element computation program is usually written in Fortran language. This choice follows both from the language low-level controlling characteristics, alleviating the needed time to perform computationally intensive calculations, but also for its syntax strictness.

MadGraph's matrix element is computed numerically by making use of the HElicity Amplitude Subroutines (HELAS) from [9] which are automatically generated by the ALOHA package. There is a large library of subroutines that can be called following the requirements of the considered process. These subroutines take as variables the type of field

(if scalar, vectorial or spinorial), the mass, the helicity (or polarization), and the direction of the flow towards the interactions. Coupling values, which may have a chirality dependence, as well as masses and widths for off-shell particles must also be provided as variables and are imported directly from the model. Then, according to each tree-level Feynman diagram for a given process, and for any set of the aforementioned variables, `HELAS` subroutines compute either the wavefunctions of external particles, the interactions and propagators, or the diagram helicity subamplitude. We will illustrate this in the next subsection.

A simple recipe is to think of each `HELAS` subroutine as either an external line or as a vertex plus a propagator. The result of each subroutine will therefore be either a four-vector of real values, a four-spinor of complex values or simply a complex valued diagram subamplitude.

If the analyzed process does not contain any $SU(3)$ coloured particles, it just suffices to square the matrix element to obtain the desired result, at the specific phase space value of the combinations of external momenta. When a process involves QCD interactions, colour structures associated with each Feynman diagram will intervene in the squared matrix element computation. These are not added by `HELAS` subroutines, and must be dealt on their own. Their computation and formalism will be detailed in subsection 2.3.

2.1.1 `HELAS` Conventions: γ-matrices and spinors

Throughout this program, γ-matrices and spinors are represented in the Weyl basis using the following conventions from [9]:

Let the four-momentum of the spinor be

$$p^\mu = (E, p_x, p_y, p_z). \tag{2.1}$$

We set the chirality-left sector in the upper component:

$$\gamma^5 = \begin{pmatrix} -\mathbb{1} & 0 \\ 0 & \mathbb{1} \end{pmatrix}.$$

Correspondigly the γ-matrices are

$$\gamma^\mu = \begin{pmatrix} 0 & \sigma_+^\mu \\ \sigma_-^\mu & 0 \end{pmatrix},$$

where $\sigma_\pm^\mu \coloneqq (\mathbb{1}, \pm\vec{\sigma})$ and where $\vec{\sigma}$ is the vector containing Pauli matrices.

We define the helicity-eigenspinors $\chi_+(\vec{p})$ and $\chi_-(\vec{p})$ as

$$\chi_+(\vec{p}) = \frac{1}{\sqrt{2|\vec{p}|(|\vec{p}|+p_z)}} \begin{pmatrix} |\vec{p}|+p_z \\ p_x+ip_y \end{pmatrix},$$

$$\chi_-(\vec{p}) = \frac{1}{\sqrt{2|\vec{p}|(|\vec{p}|+p_z)}} \begin{pmatrix} -p_x+ip_y \\ |\vec{p}|+p_z \end{pmatrix}.$$

These helicity eigenspinors satisfy

$$\frac{\vec{\sigma}\cdot\vec{p}}{|\vec{p}|}\chi_\lambda(\vec{p}) = \lambda\chi_\lambda(\vec{p}),$$

where $\lambda = \pm 1$. These definitions are ill-defined when the momentum is solely in the negative z-direction $\vec{p} = -|\vec{p}|$. We fix the notation by taking the limit where $p_y = 0$ and $p_x \to 0^+$, giving thus

$$\chi_+(\vec{p}) = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$\chi_-(\vec{p}) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

The four-spinors are then defined as follows:

$$u_\lambda(p) = \begin{pmatrix} \omega_{-\lambda}(p)\chi_\lambda(\vec{p}) \\ \omega_\lambda(p)\chi_\lambda(\vec{p}) \end{pmatrix},$$

$$v_\lambda(p) = \begin{pmatrix} -\lambda\omega_\lambda(p)\chi_{-\lambda}(\vec{p}) \\ \lambda\omega_{-\lambda}(p)\chi_{-\lambda}(\vec{p}) \end{pmatrix},$$

where

$$\omega_\pm(p) \coloneqq \sqrt{E \pm |\vec{p}|}.$$

These conventions are such that, as expected, for any massless on-shell fermion, u_+ and v_- are right-handed spinors, while u_- and v_+ are left-handed spinors[1].

2.1.2 HELAS Conventions: polarization vectors

In a similar way, we want to fix the helicity eigenvectors of the vector bosons. If we consider that the four-momentum of the vector is

$$k^\mu = (E, k_x, k_y, k_z)$$

[1]For massless fermions, helicity coincides with chirality.

the three-independent polarization vectors which satisfy $k \cdot \epsilon(k) = 0$ are:

$$\epsilon^\mu(k,1) = \frac{1}{\left|\vec{k}\right| k_T}(0, k_x k_z, k_y k_z, -k_T^2),$$
$$\epsilon^\mu(k,2) = \frac{1}{k_T}(0, -k_y, k_x, 0),$$
$$\epsilon^\mu(k,3) = \frac{E}{m\left|\vec{k}\right|}(\left|\vec{k}\right|^2/E, k_x, k_y, k_z),$$

where

$$m = \sqrt{E^2 - \left|\vec{k}\right|^2},$$
$$k_T = \sqrt{k_x^2 + k_y^2}.$$

Since $\epsilon^\mu(k,1)$ and $\epsilon^\mu(k,2)$ are ill-defined for $k_T = 0$, we fix the notation by taking the limit where $k_y = 0$ and

$$\begin{cases} k_x \to 0^+ & (k_z > 0) \\ k_x \to 0^- & (k_z < 0). \end{cases}$$

The helicity eigenvectors for $\lambda = \pm, 0$ are:

$$\epsilon^\mu(k, \lambda = \pm) = \frac{1}{\sqrt{2}}(\mp\epsilon^\mu(k,1) - i\epsilon^\mu(k,2)),$$
$$\epsilon^\mu(k, \lambda = 0) = \epsilon^\mu(k,3).$$

There is no $\lambda = 0$ helicity component, corresponding to a longitudinal polarization, for the massless vector boson. Also $\lambda = +1$ is the right-handed polarization and $\lambda = -1$ the left-handed one.

2.1.3 Feynman rules and `HELAS` routines

We can now get into the equivalence between some common Feynman rules for SM interactions and `HELAS` routines. During this chapter and the rest of this work, we will make use of pseudocode to illustrate the principal ideas behind some important `HELAS` routine.

Both fermion four-spinors and boson vectors wavefunctions are coded into an object called an `aloha_object`. This is a user-defined data type containing the following information:

```
class aloha_object:
    wf: list of 4 complex numbers
    p: list of 4 real numbers (indexed 0 to 4)
```

These arrays are interpreted differently for each case; in the fermion spinor case, the values in the complex array `wf` must be interpreted as the helicity eigenspinor components. In the boson vector case, values in `wf` correspond to polarization eigenvector in spatiotemporal indexes. In both cases the values in the real array `p` are the four-momenta components.

An important physical remark is that the only non coded physical information in a `aloha_object` for fermions is wether the spinor wavefunction is a column or a row vector and wether it is a particle or an antiparticle. Therefore, in order to perform any matrix element computation, both must be specified as variables of the routines calling external particles. We will detail in the following subsection how this information propagates through the algorithm.

Table 2.1 presents the Feynman diagram for external wavefunctions, along with the Feynman rules and their corresponding `HELAS` routines. In the same manner Table 2.2 presents the Feynman diagrams for vertexes along with off-shell wavefunctions. The explicit nature as row or column vectors is detailed for each spinor wavefunctions resulting routine. For convenience we call the resulting row spinors "bra-like" and column spinors "ket-like", and denoted them symbolically by $\langle\psi|$ and $|\psi\rangle$.

2.2 Example overview: $\gamma e^- > \gamma e^-$

Suppose we wish to compute the squared matrix element for the process $\gamma e^- \to \gamma e^-$[2]. We consider this process within the high-energy approximation where the electron mass is zero and also within the SM framework. There are two Feynman diagrams for this process, corresponding to s and t channels as showed in Figure 2.1.

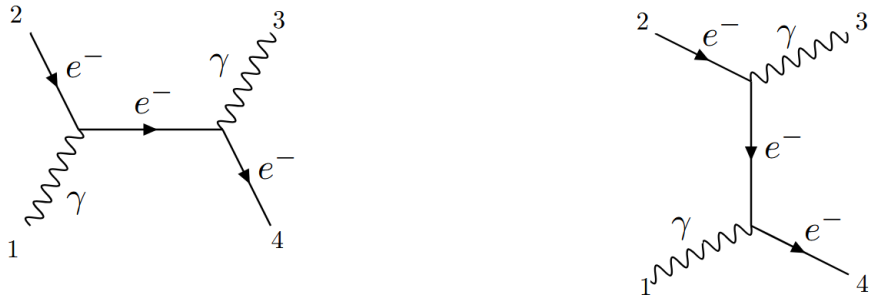


Figure 2.1: Tree level diagrams for the $\gamma e^- \to \gamma e^-$ process

Using `HELAS` routines, the pseudocode to compute the helicity subamplitudes from this example is presented in Table 2.3, along with a step-by-step diagramatical illustration

[2]in `MadGraph`'s terminal, this process is called by typing `a e- > a e-`

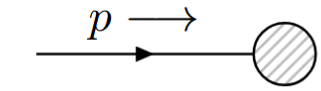
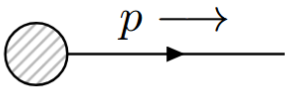
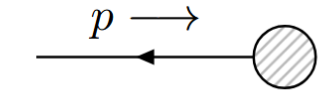
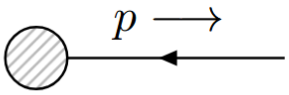
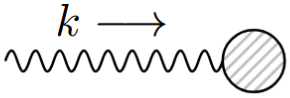
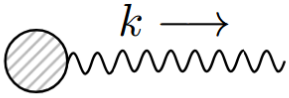
Feynman diagram	Feynman rule	HELAS routine pseudocode	$\langle\psi\|$ or $\|\psi\rangle$
	$u_\lambda(p)$	`incoming_fermion(p, hel, sf = +1)`	$\|\psi\rangle$
	$\bar{u}_\lambda(p)$	`outcoming_fermion(p, hel, sf = +1)`	$\langle\psi\|$
	$\bar{v}_\lambda(p)$	`outcoming_fermion(p, hel, sf = -1)`	$\langle\psi\|$
	$v_\lambda(p)$	`incoming_fermion(p, hel, sf = -1)`	$\|\psi\rangle$
	$\epsilon^\mu_\lambda(k)$	`vector(k, hel)`	
	$\epsilon^{\mu*}_\lambda(k)$	`vector_star(k, hel)`	

Table 2.1: `HELAS` subroutines for external wave functions. On the thirds column `hel` $= \pm 1$ is the integer helicity indicator variable and `sf` $= \pm 1$ is the integer particle or antiparticle indicator variable. Names of the routines depend on the flow direction with respect to the interaction, here represented by a blob. Be wary of the antiparticle case, as the names can seem unintuitive. A final state anti-fermion would be called with `incoming_fermion(... sf = -1)` since the fermionic flow (not the momenta) is getting out of the interation. A nice way to think about these routines is by fixing the temporal direction and the blob position and ask oneself in which cases the fermionic flow can be incoming and in which cases outcoming. We're also assuming the convention of positive energies $p^0 > 0$ and $k^0 > 0$. The fourth column specifies if the resulting wavefunction is "bra-like" or "ket-like" for the fermions.

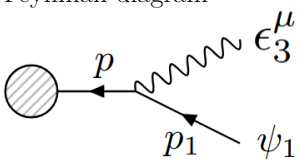
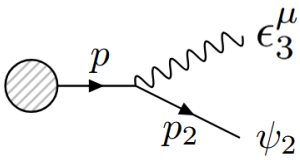
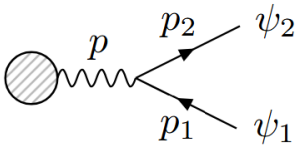
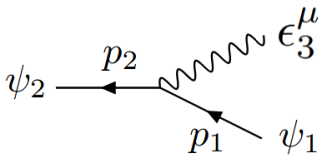
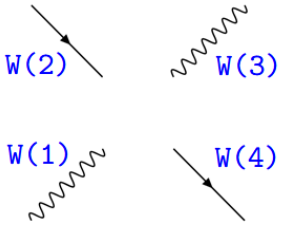
Feynman diagram	Feynman rule (unitary gauge)	`HELAS` routine pseudocode	$\langle\psi\vert$ or $\vert\psi\rangle$ or $\mathbb{C}$
	$\psi = ig\frac{\not{p}+m}{p^2-m^2+im\Gamma}\gamma^\mu\psi_1(p_1)\epsilon_{3\mu}(k)$	`FFV_2`$(\psi_1, \epsilon_{3\mu})$	$\vert\psi\rangle$
	$\bar{\psi} = ig\bar{\psi}_2(p_2)\gamma^\mu\frac{\not{p}+m}{p^2-m^2+im\Gamma}\epsilon_{3\mu}(k)$	`FFV_1`$(\psi_2, \epsilon_{3\mu})$	$\langle\psi\vert$
	$j^\nu = \bar{\psi}_2(p_2)\gamma_\mu\psi_1(p_1)\cdot -i\frac{\eta^{\mu\nu}-p^\mu p^\nu/m^2}{p^2-m^2+im\Gamma}$ $j^\nu = \bar{\psi}_2(p_2)\gamma_\mu\psi_1(p_1)\cdot -i\frac{\eta^{\mu\nu}}{p^2}$ if $m=0$,	`FFV_3`$(\psi_1, \psi_2, \epsilon_{3\mu})$	
	$A = g\bar{\psi}_2(p_2)\gamma^\mu\psi_1(p_1)\epsilon_\mu(k)$	`FFV_0`$(\psi_1, \psi_2, \epsilon_{3\mu})$	$\mathbb{C}$

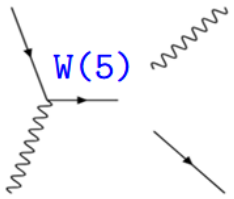
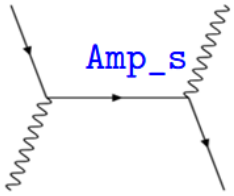
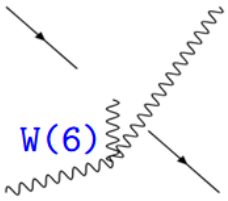
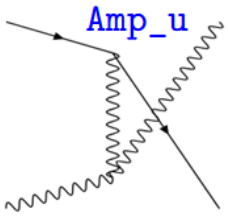
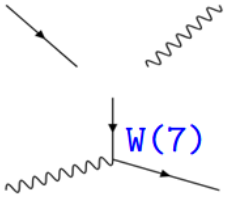
Table 2.2: `HELAS` subroutines for vertexes. On the second column, variables without a subscript p, m and Γ correspond to the off-shell propagator momenta, mass and width. The pseudocode name `FFV` stands for Fermion-Fermion-Vector interactions. The subscript for each routine depend on the resulting `aloha_object`, as either a ket-like spinor for `FFV_1`, a bra-like spinor for `FFV_2`, a vector for `FFV_3` or a complex variable equal to a Feynman diagram subamplitude for `FFV_0`. We have adopted `ALOHA`'s convention to name ψ_1 to bra-like spinors and ψ_2 to ket-like spinors. At the risk of being repetitive, the fourth column once again specifies if the resulting wavefunction is bra-like or ket-like.

of each subroutine. In this same table, we have included a somewhat strange u-channel with a triple vector vertex, which may be ignored for now.

Once all each the both channels have had their corresponding subamplitudes `AMP_s` and `AMP_t` computed, then the matrix element amplitude $\mathcal{M}$ is simply `AMP` = `AMP_s` + `AMP_t`. This same pattern, of simply adding each individual subamplitude for every Feynman diagram holds for more complex processes as long as they invoke only QED interactions. Then, the square matrix element is computed through the individual sum of each possible permutation of helicities and polarizations of the external particles for each Feynmann diagram subamplitude as exposed in Equation 1.2.

Pseudocode	**Respective Diagram**

# Start by calling all external particles 	
`W(1) = vector(p_1, hel_1)`
`W(2) = incoming_fermion(p_2, hel_2, sf = +1)`
`W(3) = vector_star(p_3, hel_3)`
`W(4) = outcoming_fermion(p_4, hel_4, sf = +1)`
	

# Then compute the amplitude for each diagram *# s-channel* `W(5) = FFV_2(W(2) , W(1))`	
`Amp_s = FFV_0(W(5), W(4), W(3))`	
# u-channel (only for the g u > g u case) `W(6) = VVV_1(W(1) , W(3))`	
`Amp_u = FFV_0(W(2), W(4), W(6))`	
# t-channel	

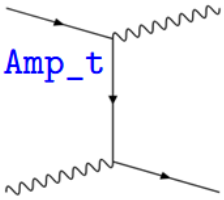
Pseudocode (continued)	Diagram (continued)
`W(7) = FFV_1(W(4) , W(1))`	
`Amp_t = FFV_0(W(2), W(7), W(3))`	

Table 2.3: Pseudocode of helicity matrix element computation for each subdiagrams of $\gamma e^- \to \gamma e^-$ and $gu \to gu$ process mirrored by their corresponding step-by-step diagram generation.

2.3 Example overview: $gu \to gu$, Colour ordered amplitudes and colour matrix

We will now detail the operations performed by `MadGraph` to compute contributions from $SU(N_c)$ Yang-Mills theories to the squared matrix element $|\mathcal{M}|^2$, such as when dealing with coloured charged quarks and gluons in SM. We will restrict our analysis mainly to $SU(3)$, but it must be noted that current versions of `MadGraph` are flexible enough to work with any value of N_c.

When dealing with QCD processes, after diagram generation, computations of the squared matrix element are performed in two steps. Suppose we now wish to compute the squared matrix element for the process $gu \to gu$. We consider this process to be within the high-energy approximation (meaning all quarks but tau τ and bottom b are massless) and also within the SM framework. In this case, we have three Feynman diagrams, as shown in Figure 2.2, corresponding to s, u and t-channels.

First, for a given set of helicities/polarizations for external particles wavefunctions, an amplitude A_i = `Amp_i` is assigned to each diagram for $i = s, u, t$ following the same algorithm as for the QED case from Table 2.3. It suffices to picture the vector wavefunctions appearing in the step-by-step diagrams as gluons instead of photons.

Secondly, according to each diagram configuration and interactions, a factor of certain functions of $\mathfrak{su}(3)$ elements is multiplied to each one of these helicity amplitudes. Functions of $\mathfrak{su}(3)$ are called coloured objects. Table 2.4 summarizes the coloured object associated to interactions according to QCD Feynman Rules for the SM.

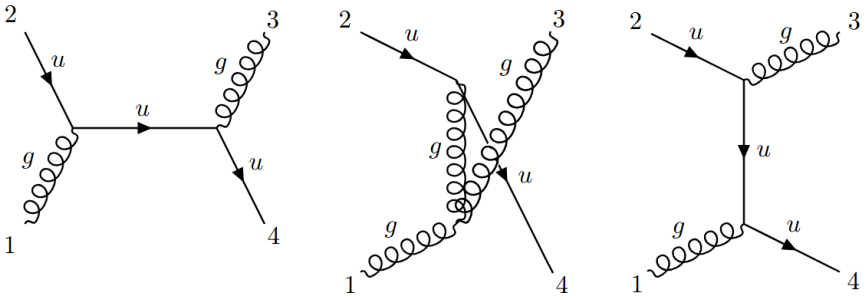


Figure 2.2: Tree level diagrams for the $gu \to gu$ process

The key idea when writing the matrix element of a QCD interaction is to factorize the colour part of each diagram, separating it from the kinematics, as in [1] and [**?**].

Thus, in an interaction with n colour charged external particles, each with momenta p_i and helicity (or polarization) h_i, we have that the sum over all helicity amplitudes $\sum_{i=1}^{m} \mathcal{M}_i$ from Equation 1.2 can be computed as

$$\sum_{i=1}^{m} \mathcal{M}_i(p_1, h_1; \ldots; p_n, h_n) := \sum_{\sigma} F^{\sigma}(\mathfrak{su}(3))\, J^{\sigma}(p_1, h_1; \ldots; p_n, h_n) \ , \tag{2.2}$$

where, σ is a given permutation of the possible colour orderings of the external particles, F^{σ} is a function of the gauge algebra $\mathfrak{su}(3)$ elements, and J^{σ} is defined as the kinematic colour ordered amplitude, which is a function of the momenta and helicities (or polarizations) of the particles.

With this factorization, the helicity dependent squared matrix element is

$$\left| \sum_{i=1}^{m} \mathcal{M}_i \right|^2 = \sum_{\sigma,\rho} J^{\sigma} F^{\sigma} F^{*\sigma'} J^{*\rho} \tag{2.3}$$

$$= \sum_{\sigma,\rho} J^{\sigma} C_F^{\sigma\rho} J^{*\rho} \, , \tag{2.4}$$

where we define $C_F^{\sigma\rho} := F^{\sigma} F^{*\rho}$ to be the colour matrix.

`MadGraph` deals with coloured objects in a purely symbolic manner. It makes use of the following properties of $SU(3)$ to express any coloured object in the fundamental representation of $\mathfrak{su}(3)$, writing them in terms of linear combination of products of $\mathfrak{su}(3)$

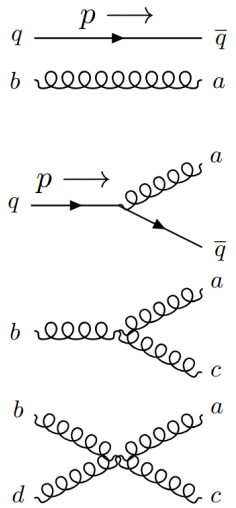
Diagram	Colour object
	$\delta_{\bar{q}q}$ δ^{ab}
	$t^a_{\bar{q}q}$
	f^{abc}
	$f^{abe}f^{cde}$; $f^{ace}f^{bde}$; $f^{ade}f^{bce}$

Table 2.4: Couloured object factors. All repeated indices are summed. We have taken the convention for fermions with energy $E > 0$ traveling with positive momenta from left to right. In the antifermion case, all fundamental and antifundamental indices order must be reversed (for instance $\delta_{\bar{q}q}$ becomes $\delta_{q\bar{q}}$). For the 4-vertex, there are Lorentz factors contributions associated with each object. Nevertheless, `MadGraph` treats the 4-vertex as three separate 3-vertexes and computes each term into separate helicity amplitudes, such that those Lorentz factors are absorbed within the coloured ordered amplitudes.

generator matrices.

$$\begin{aligned} \mathrm{Tr}\{t^a\} &= 0\,, & \mathrm{Tr}\Big\{t^a t^b\Big\} &= T_R\delta^{ab}\,, \\ t^a_{ij}t^b_{kl} &= T_R\left(\delta_{il}\delta_{jk} - \frac{1}{N_c}\delta_{ij}\delta_{kl}\right) \quad , & if^{abc}t^c &= [t^a, t^b]\,, \\ \delta_{ii} &= N_c & \delta^{aa} &= N_c^2 - 1\,. \end{aligned} \tag{2.5}$$

Where all repeated indices are summed, $i, j, k, l = 1, 2, 3$ are (anti)fundamental indices, $a, b, c = 1, 2, \ldots, 8$ are adjoint indices, t^a_{ij} is a generator of $\mathfrak{su}(3)$ in the fundamental representation, f^{abc} are the structure constants, and T_R is a normalization factor, set to $1/2$ although in the literature it is often set to one.

Importantly, performing these symbolic calculations allows to have an explicit expression the colour matrix $C_F^{\sigma\rho}$. No matter the considered process, `MadGraph` will be able to reduce the expressions of the associated coloured objects to mainly multiplications of $\mathfrak{su}(3)$ generators (or their traces). It must also be noted that the colour matrix is independent of the chosen representation of SU(3).

Going back to the $gu \to gu$ example, we can have a more straightforward expression for

Equation 2.2 by reading in the Table 2.4 which coloured objects are associated with the Feynman diagrams. There are two $\mathfrak{su}(3)$ generator matrices t^1 and t^3, one per each external gluon.[3] Thus, from step 1 we used `HELAS` routines to compute A_s, A_u and A_t, and now from step 2 we can project these amplitudes over the colour basis

$$A_s \times (t^3 t^1)_{42} \tag{2.6}$$
$$A_u \times f^{13a}(t^a)_{42} = A_u \times (-i(t^1 t^3)_{42} + i(t^3 t^1)_{42}) \tag{2.7}$$
$$A_t \times (t^1 t^3)_{42} \tag{2.8}$$

where we have adopted the (anti)fundamental indexes from the particles labels in the diagrams from Figure 2.2 and made use of the constant structures identity from Equations 2.5 to project A_u.

The coloured ordered amplitudes J^σ can hence be computed as

$$J^{13} = -iA_u + 1A_t \tag{2.9}$$
$$J^{31} = 1A_s + iA_u \tag{2.10}$$

providing thus all required elements to compute the squared matrix element by plugging 2.3 in Equation 1.2.

We can generalize this example for any QCD case in which processes have only one coloured charged fermion current of the same flavour.

Suppose that amongst the n coloured charged particles k of them are gluons. Each external gluon induces and additioanl t^a factor. The factorized element matrix expressed in terms of coloured ordered amplitudes is therefore

$$\mathcal{M}(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n) \tag{2.11}$$
$$= \sum_{i=1}^{m} \sum_{\sigma \in S_k} a_i^\sigma \times A_i(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n) \times (t^{\sigma(1)} t^{\sigma(2)} \ldots t^{\sigma(k)}) \tag{2.12}$$
$$= \sum_{\sigma \in S_k} J^\sigma(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n) \times (t^{\sigma(1)} t^{\sigma(2)} \ldots t^{\sigma(k)}) \tag{2.13}$$
$$= \sum_{\sigma \in S_k} J^\sigma(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n) \times T^\sigma \tag{2.14}$$

where a_i^σ are complex coefficients, i is the Feynman diagram index corresponding to either the s, t and u-channels, $A_i =$ `Amp_`i is the helicity amplitude for the i-th diagram,

[3] We are using the superscripts 1 and 3 in t^1 and t^3 to reference the particle labels of the initial and final state gluons. These could match the λ^1 and λ^3 Gell-Mann generator matrices of $\mathfrak{su}(3)$, but also any other generator matrices.

and where we have also defined

$$J^{\sigma}(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n) := \sum_{i=1}^{m} a_i^{\sigma} \times A_i(p_1, h_1; \ldots; p_k, h_k; \ldots; p_n, h_n)\,, \tag{2.15}$$

as the coloured ordered amplitudes and

$$T^{\sigma} := (t^{\sigma(1)} t^{\sigma(2)} \ldots t^{\sigma(k)}) \qquad \text{for } \sigma \in S_k \tag{2.16}$$

as the colour algebra basis element.

As a final remark, note that elements of the colour matrix $C_F^{\sigma\rho}$ are real numbers, since using the relations from Equations 2.5, it is possible to show that $C_F^{\sigma\rho}$ elements are polynomials of N_C for amplitudes with at most one quark line [**?**]. Therefore, for any two σ and ρ of S_k, we have that (assuming sumation over repeated indices) :

$$\begin{aligned}
C_F^{\sigma\rho} &= T^{\sigma}_{\bar{q}q} T^{\rho *}_{\bar{q}q}\ , && (2.17)\\
&= \mathrm{Tr}\{T^{\sigma} T^{\rho *}\}\ , && (2.18)\\
&= \mathrm{Tr}\{T^{\rho *} T^{\sigma}\}\ , && (2.19)\\
&= \mathrm{Tr}\{(T^{\rho} T^{\sigma *})^{*}\}\ , && (2.20)\\
&= (T^{\rho}_{\bar{q}q} T^{\sigma *}_{\bar{q}q})^{*}\ , && (2.21)\\
&= (C_F^{\rho\sigma})^{*}\ , && (2.22)\\
&= C_F^{\rho\sigma}\ , && (2.23)
\end{aligned}$$

which shows that the colour matrix $C_F^{\sigma\rho}$ is symmetric.

Chapter 3

Efficient Amplitude Computation

This chapter is the core of the present thesis. From Chapter 1, we have proposed some optimizations to diminish the number of needed kernels when computing several interaction processes at the same time. At the end of Chapter 1, we have seen that Crossing-symmetries had disadvantages with respect of the Monte-Carlo integration context. We will now make use of strategic computational methods to deal with the two other propositions, which we have called Flavour optimization and C-symmetry inspired optimization. In particular, we will discuss the physical validity of the aforementioned optimization methods, by highlighting the physical hypothesis upon which they are based. This is an important step to maintain model flexibility of `MadGraph` and letting users choose not to use our implemented methods in case of needing other physical hypothesis. Lastly, we will show computational verifications by comparing the optimized squared matrix amplitudes obtained by our modified algorithms with those of current `MadGraph`'s code.

All contributed optimizations presented in the current chapter will be dealt with at the Fortran code level, and not at the automatized diagram generation algorithms provided by `MadGraph`'s Python metacode. To do this, we will restrict our attention to `standalone` outputs, the objective being to show mainly proof-of-concept.

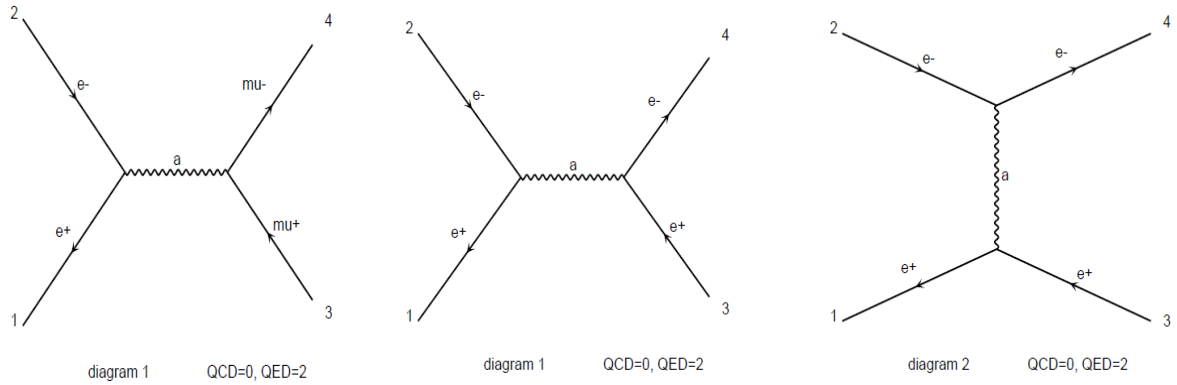
As a side remark, at the start of this work, we began to use Fortran objects in order to simplify some coding tasks by redefining all `HELAS` routines in terms of the object-like structure `aloha_object` presented in Chapter 2. Just changing the Fortran code to include these object-like structures yields by itself some surprising result in diagram generation efficiency[1], and will most likely become an integral part of the next versions of `MadGraph`.

3.1 Flavour optimization

Let's consider the purely QED process $e^+e^- \to l^+l^-$, using the multiparticle label $l^\pm := e^\pm, \mu^\pm$. In SM and using high-energy approximations, it is usual to compute

[1] These gains are currently unexplained

these processes as if the leptons were massless. In this approximation, particles $\mu^{\pm}$ and $e^{\pm}$ have the same spin, colour (as they are colourless), and (negligeable) mass. This means that two Feynman diagrams sharing the same topology but differing solely on the lepton flavours must have the exact same amplitude contribution to their respective total squared matrix elements. Figures 3.1a and 3.1b show the Feynman diagrams generated from the multiparticle process $e^+e^- \to l^+l^-$, where we have intentionally ignored any non-QED process (effectively excluding Z boson exchanges). We have chosen this textbook example as there are no colour algebra considerations in purely QED processes.



(a) $e^+e^- \to \mu^+\mu^-$ (b) $e^+e^- \to e^+e^-$

Figure 3.1: `MadGraph`'s generated tree-level Feynman diagrams for $e^+e^- \to l^+l^-$. Both processes share the same s-channel diagram structure, while $e^+e^- \to e^+e^-$ has an additional t-channel.

For `MadGraph`, this induces a certain redundancy of computed amplitudes, as the exact same calculation of the s-channel has been done twice. This double amplitude computation exists because each Fortran matrix file follows the same building steps as in Table 2.3; meaning that amplitudes corresponding to each subprocess are computed channel-by-channel, and then their overall contribution is summed (or projected over colour basis for QCD interactions) before looping through helicities. For the s-channels, the same `HELAS` routines with same arguments were called. As we have two different matrix files, the information about the equal amplitudes gets lost in the squared matrix elements calculations.

We propose a simple yet effective way of exploiting the similarities between these two processes. Our aim is to show proof-of-concept that we could have some type of flag that would control the flavour of the interactions, and hence be able to call the necessary `HELAS` routines for either the $e^+e^- \to \mu^+\mu^-$ or the $e^+e^- \to e^+e^-$ processes.

Define a global integer variable `flv` $= \pm 1$, which we define such as when `flv` $= +1$, the computed matrix element corresponds to the $e^+e^- \to \mu^+\mu^-$ subprocess and when

`flv` $= -1$ it corresponds to $e^+e^- \to e^+e^-$. Then it suffices to implement the following code:

```
#Global variable
flv = ±1

# Start by calling all external particles
W(1) = outcoming_fermion(p_1, hel_1, sf = -1)
W(2) = incoming_fermion(p_2, hel_2, sf = +1)
W(3) = incoming_fermion(p_3, hel_3, sf = -1)
W(4) = outcoming_fermion(p_4, hel_4, sf = +1)

#s-channel (common diagram for both flavours)
W(5) = FFV_3(W(2),W(1))
Amp_s = FFV_0(W(3),W(4),W(5))

if flv == -1 :
    #t-channel (only for e+e- > e+e-)
    W(6) = FFV_3(W(3),W(1))
    Amp_t = FFV_0(W(2),W(4))
```

Figures 3.2, 3.3 and 3.4 show `MadGraph`'s validation for the pseudocode above, by showing respectively the squared matrix elements for the $e^+e^- \to \mu^+\mu^-$, the $e^+e^- \to e^+e^-$ and the flavour-optimized `flv = -1` $e^+e^- \to l^+l^-$ processes.

We now have to ask about the scalability of this optimization. Leaving `FLV` as a global variable presents the challenge of requiring the `ALOHA` module to set each time these cumbersome "if-statements", rendering the automatization not very straightforward. A more complex process involving a larger set of possible leptons would require to multiply the amount of these "if statements". This is all the more bothersome for GPU usage perspectives, as they may hinder parallelization.

```
e+ e- > mu+ mu-
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    1.0426354430749244E-002  GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.2: `MadGraph`'s computed $e^+e^- \to \mu^+\mu^-$ squared matrix element

```
e+ e- > e+ e-
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    4.5858376851767697E-002  GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.3: `MadGraph`'s computed $e^+e^- \to e^+e^-$ squared matrix element

```
Flavour optimization flv = -1  e+e- > l+l-
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    1.0426354430749244E-002  GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.4: `MadGraph`'s computed $e^+e^- \to l^+l^-$ squared matrix element with `flv = +1`

Nonetheless, as simple as this implementation may be in this example, it has lead to an improved version, where flavours are coded internally and not globally. This was not done through the `aloha_objects` structures introduced in Chapter 2, but by setting flavours inside coupling constants of the model. Hence, during diagram generation, all possible vertices between external and off-shell particles are build from a "flavour matrix" where every coupling constant has a real positive value[2] for valid vertices or it is equal to zero for non-valid ones. Following our example, we would have the (μ^+, μ^-, γ) vertex as a real positive coupling, while (μ^+, e^+, γ) that violates flavour conservation, would be zero. There are current branchs of `MadGraph`'s github repository exploring these optimizations.[3]

3.2 C-symmetry inspired optimization

Let's consider once more the familiar example of $gu \to gu$ at the high-energy approximation and the SM framework. From theory, we know that the discrete transformation of charge conjugation leaves both the QED and QCD Lagrangians invariant [10] [**?**].

[2] equal to its usual coupling constant value for the prescribed interaction α, g_s, g_W, etc.
[3] https://github.com/mg5amcnlo/mg5amcnlo/tree/3.6.1_aloha_obj

Therefore, we expect $g\tilde{u} \to g\tilde{u}$ to yield the same squared matrix element, computed from tree-level interactions, as $gu \to gu$. This is confirmed easily by generating both processes in `MadGraph` as shown in Figures 3.5 and 3.6.

```
 g u > g u
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    8.4448739525537402       GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.5: `MadGraph`'s computed $gu \to gu$ squared matrix element.

```
 g u~ > g u~
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    8.4448739525537402       GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.6: `MadGraph`'s computed $g\tilde{u} \to g\tilde{u}$ squared matrix element.

Some remarks regarding why there is currently a lack of optimization on these processes are in order. From a theory point of view, SM is not a local theory that preserves C-symmetry in general, and therefore it is natural for `MadGraph`to not assume it. We do not want to rely on it for our optimization either. From a technical point of view, as presented in Table 2.1 when generating processes encompassing particles and antiparticles, the called fermion wavefunctions routines are different. Specifically `incoming_fermion` and `outcoming_fermion` subroutines, used to introduce external on-shell fermions, do not act in the same way.

To see how the `HELAS` routines relate to each other, see the following Table 3.1:

Pseudocode $gu \to gu$	**Pseudocode** $g\tilde{u} \to g\tilde{u}$
# Start by calling all external particles	
`W(1) = vector(p_1, hel_1)`	`W(1) = vector(p_1, hel_1)`
`W(2) = incoming_fermion(p_2, hel_2, sf = +1)`	`W(2) = outcoming_fermion(p_2, hel_2, sf = -1))`
`W(3) = vector_star(p_3, hel_3)`	`W(3) = vector_star(p_3, hel_3)`

Pseudocode (continued)	**Pseudocode (continued)**
`W(4) = outcoming_fermion(p_4, hel_4, sf = +1))`	`W(4) = incoming_fermion(p_4, hel_4, sf = -1)`
# s-channel	
`W(5) = FFV_2(W(2) , W(1))`	`W(5) = FFV_1(W(2) , W(1))`
`Amp_s = FFV_0(W(5), W(4), W(3))`	`Amp_s = FFV_0(W(4), W(5), W(3))`
# u-channel	
`W(6) = VVV_1(W(1) , W(3))`	`W(6) = VVV_1(W(1) , W(3))`
`Amp_u = FFV_0(W(2), W(4), W(6))`	`Amp_u = FFV_0(W(4), W(2), W(6))`
# t-channel	
`W(7) = FFV_1(W(4) , W(1))`	`W(7) = FFV_2(W(4) , W(1))`
`Amp_t = FFV_0(W(2), W(7), W(3))`	`Amp_t = FFV_0(W(7), W(2), W(3))`

Table 3.1: Comparison of pseudocode for $gu \to gu$ and $g\tilde{u} \to g\tilde{u}$. We have coloured in blue all ket-like and in red all bra-like fermion wavefunctions routines.

To complete the picture of all parallels between those two processes, we also show `MadGraph` generated Feynman diagrams in Figures 3.7 and 3.8[4].

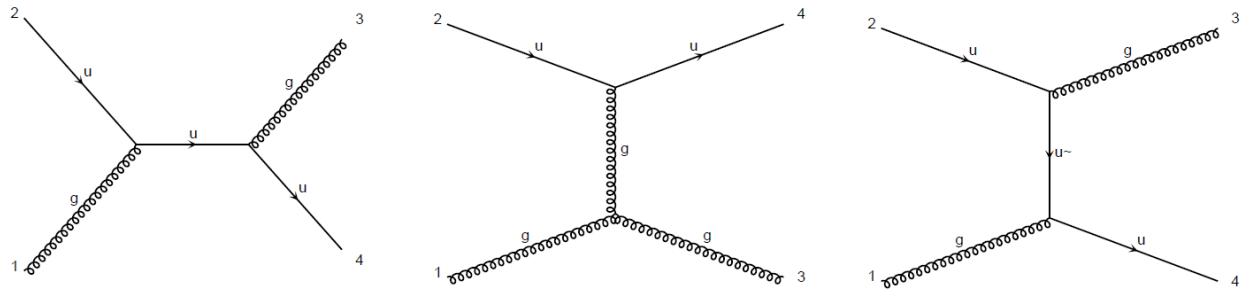


Figure 3.7: `MadGraph`'s generated tree-level Feynman diagrams for the $gu \to gu$ process

[4]These are the same s, u and t-channels as previously presented. `MadGraph` does not follow a precise order of initial state and final state particle labels.

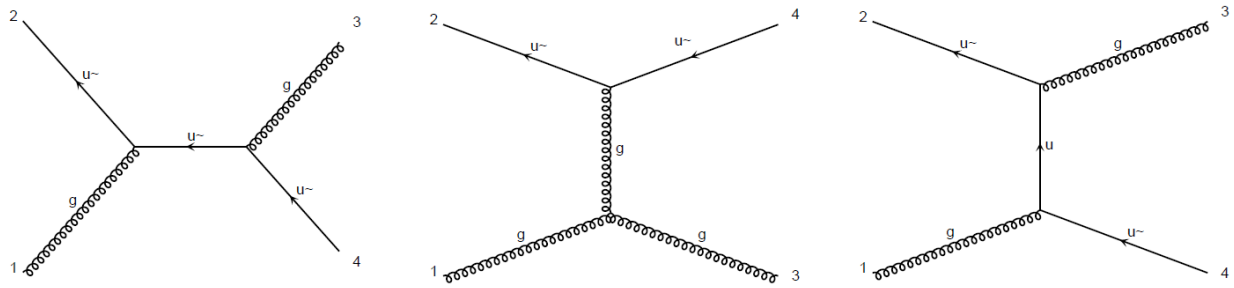


Figure 3.8: `MadGraph`'s generated tree-level Feynman diagrams for the $g\tilde{u} \to g\tilde{u}$ process

Topologically identical diagrams that differ only by the direction of a single single fermionic flow are treated as completely different diagrams by the algorithm, and hence require for two kernels to be computed.

In order to perform a C-symmetry-like transformation we basically want to be able to exchange:

$$\begin{aligned} u(p) &\rightleftharpoons \bar{v}(p) \\ v(p) &\rightleftharpoons \bar{u}(p) \end{aligned} \tag{3.1}$$

and have all necessary vertices and propagators in the matrix element to be well suited after the exchange. Note that what we are considering here is not exactly C-symmetry, as a charge conjugation transformation of the solutions of a relativistic gauge wave equation. We are aiming to take advantage of the almost identical diagrams subamplitudes and map one onto the other, which is why we call our optimization as solely C-symmetry *inspired*. This fits well our intended purposes from Chapter 1, since this mapping preserves the same channel structures and wouldn't bother the SDE integration.

To show proof-of-concept of this optimization, our goal is to be able to have a new global variable `flip` $= \pm 1$ which could be assigned automatically when evaluating the squared matrix elements for both processes, such that, when `flip` $= +1$ we evaluate $gu \to gu$ and when `flip` $= -1$ we evaluate $g\tilde{u} \to g\tilde{u}$. What is important is that we can obtain the squared matrix element value from a same matrix element algorithm.

Our strategy will be as follows. From Table 3.1 we note that the main differences between both pseudocodes are the bra or ket-like nature of each external fermions coded through `aloha_object` wavefunction routines. Therefore, we introduce a new local variable called `brkt` inside the `aloha_object`

```
class aloha_object:
    wf: list of 4 complex numbers
    p: list of 4 real numbers (indexed 0 to 4)
    brkt: integer (either +1 or -1)
```

where we define `brkt` as +1 when the object is a ket-like wavefunction and -1 when it is bra-like. Doing so passes the information locally inside the wavefunctions routines, allowing it to be propagated through the code. There is an argument to be made on the reason for granting this local condition, similarly to the Flavour optimization case. First, if there are more than one fermionic flow, this could give us control to handpick which fermionic flows to invert. Secondly, it has the scalability advantage of not needing global "if statements".

Another important point of the strategy is that, since the exchanges proposed in Equation 3.1 require to call either the `incoming_fermion` routine or the `outcoming_fermion` routine, we have to make a small concession and define another new global integer variable called `flow` which carries the information of if a fermion flow is incoming or outcoming towards the interaction (see the description of Table 2.1). Either way, the fact that the external particle was in or outcoming was hard-coded in the names of the routines, so we have just added an extra small step easily automatable from `ALOHA`.

We can now redefine the `incoming_fermion`, `outcoming_fermion` incorporating them into a single routine `new_in_out_fermion` depending on `flip`:

```
def ψ = new_in_out_fermion(p, hel, flip*sf, flip*flow):
    if (flip*flow == 1) :
        ψ = incoming_fermion(p, hel, flip*sf)
        ψ.brkt = +1
        #ψ has gained the attribute ket-like
    elseif (flip*flow == -1) :
        ψ = outcoming_fermion(p, hel, flip*sf)
        ψ.brkt = -1
        #ψ has gained the attribute bra-like
    else
        print("error")
```

Figure 3.9 shows diagrammatically the possible outcomes of the `new_in_out_fermion` routine.

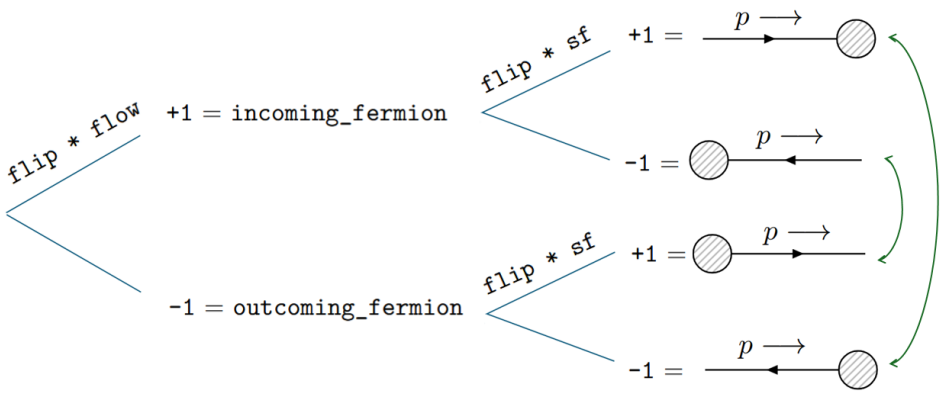


Figure 3.9: Diagram of the the `new_in_out_fermion` routine. Changing `flip` $= \pm 1$ exchanges the called wavefunction as shown by the two green arrows.

In a similar fashion we can now merge the vertices `FFV_1` and `FFV_2` routines from Table 2.2, as it is now possible to calculate only the suitable resulting vertex and propagator depending on the bra or ket-like attribute of the `aloha_object` argument.

```
def ψ = new_FFV( ψ_arg , ε_3^μ ):
    if ψ_arg.brkt == +1 :
        ψ = FFV_2( ψ_arg , ε_3^μ )
        ψ.brkt = +1
        #ψ has gained the attribute ket-like

    else if ψ_arg.brkt == -1 :
        ψ = FFV_1( ψ_arg , ε_3^μ )
        ψ.brkt = -1
        #ψ has gained the attribute bra-like
```

The resulting `aloha_object` of the `new_FFV` routine propagates the bra or ket-like attribute of its argument.

Finally, to compute each Feynman diagram subamplitude, we can make sure that `FFV_0` has the correct ordering of arguments by defining

```
def Amp = new_FFV_0( ψ_1 , ψ_2 , ε_3^μ ):
    if ψ_1.brkt == +1 and ψ_2.brkt == -1 :
        Amp = FFV_0( ψ_1 , ψ_2 , ε_3^μ )

    else if ψ_1.brkt == -1 and ψ_2.brkt == +1 :
        Amp = FFV_0( ψ_2 , ψ_1 , ε_3^μ )

    else
        print("error")
```

This concludes the computation of each Feynman diagram subamplitudes. We have not changed how the amplitudes were computed, we have just succesfully propagated the bra-like or ket-like nature of the external fermions and adapted accordingly the `HELAS` routines. As an important remark, note that our proposed optimization needs not C-symmetry to hold at a Lagrangian level for the analyzed process. A key difference is that the charge conjugation transformation implies for helicities to get mapped as $u_+(p) \rightarrow v_-(p)$ [**?**], while what we're proposing acts as $u_+(p) \rightarrow \bar{v}_-(p)$. Nonetheless, as we perform our computation of the squared matrix element using Equation 1.2, there is an average over all helicities, and while the incoming beams are unpolarized (as they usually are at LHC), we need not to worry about helicities exchanges[5].

If we use the code as it is for a QED example, such as for transforming the process $\gamma e^- \rightarrow \gamma e^-$ into $\gamma e^+ \rightarrow \gamma e^+$, the resulting squared matrix amplitudes would be the same. However, the QCD case is not as simple, as one has to be careful with the coloured ordered amplitudes projections.

3.2.1 Colour algebra considerations

The results of the current section are general enough to be applied to any QCD process containing only one QCD fermionic flow and containing a k number of gluons. Nonetheless, to not hinder the notation we will keep stating the results as if they were limited only to the $gu \rightarrow gu$ and $g\tilde{u} \rightarrow g\tilde{u}$ C-symmetry inspired transformation. Let's start by defining some notation:

- A_i : the i-th Feynman diagram subamplitude for $gu \rightarrow gu$. When implementing the last section optimizations, these would be the amplitudes algorithmically computed by setting `flip = +1`.
- B_i : the i-th Feynman diagram subamplitude for $g\tilde{u} \rightarrow g\tilde{u}$. When implementing the last section optimizations, these would be the amplitudes algorithmically computed by setting `flip = -1`.

Naively trying to project the B_i amplitudes over the same basis elements T^σ as for the unmodified amplitudes A_i yields wrong results, as shown numerically in Figure 3.10, where we can clearly spot a difference relative to the expected value presented in Figure 3.8.

After analyzing `MadGraph`'s outputs, we have arrived empirically at two different ansatz that will allow us to correctly project the coloured ordered amplitudes for the C-symmetry inspired transformed process without needing to recalculate the coloured ordered amplitudes of the process $g\tilde{u} \rightarrow g\tilde{u}$. We will proceed to present both ansatz and prove them.

[5] if needed, we can recast to have the exact same helicities mapped into one another, by setting `hel*flip` when calling `new_in_out_fermion`.

```
g ux > g ux projected over basis of g u > g u
Phase space point:

------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
------------------------------------------------------------------------
Matrix element =    3.6134341484319403       GeV^          0
------------------------------------------------------------------------
```

Figure 3.10: `MadGraph`'s computed squared matrix element resulting of naive projection of B_i amplitudes over colour element basis of $gu \to gu$.

Using Equation 2.15 we define and set the notation for the coloured ordered amplitudes. Let

$$J^\sigma := \sum_{i=1}^{m} a_i^\sigma \times A_i \,, \tag{3.2}$$

be the coloured ordered amplitude for the $gu \to gu$ process.

Then let

$$\bar{K}^\sigma := \sum_{i=1}^{m} b_i^\sigma \times B_i \,, \tag{3.3}$$

be the coloured ordered amplitude for the $g\tilde{u} \to g\tilde{u}$ process

Lastly let's define,

$$\tilde{J}^\sigma := \sum_{i=1}^{m} \tilde{a}_i^\sigma \times B_i \,, \tag{3.4}$$

as the coloured ordered amplitudes for the transformed process, which computes $g\tilde{u} \to g\tilde{u}$ matrix element based on $gu \to gu$. Our only objective is to be able to find the correct complex coefficients $\tilde{a}_i^\sigma = \tilde{a}_i^\sigma(a_i^\sigma)$ such that

$$\sum_{\sigma,\sigma'} K^\sigma C_F^{\sigma\sigma'} K^{*\sigma'} = \sum_{\sigma,\sigma'} \tilde{J}^\sigma C_F^{\sigma\sigma'} \tilde{J}^{*\sigma'} \,. \tag{3.5}$$

Proposition 3.2.1. Let σ be an element of the permutation group S_k. Let $r(\sigma)$ be its "reversed" element, meaning that if $\sigma = (1, 2, ..., k-1, k)$ then $r(\sigma) = (k, k-1, ..., 2, 1)$. The complex coefficients $\tilde{a}_i^\sigma = a_i^{*r(\sigma)}$ and $\tilde{a}_i^\sigma = a_i^{*\sigma}$ are solutions to Equation 3.5.

There is an important remark to me made here, as we are assuming that elements of the colour matrix are the same for both the $gu \to gu$ and the $g\tilde{u} \to g\tilde{u}$ processes. This fact follows both from definition in Equation 2.3 but also, from the symmetric property of C_F proven in 2.23. Indeed, when evaluating the colour object associated to a Feynman diagram subamplitude with at most one QCD fermionic flow, the order in

which the generators of $\mathfrak{su}(3)$ appear will either be σ, in the particle case, or $r(\sigma)$, in the antiparticle case. Therefore, we must have as a condition that $C^{\sigma\rho} = C^{r(\sigma)(\rho)}$, which will be proven right next in Equation 3.11.

Let's first show that both ansatz are equivalent.

See that for any $\sigma \in S_k$, hermicity of each t^a matrix implies:

$$T^{\sigma *}_{ij} = (t^{\sigma(1)} ... t^{\sigma(k)})^*_{ij} = (t^{\sigma(1)*} ... t^{\sigma(k)*})_{ij} = (t^{\sigma(k)} ... t^{\sigma(1)})_{ji} = T^{r(\sigma)}_{ji} \ . \tag{3.6}$$

Then, we can state that, from definition,

$$C^{\sigma\rho}_F = T^{\sigma}_{ij} T^{\rho *}_{ij}, \tag{3.7}$$

and by applying the result from Equation 3.6 to each colour basis element,

$$= T^{\sigma}_{ij} T^{r(\rho)}_{ji} \ , \tag{3.8}$$

$$= T^{r(\sigma)*}_{ji} T^{r(\rho)}_{ji} \ . \tag{3.9}$$

Then, by retaking the definition of C_F,

$$= C_F^{r(\rho)r(\sigma)} \ . \tag{3.10}$$

Lastly by the symmetric property that was shown in Equation 2.23,

$$= C_F^{r(\sigma)r(\rho)} \ . \tag{3.11}$$

If we now call $|M_1|^2$ the expression of the squared matrix element computed from coloured ordered amplitudes as in Equation 2.3 for the first ansatz, and $|M_2|^2$ the one for the second ansatz, we get:

$$|M_2|^2 = \sum_{\sigma\rho} \sum_{ij} a^{\sigma *}_j B_j \times a^{\rho}_i B^*_i \times C^{\sigma\rho}_F \ , \tag{3.12}$$

by changing the dummy summing variables $\sigma \to r(\sigma)$ and $\rho \to r(\rho)$

$$= \sum_{r(\sigma)r(\rho)} \sum_{ij} a^{r(\sigma)*}_j B_j \times a^{r(\rho)}_i B^*_i \times C_F^{r(\sigma)r(\rho)} \ , \tag{3.13}$$

then, lastly by making use of the result in Equation 3.11,

$$= \sum_{r(\sigma)r(\rho)} \sum_{ij} a^{r(\sigma)*}_j B_j \times a^{r(\rho)}_i B^*_i \times C^{\sigma\rho}_F \ , \tag{3.14}$$

$$= |M_1|^2 \ . \tag{3.15}$$

This equivalence between the two ansatz is very useful for us, as we will build the proof of the proposition by using the first ansatz but, algorithmically speaking, it is much

easier to implement the second one. Indeed, as it simply requires to take the complex conjugate of the coefficients a_i^σ, we could define a new global variable equal to either the complex unit i or $-i$ and then multiply it at some strategic points of the code to choose between projecting over the bases for $gu \to gu$ or $g\tilde{u} \to g\tilde{u}$.

Proof. (of Proposition 3.2.1) At tree-level, any process with only one QCD fermion flow can be drawn as a fermionic line to which every gluon or set of interacting gluons are connected. We don't have to worry as if the external gluons are from the initial or final states, as the coloured objects associated are the same in both cases.

We will build a proof by induction.

Step 1 - Initialization:

We start by considering the case for which gluons or sets of gluons are connected exactly once to the fermionic line.

For each case, we will consider diagrams with same topology for both a particle and an antiparticle (basically reversing the fermionic flow). We recall that A is the colourless Feynman diagram subamplitude for the particle and B is the one for the antiparticle. We also call M_{kg} the coloured matrix element for the particle with k number of gluons and N_{kg} the one for the antiparticle. Figure 3.11 shows all diagrams for the first cases, with k=1,2,3.

Suppose $k = 1$, meaning that the fermionic flow is connected to only one gluon, then

$$M_{1g} = A \times 1 \times t^a_{ij}\,, \tag{3.16}$$
$$N_{1g} = B \times 1 \times t^a_{ji}\,. \tag{3.17}$$

Hence for $k = 1$ we have that for the unique permutation $\sigma \in S_1$, the relation $\tilde{a}^{\sigma *} = 1 = b^{r(\sigma)}$ trivially holds.

Suppose $k = 2$, using the structure constants relation from Equations 2.5, we obtain

$$M_{2g} = A \times f^{abz} t^z_{ij} = A \times (-i(t^a t^b)_{ij} + i(t^b t^a)_{ij})\,, \tag{3.18}$$
$$N_{2g} = B \times f^{abz} t^z_{ij} = B \times (-i(t^a t^b)_{ji} + i(t^b t^a)_{ji})\,. \tag{3.19}$$

We have that for all permutations $\sigma = (ab)$, $\rho = (ba) \in S_2$, the relations $\tilde{a}^{(ab)*} = i = b^{(ba)}$ and $\tilde{a}^{(ba)*} = -i = b^{(ab)}$ hold.

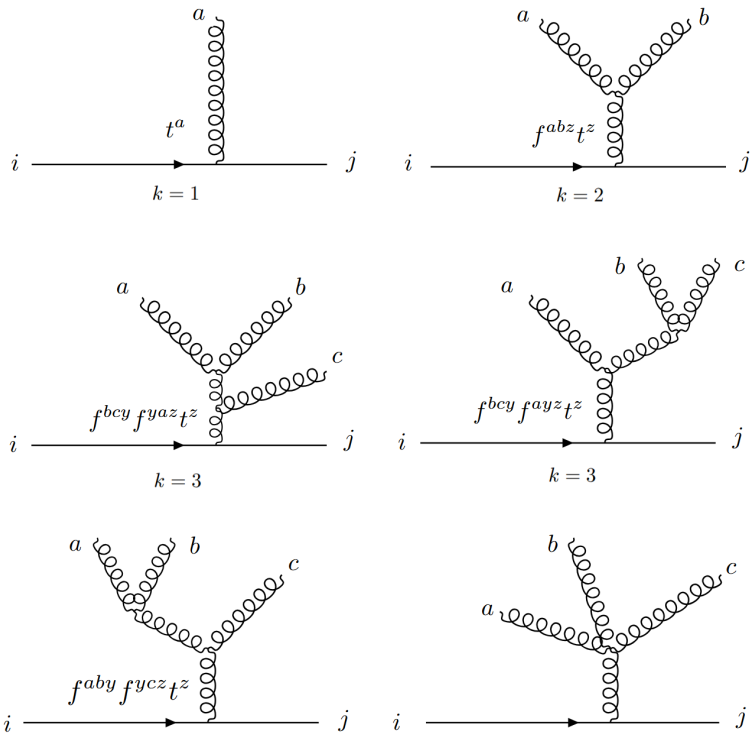


Figure 3.11: Colour objects associated to Feynman diagrams with $k = 1, 2, 3$ number of gluons and exactly one gluon-fermion vertex.

Lastly suppose $k = 3$. There are four possible diagrams associated with 3 gluons. Three of them contain two three-point vertices, which relate to each other by a minus sign, because the structure constants are totally antisymmetric. This is given by $f^{cby}f^{yaz}t^z_{ij} = -f^{bcy}f^{yaz}t^z_{ij} = f^{bcy}f^{ayz}t^z_{ij}$. The fourth case, which contains a four-point vertex, for our intends and purposes can be considered as equivalent as having the other three $k = 3$ configurations at the same time, as explained in Table 2.4. Then, without lost of generality, we can choose the first one presented in Figure 3.11:

$$M_{3g} = A \times f^{bcy}f^{yaz}t^z_{ij} = A \times -if^{bcy}[t^y_{ij}, t^a_{ij}] \ , \tag{3.20}$$

$$= A \times \left(+(t^at^bt^c)_{ij} - (t^at^ct^b)_{ij} - (t^bt^ct^a)_{ij} + (t^ct^bt^a)_{ij}\right) \ , \tag{3.21}$$

$$N_{3g} = B \times f^{bcy}f^{yaz}t^z_{ji} = B \times -if^{bcy}[t^y_{ji}, t^a_{ji}] \ , \tag{3.22}$$

$$= B \times \left(+(t^at^bt^c)_{ji} - (t^at^ct^b)_{ji} - (t^bt^ct^a)_{ji} + (t^ct^bt^a)_{ji}\right) \ . \tag{3.23}$$

We have here that for all permutations $\sigma \in S_3$, the relations

$$\tilde{a}^{(abc)*} = +1 = b^{(cba)} \ , \tag{3.24}$$

$$\tilde{a}^{(acb)*} = -1 = b^{(bca)} \ , \tag{3.25}$$

$$\tilde{a}^{(bca)*} = -1 = b^{(acb)} \ , \tag{3.26}$$

$$\tilde{a}^{(cba)*} = +1 = b^{(abc)} \ , \tag{3.27}$$

$$\tilde{a}^{(bac)*} = 0 = b^{(cab)} \ , \tag{3.28}$$

$$\tilde{a}^{(cab)*} = 0 = b^{(bac)} \ , \tag{3.29}$$

hold. Hence for all k=3 possible cases, no matter which of the possible diagrams, we have that $\tilde{a}^{\sigma *} = b^{r(\sigma)}$ holds. Doing up to k=3 cases for the induction's initialization may seem an overshooting, but it has allowed us to explore all possible cases to join gluons together and it has also allowed us to remark that no matter how a new n-th gluon is attached to a set of interacting gluons, the complex coefficients will just vary as powers of i. This property will be proven in the next step.

Step 2 - Induction:

Suppose now that $\tilde{a}^{\sigma *} = b^{r(\sigma)}$ holds for all Feynman diagrams with k gluons and $\sigma \in S_k$.

Let $\sigma = (\sigma_1, z, \sigma_2) \in S_k$ where σ_1 and σ_2 are two smaller permutations of indices that come before and after the index z. Therefore, the corresponding matrix element for the one QCD fermionic flow plus k gluons process is

$$M_{kg} = \sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T^{\sigma}_{ij} \ , \tag{3.30}$$

$$= \sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T^{\sigma_1}_{ik} t^{z}_{kh} T^{\sigma_2}_{hj} \ . \tag{3.31}$$

Then, adding a vertex from a $k+1$th new external gluon to an already present one labeled, labeled with z :

$$M_{(k+1)g} = f^{xyz} \left(\sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T^{\sigma}_{ij} \right) \ , \tag{3.32}$$

$$= \sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T^{\sigma_1}_{ik} f^{xyz} t^{z}_{kh} T^{\sigma_2}_{hj} \ . \tag{3.33}$$

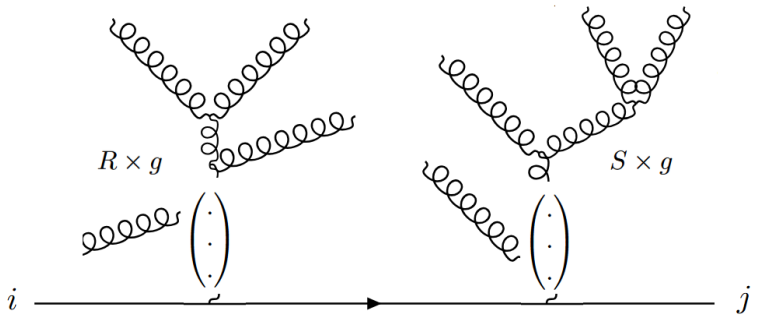


Figure 3.12: A set of R gluons and a set of S gluons.

Using the structure constants definition from Equation 2.5,

$$M_{(k+1)g} = \sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T_{ik}^{\sigma_1} (-i \times [t^x, t^y]_{kh}) T_{hj}^{\sigma_2} \tag{3.34}$$

$$, = \sum_{\sigma \in S_k} A \times \tilde{a}^{\sigma} T_{ik}^{\sigma_1} (-i(t^x t^y)_{kh} + i(t^y t^x)_{kh}) T_{hj}^{\sigma_2} \ , \tag{3.35}$$

$$= \sum_{\sigma \in S_k} A \times (-i\tilde{a}^{\sigma}) T_{ij}^{(\sigma_1, x, y, \sigma_2)} + A(+i\tilde{a}^{\sigma}) T_{ij}^{(\sigma_1, y, x, \sigma_2)} \ . \tag{3.36}$$

On the other hand, if we perform the same sum, but with the change of variable $r(\sigma) = (r(\sigma_2), z, r(\sigma_1)) \in S_k$ for the antifermion case we get:

$$N_{(k+1)g} = \sum_{r(\sigma) \in S_k} B \times (-ib^{r(\sigma)}) T^{(r(\sigma_1), x, y, r(\sigma_2))} + B(+ib^{r(\sigma)})) T^{(r(\sigma_1), y, x, r(\sigma_2)} \tag{3.37}$$

By defining $\sigma' = (\sigma_1, x, y, \sigma_2) \in S_{k+1}$, we can establish the relation $(-i\tilde{a}^{r(\sigma')})^* = (+ib^{\sigma'})$, hence proving the induction.

Step 3 - diagrams with more than only one attached gluon or set of gluons:

Generalizing the past results for diagrams whose topologies encompass any number of gluons is quite straightforward. Let there be two sets of interacting gluons connected at two vertices to the QCD fermionic line, one with R gluons and the other with S, as shown in Figure 3.12, such that for both of them individually the relation from ansatz 1 holds.

Then, the whole diagram matrix element for the particle case will be

$$M_{(S+R)\times g} = A \left(\sum_{\sigma \in S_S} \tilde{a}^{\sigma} T^{\sigma}_{ik} \right) \left(\sum_{\rho \in S_R} \tilde{a}^{\rho} T^{\rho}_{kj} \right) , \tag{3.38}$$

$$= A \left(\sum_{(\sigma,\rho) \in S_{S+R}} \tilde{a}^{(\sigma,\rho)} T^{(\sigma,\rho)}_{ij} \right) , \tag{3.39}$$

$$= A \sum_{\kappa \in S_{S+R}} \tilde{a}^{\kappa} T^{\kappa}_{ij} , \tag{3.40}$$

where we have defined $\kappa = (\sigma, \rho) \in S_{S+R}$. Using the same trick as in the previous step to change the summation dummy variable by using the reverse of the indexes we get, for the antiparticle case,

$$N_{(S+R)\times g} = B \left(\sum_{r(\rho) \in S_R} b^{r(\rho)} T^{r(\rho)}_{jk} \right) \left(\sum_{r(\sigma) \in S_S} b^{r(\sigma)} T^{r(\sigma)}_{ki} \right) , \tag{3.41}$$

$$= B \sum_{r(\kappa) \in S_R} b^{r(\kappa)} T^{r(\kappa)}_{ji}, \tag{3.42}$$

where we have used the fact that $r(\kappa) = (r(\rho), r(\sigma))$. Hence, we have proved that even when there is more than just one vertex to which a set of gluons is connected to the QCD fermionic line, the relationship $\tilde{a}^{\kappa} = b^{r(\kappa)}$ holds for $\kappa \in S_{S+R}$. Note that there will be several values of $\kappa \in S_{S+R}$ for which the complex coefficients will be zero, notably all of them that mix indices belonging to the set of gluons S with indices belonging to the set of gluons R. This result generalizes trivially for any number of sets of gluons interacting with the QCD fermionic line. □

3.2.2 `MadGraph` numerical validations

We can proceed to show `MadGraph` numerical validations for the C-symmetry inspired optimization, for three different cases.

- $gu \to gu$ and $g\tilde{u} \to g\tilde{u}$

```
 g ux > g ux by Symmetry inspired transformation of g u > g u
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.5000000E+03  0.1109243E+03  0.4448308E+03 -0.1995529E+03  0.9725608E-05
4   0.5000000E+03 -0.1109243E+03 -0.4448308E+03  0.1995529E+03  0.1009274E-04
-------------------------------------------------------------------------------
Matrix element =    8.4448739525537402       GeV^          0
-------------------------------------------------------------------------------
```

Figure 3.13: MadGraph's computed squared matrix element resulting of correctly applying C-symmetry inspired transformation to $gu \to gu$.

- $gu \to gug$ and $g\tilde{u} \to g\tilde{u}g$

```
g u > g u g
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.4585788E+03  0.1694532E+03  0.3796537E+03 -0.1935025E+03  0.6607249E-05
4   0.3640666E+03 -0.1832987E+02 -0.3477043E+03  0.1063496E+03  0.7979012E-05
5   0.1773546E+03 -0.1511234E+03 -0.3194936E+02  0.8715287E+02  0.1348699E-05
-------------------------------------------------------------------------------
Matrix element =    2.8517643830888537E-003  GeV^         -2
-------------------------------------------------------------------------------
```

Figure 3.14: MadGraph's computed squared matrix element of $gu \to gug$.

```
g ux > g ux g by C-symmetry inspired transformation of g u > g u g
 Phase space point:

-------------------------------------------------------------------------------
n       E             px             py             pz             m
1   0.5000000E+03  0.0000000E+00  0.0000000E+00  0.5000000E+03  0.0000000E+00
2   0.5000000E+03  0.0000000E+00  0.0000000E+00 -0.5000000E+03  0.0000000E+00
3   0.4585788E+03  0.1694532E+03  0.3796537E+03 -0.1935025E+03  0.6607249E-05
4   0.3640666E+03 -0.1832987E+02 -0.3477043E+03  0.1063496E+03  0.7979012E-05
5   0.1773546E+03 -0.1511234E+03 -0.3194936E+02  0.8715287E+02  0.1348699E-05
-------------------------------------------------------------------------------
Matrix element =    2.8517643830888537E-003  GeV^         -2
-------------------------------------------------------------------------------
```

Figure 3.15: MadGraph's computed squared matrix element resulting of correctly applying C-symmetry inspired transformation to $gu \to gug$.

Conclusion

This thesis addressed the problem of computational cost in squared matrix element evaluations for certain family of multiprocess collider simulations. By analyzing thoroughly `HELAS` routines, we implemented two concrete optimizations at the level of standalone Fortran output: the Flavour optimization and the C-symmetry inspired optimization. We had to set aside the Cross-symmetry optimization as it was conflicting with the SDE integration method. Nonetheless, the implemented optimizations did manage to preserve the channels of their base Feynman diagrams, allowing them to not require for new integrators in SDE.

The Flavour optimization was a simple one, yet it managed to prove useful enough to be adapted largely to a current branch of `MadGraph` where fermion flavours are encoded inside coupling constant matrices.

We have also succesfully exploited some properties of the colour matrix C_F that enabled us to recast coloured ordered amplitudes into appropriate colour basis elements when exchanging a QCD fermion by its respective antifermion. This implementaiton is for now limited to processes where there is only one fermionic current. Nonetheless, all exploited properties came directly from generators of the $\mathfrak{su}(3)$ algebra.

We suspect that as these properties are general enough to encompass more complicated simulations, even maybe NLO ones. Hence, we expect that a more general C-symmetry inspired optimization could be coded.

Table 3.1 shows the obtained results compared to what was expected from Table 1.1 in Chapter 1. We can see that the lack of optimization allowing us to work with two fermionic currents takes a toll on the expected kernel reductions.

While this work focuses on proof-of-concept implementations, we hope that the kernel reductions from the proposed optimizations suggest a direction toward GPU-friendly architecture for `MadGraph`. Further extensions could consider automating these optimizations thanks to `ALOHA`, and generalizing them to next-to-leading order (NLO) computations.

Overall, this work highlights how insights from theoretical physics such as symmetries, group theory, and simple diagrammatic analysis can translate into tangible improvements in simulation software.

Default kernels	Flavour optimized kernels	C-inspired optimized kernels
$gg \to gg$	$gg \to gg$	$gg \to gg$
$gg \to u\tilde{u}$	$gg \to u\tilde{u}$	$gg \to u\tilde{u}$
$u\tilde{u} \to gg$	$u\tilde{u} \to gg$	$u\tilde{u} \to gg$
$gu \to gu$	$gu \to gu$	$gu \to gu$
$g\tilde{u} \to g\tilde{u}$	$g\tilde{u} \to g\tilde{u}$	
$u\tilde{c} \to u\tilde{c}$		
$uc \to uc$	$uu \to uu$	$uu \to uu$
$uu \to uu$		
$u\tilde{u} \to u\tilde{u}$	$u\tilde{u} \to u\tilde{u}$	$u\tilde{u} \to u\tilde{u}$
$u\tilde{u} \to c\tilde{c}$		
$\tilde{u}\tilde{c} \to \tilde{u}\tilde{c}$	$\tilde{u}\tilde{u} \to \tilde{u}\tilde{u}$	$\tilde{u}\tilde{u} \to \tilde{u}\tilde{u}$
$\tilde{u}\tilde{u} \to \tilde{u}\tilde{u}$		

Table 3.1: List of kernels created by `MadGraph` for the $pp \to jj$ process and reduced list of necessary kernels after applying our successfully implemented optimizations.

Bibliography

[1] Johan Alwall, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. MadGraph 5 : Going Beyond. *JHEP*, 06:128, 2011.

[2] T. Aoyama, N. Asmussen, M. Benayoun, J. Bijnens, T. Blum, M. Bruno, I. Caprini, C.M. Carloni Calame, M. Cè, G. Colangelo, F. Curciarello, H. Czyż, I. Danilkin, M. Davier, C.T.H. Davies, M. Della Morte, S.I. Eidelman, A.X. El-Khadra, A. Gérardin, D. Giusti, M. Golterman, Steven Gottlieb, V. Gülpers, F. Hagelstein, M. Hayakawa, G. Herdoíza, D.W. Hertzog, A. Hoecker, M. Hoferichter, B.-L. Hoid, R.J. Hudspith, F. Ignatov, T. Izubuchi, F. Jegerlehner, L. Jin, A. Keshavarzi, T. Kinoshita, B. Kubis, A. Kupich, A. Kupść, L. Laub, C. Lehner, L. Lellouch, I. Logashenko, B. Malaescu, K. Maltman, M.K. Marinković, P. Masjuan, A.S. Meyer, H.B. Meyer, T. Mibe, K. Miura, S.E. Müller, M. Nio, D. Nomura, A. Nyffeler, V. Pascalutsa, M. Passera, E. Perez del Rio, S. Peris, A. Portelli, M. Procura, C.F. Redmer, B.L. Roberts, P. Sánchez-Puertas, S. Serednyakov, B. Shwartz, S. Simula, D. Stöckinger, H. Stöckinger-Kim, P. Stoffer, T. Teubner, R. Van de Water, M. Vanderhaeghen, G. Venanzoni, G. von Hippel, H. Wittig, Z. Zhang, M.N. Achasov, A. Bashir, N. Cardoso, B. Chakraborty, E.-H. Chao, J. Charles, A. Crivellin, O. Deineka, A. Denig, C. DeTar, C.A. Dominguez, A.E. Dorokhov, V.P. Druzhinin, G. Eichmann, M. Fael, C.S. Fischer, E. Gámiz, Z. Gelzer, J.R. Green, S. Guellati-Khelifa, D. Hatton, N. Hermansson-Truedsson, S. Holz, B. Hörz, M. Knecht, J. Koponen, A.S. Kronfeld, J. Laiho, S. Leupold, P.B. Mackenzie, W.J. Marciano, C. McNeile, D. Mohler, J. Monnard, E.T. Neil, A.V. Nesterenko, K. Ottnad, V. Pauk, A.E. Radzhabov, E. de Rafael, K. Raya, A. Risch, A. Rodríguez-Sánchez, P. Roig, T. San José, E.P. Solodov, R. Sugar, K. Yu. Todyshev, A. Vainshtein, A. Vaquero Avilés-Casco, E. Weil, J. Wilhelm, R. Williams, and A.S. Zhevlakov. The anomalous magnetic moment of the muon in the standard model. *Physics Reports*, 887:1–166, December 2020.

[3] Gianfranco Bertone, Dan Hooper, and Joseph Silk. Particle dark matter: evidence, candidates and constraints. *Physics Reports*, 405(5–6):279–390, January 2005.

[4] Lance J. Dixon. A brief introduction to modern amplitude methods. In *Theoretical Advanced Study Institute in Elementary Particle Physics: Particle Physics: The Higgs Boson and Beyond*, pages 31–67, 2014.

[5] HEP Foundation, Thea Aarrestad, Simone Amoroso, Markus Julian Atkinson, Joshua Bendavid, Tommaso Boccali, Andrea Bocci, Andy Buckley, Matteo Cacciari,

Paolo Calafiura, et al. Hl-lhc computing review: Common tools and community software. *arXiv preprint arXiv:2008.13636*, 2020.

[6] Takaaki Kajita. Nobel Lecture: Discovery of atmospheric neutrino oscillations. *Rev. Mod. Phys.*, 88(3):030501, 2016.

[7] Fabio Maltoni and Tim Stelzer. MadEvent: Automatic event generation with MadGraph. *JHEP*, 02:027, 2003.

[8] Olivier Mattelaer and Kiran Ostrolenk. Speeding up madgraph5_amc@ nlo. *The European Physical Journal C*, 81(5):435, 2021.

[9] H. Murayama, I. Watanabe, and Kaoru Hagiwara. HELAS: HELicity amplitude subroutines for Feynman diagram evaluations. 1 1992.

[10] Michael E Peskin. *An Introduction to quantum field theory*. CRC press, 2018.

[11] Antonio Riotto and Mark Trodden. R<scp>ecent</scp> p<scp>rogress in</scp> b<scp>aryogenesis</scp>. *Annual Review of Nuclear and Particle Science*, 49(1):35–75, December 1999.

[12] Andrea Valassi, Taylor Childers, Laurence Field, Stephan Hageböck, Walter Hopkins, Olivier Mattelaer, Nathan Nichols, Stefan Roiser, David Smith, Jorgen Teig, et al. Speeding up madgraph5 amc@ nlo through cpu vectorization and gpu offloading: towards a first alpha release. *arXiv preprint arXiv:2303.18244*, 2023.

[13] Stefan Weinzierl. Introduction to monte carlo methods, 2000.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/sc**