Automated loop-level perturbative calculations beyond the Standard Model

Grégoire Uhlrich

N°d'ordre NNT : 2021LYSE1182

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° 52
Ecole Doctorale de Physique et Astrophysique

Spécialité de doctorat : Physique Théorique
Discipline : Physique des Particules

Soutenue publiquement le 01/10/2021, par :
Grégoire Uhlrich

Automatisation des calculs perturbatifs à boucle au-delà du Modèle Standard

---

Devant le jury composé de :

Kraml, Sabine	Directrice de Recherche	Université Grenoble Alpes	Rapporteure
Raklev, Are	Professeur	Université d'Oslo	Rapporteur
Belyaev, Alexander	Professeur	Université de Southampton	Examinateur
Deandrea, Aldo	Professeur	Université Lyon 1	Examinateur
Isidori, Gino	Professeur	Université de Zürich	Examinateur
Mahmoudi, Farvah	Maître de Conférences	Université Lyon 1	Directrice de thèse

Résumé

Le Modèle Standard (MS) de la physique des particules a été formulé théoriquement au cours du vingtième siècle et achevé dans les années 1970. Depuis lors, toutes les particules dans ce modèle ont été observées expérimentalement. La dernière découverte, celle du boson de Higgs en 2012, a confirmé que le MS décrit très bien le monde des particules élémentaires, du moins à des énergies inférieures à $1\ \mathrm{TeV}$. Le moment dipolaire magnétique anomal de l'électron, qui est souvent considéré comme "la prédiction la plus précise de l'histoire de la physique" grâce à des décennies de calculs très complexes, a été prédit jusqu'à dix chiffres significatifs par le MS.

Nous savons cependant que le MS est incomplet et qu'il ne peut pas décrire correctement la physique des particules élémentaires à des énergies très élevées. Les études au-delà du modèle standard (AMS) deviendront de plus en plus importantes dans un futur proche avec l'augmentation rapide de la quantité de données provenant de différentes expériences dans le monde. L'étude complète des modèles AMS est en général une tâche extrêmement longue impliquant des calculs difficiles. En pratique il n'est pas possible de faire des prédictions exhaustives dans ces modèles à la main, en particulier si l'on veut effectuer une comparaison statistique avec les données et le MS.

Nous présentons dans quelle mesure les calculs nécessaire pour la phénoménologie AMS peuvent être entièrement automatisés pour des scénarios AMS généraux à travers la présentation de `MARTY`, le programme C++ que nous avons développé pour relever ce défi, qui peut devenir un outil très puissant pour la phénoménologie AMS dans tous les domaines de la physique des particules. À travers des exemples de calculs dans des cas particuliers qui reposent sur une grande diversité de techniques, nous montrons ensuite que `MARTY` est capable de calculer des amplitudes, amplitudes carrées et coefficients de Wilson dans des scénarios AMS généraux, à l'arbre et à une boucle.

Keywords: Calculs automatiques, Phénoménologie, Au-delà du Modèle Standard, Calculs à boucle, Amplitudes, Coefficients de Wilson

Abstract

The Standard Model (SM) of particle physics was formulated theoretically during the twentieth century and completed in the 1970s. Since then, all the particles in this model have been observed experimentally. The latest discovery was the Higgs boson in 2012 and confirmed that the SM describes very well the world of elementary particles, at least when considering energies below $1\ \mathrm{TeV}$. The anomalous magnetic dipole moment of the electron – which is often credited as "the most accurate prediction in the history of physics" thanks to decades of very involved calculations – was computed up to ten significant digits by the SM.

We know however that the SM is incomplete and cannot describe correctly elementary particle physics at very high energies. Studies Beyond the Standard Model (BSM) will become more and more important in the near future with the rapidly increasing amount of data from different experiments around the world. The full study of BSM models is in general an extremely time-consuming task involving long and difficult calculations. It is in practice not possible to do exhaustive predictions in these models by hand, in particular if one wants to perform a statistical comparison with data and the SM.

We present how the calculations required for BSM phenomenology can be fully automated for general BSM scenarios through the presentation of `MARTY` – the C++ framework that we developed to address this challenge – that can become a very powerful tool for BSM phenomenology in all domains of particle physics. Through particular calculation examples which rely on a wide range of techniques we then present `MARTY`'s ability to calculate amplitudes, squared amplitudes and Wilson coefficients in general BSM scenarios at the tree level and the one-loop level.

Keywords: Automated calculation, Phenomenology, Beyond the Standard Model, One-loop, Amplitudes, Wilson coefficients

Acknowledgments

J'aimerais tout d'abord remercier toute l'équipe de l'IP2I pour son accueil chaleureux, et l'école doctorale de physique et astrophysique (ED PHAST 52) grâce à qui j'ai pu effectuer mon doctorat. Merci à Dany pour son écoute et sa franchise, et à Sylvie pour son implication dans le bien-être des doctorants.

Merci à tous les membres du jury d'avoir accepté cette responsabilité, et à ceux qui ont fait le déplacement quand la situation sanitaire le permettait. Merci également aux rapporteurs qui ont pris le temps de lire cette thèse et de proposer des corrections pour le manuscrit.

Je suis très reconnaissant d'avoir pu travailler sur le sujet qui a été le mien, et qui correspondait parfaitement à ce sur quoi j'avais envie de réfléchir pendant trois années. Cela n'aurait pas été possible sans le soutien et la confiance de Nazila, ma directrice de thèse, que je remercie donc tout particulièrement. Un grand merci aussi à Alexandre, tous les deux ont suivi de très près ce projet et ont toujours su me conseiller avec pertinence, gentillesse et bonne humeur.

Un grand merci également aux premiers utilisateurs de `MARTY` qui m'ont fait des retours précieux pour faire des ajustements nécessaires, et notamment Arnab, Marco et Amine. Merci pour leur temps, leur patience et leur confiance.

Je me dois également de remercier tous les doctorants du laboratoire et ceux avec qui j'ai passé le plus de temps, qui ont égayé ce doctorat. La salle du billard et l'Oxxo se souviendront de nous. Et bien sûr, comment ne pas remercier Aurélien tout particulièrement qui, pendant trois ans, sans relâche, m'a bien fait comprendre que son nom devait apparaître explicitement dans mes remerciements de thèse. Le voilà donc Aurélien, merci pour tout et surtout pour toute cette joie, partagée j'espère, pendant cette inoubliable partie de Mario Kart.

Ensuite, je remercie tous mes amis en dehors du laboratoire qui m'ont accompagné pendant tout mon cursus de physique. Ces 7 dernières années, cela a toujours été un grand plaisir de partager des discussions philosophiques, et des moments plus festifs, avec eux. J'espère que la tradition du Nouvel An perdurera.

Un immense merci va maintenant à ma famille paisible, aimante et soudée sans qui tout ceci aurait été plus difficile. Elle m'a toujours permis d'avancer sans me poser de questions, soutenu et accompagné dans tous mes projets. Merci pour tout.

Enfin j'aimerais terminer ces remerciements en adressant un très grand merci, forcément trop sobre, à Léa et son soutien inconditionnel depuis 7 ans.

Contents

List of Figures

List of Tables

List of code samples

List of Abbreviations

BSM	Beyond the Standard Model
DM	Dark Matter
EFT	Effective Field Theory
irrep	Irreducible representation
LEP	Large Electron-Positron collider (CERN)
LFU	Lepton Flavor Universality
LFUV	Lepton Flavor Universality Violation
LHC	Large Hadron Collider (CERN)
LO	Leading Order
MSSM	Minimal Supersymmetric Standard Model
NLO	Next-to-Leading Order
NMFV	Non-Minimal Flavor Violation
NMSSM	Next-to-Minimal Supersymmetric Standard Model
PDF	Parton Distribution Function
PDG	Particle Data Group
pMSSM	phenomenological Minimal Supersymmetric Standard Model
QCD	Quantum Chromo-Dynamics
QED	Quantum Electro-Dynamics
RGE	Renormalization Group Equations
SM	Standard Model
SMEFT	Standard Model Effective Field Theory
SUSY	Super-Symmetry
UFO	Universal Feynman rules Output
VEV	Vacuum Expectation Value
2HDM	2 Higgs Doublets Model

CHAPTER 1

Introduction

1.1 The Standard Model of particle physics

Introduction

The Standard Model (SM) of particle physics is the best model we have to describe elementary particle interactions at high energies. Its content is presented in figure 1.1. This model describes three of the four fundamental forces:

- **The electromagnetic force.** Carried by the photon γ, it is responsible for light and all electromagnetic waves in general (radio, micro-waves, X-rays, γ-rays, ...). More importantly, this force is the main source of repulsion in matter e.g. preventing the gravitational collapse of planets and stars.
- **The strong nuclear force.** Carried by the gluon g it glues quarks together in nucleons (protons and neutrons). At larger scales it contains multiple nucleons in atomic nuclei and, together with electromagnetism, generates all the chemical diversity we know from the periodic table of elements.
- **The weak nuclear force.** Less critical to describe matter, the weak nuclear force is mediated by the W- and Z- bosons and is responsible for example of neutron β-decays to proton + electron + anti-electron-neutrino: $n \rightarrow p^+ e^- \bar{\nu}_e$. While for practical purposes this force is too weak to have a large impact on the world we see, it is very important theoretically in our understanding of the smallest scales of physics as we will discuss in the following.

A bit of History

The SM has been built progressively during the twentieth century. The first element in figure 1.1 to be known was the particle of light, the photon.[1] Around 1900 the electron was observed as a negatively-charged particle in matter for example in the famous

1. We cannot give a precise time for the photon discovery as our description of light has gone through a lot of debates and controversies during several centuries.

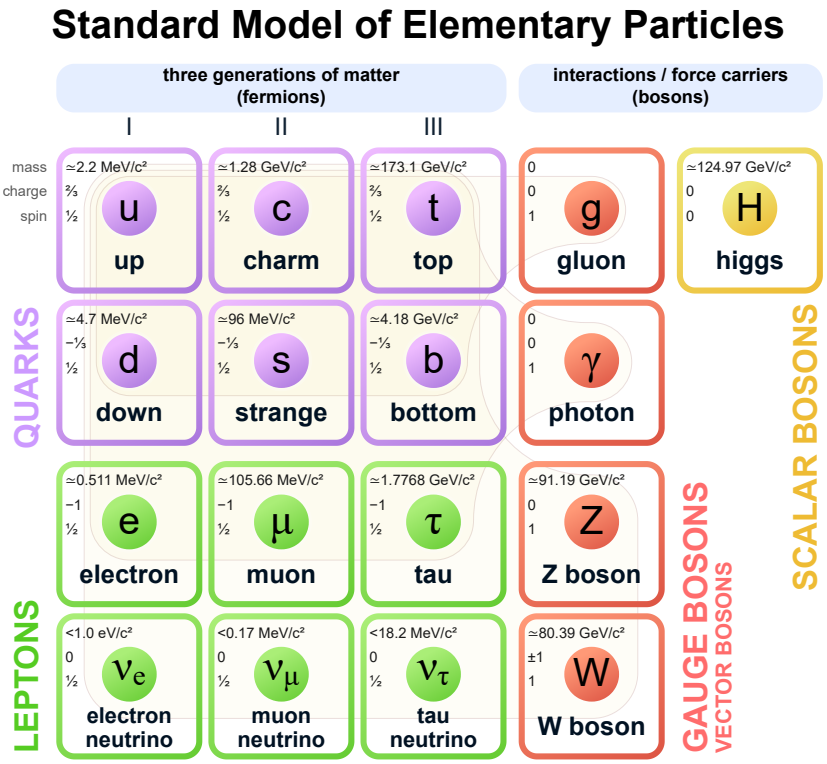


Figure 1.1 – The Standard Model content [1]. The three fermion generations are similar, only the masses are different and particles of higher generation are more massive (and therefore less abundant in the Universe).

cathode experiment [2]. The electron-neutrino was observed in 1956 [3], previously postulated to explain the apparent violation of energy conservation in neutron β-decays. In 1959 a theory describing electromagnetism and the weak nuclear force was introduced [4], postulating in particular the force mediator W that was later discovered experimentally. Although the Z-boson was not predicted initially it quickly became necessary, in particular for the Higgs mechanism [5,6] (1964) that has the ability to explain why the W-boson has a non-zero mass contrary to the photon if a new weak neutral mediator is postulated. [2] This particle, the Z-boson, was later discovered at CERN in 1983 [7]. The paper by Weinberg [8] (1967) successfully unified all the concepts above and strongly resembles to the actual Standard Model with less particle content.

As quarks cannot be observed as free particles at reasonable energies, they were only discovered later thanks to deep-inelastic scattering experiments in 1969 [9,10] that probed the internal structure of protons. Later on, in 1979, we discovered gluons in three-jets events [11] caused by the production of a quark/anti-quark pair and a gluon. Quarks were postulated theoretically in the so-called quark model that inferred the internal structure of hadrons (particles composed of quarks such as neutrons and protons) by classifying their different species observed in experiment. We later realized that quark flavors

2. Without a new particle the Higgs mechanism would predict a mass for the photon. The Z-boson is thus required to absorb this mass from the Higgs mechanism and keep the photon massless.

are not conserved by the weak interaction. This gave rise to the so-called CKM matrix [12] parametrizing the relative strengths of flavor changing charged currents through W-boson interactions.

The matter content of the second and third fermion generations has been discovered progressively such as the charm quark in 1974 [13, 14], the tau particle in 1975 [15] or the top quark at Fermilab [16] and D0 [17] (1995). The Higgs boson was the last missing piece of the Standard Model and was discovered in 2012 at the LHC [18, 19].

The fourth missing fundamental interaction

The SM does not describe gravity. This deficiency is the main motivation for **quantum gravity**, a theory that would describe gravitation at the quantum level i.e. mediated by another elementary particle, the **graviton**. For now, particle interactions can be predicted by the SM which is based on quantum field theory and the gravitational force is understood thanks to the general theory of relativity which is a non-quantized description of the space-time curvature. Several theories can have a quantized gravity such as string theory [20–23] or loop quantum gravity [24, 25] but to this day still struggle to provide predictions for elementary particle physics that we could measure at the energy scales available to us. Quantum gravity is beyond our scope as we will discuss observable phenomena in high energy physics.

A more theoretical description of the SM

The Standard Model particle content presented in figure 1.1 does not describe perfectly the fundamental structure of particles. In the following we present the SM content in a more mathematical way, introducing **gauge groups and representations**.

A gauge group corresponds to a fundamental interaction and comes with a gauge boson that mediates the associated force. Then, each particle can feel the force in different ways (or equivalently interact with the gauge boson in different ways) depending on its representation in this group. For more details about representation theory see chapter 5. Before the electroweak symmetry breaking, the SM gauge group is

$$SU(3)_c \times SU(2)_L \times U(1)_Y, \tag{1.1}$$

with $SU(3)_c$ the color group of the strong nuclear force and $SU(2)_L \times U(1)_Y$ that contains both the weak nuclear force and electromagnetism. For example, quarks are triplets of $SU(3)_c$ meaning that they have three possible color states conventionally named R, G and B that are connected by interactions with gluons carrying the force. Such representations are noted from their dimensions e.g. 3 in the triplet example. The $U(1)_Y$ group is similar to the $U(1)_{\text{em}}$ group of electromagnetism, i.e. to each particle is associated a fractional charge and the representation is denoted by the charge value.

The Standard Model content in terms of spin and gauge representations before symmetry breaking is presented in table 1.1. At high energies the SM Lagrangian has more symmetries and there are less independent fields. Q_L for example contains $3 \times 3 \times 2 = 18$ fermions because the three fermion generations, the three $SU(3)_c$ colors and the two parts of the $SU(2)_L$ doublet (later identified as up- and down-type quarks) are degenerate.

Particle	Spin	$SU(3)_c \times SU(2)_L \times U(1)_Y$ representation
Q_L	$1/2$	$(3, 2, 1/6)$
U_R	$1/2$	$(3, 1, 2/3)$
D_R	$1/2$	$(3, 1, -1/3)$
L_L	$1/2$	$(1, 2, -1/2)$
E_R	$1/2$	$(1, 1, -1)$
H	0	$(1, 2, 1/2)$
g	1	$(8, 1, 0)$
W	1	$(1, 3, 0)$
B	1	$(1, 1, 0)$

Table 1.1 – Content of the Standard Model before the electroweak symmetry breaking i.e. at very high energies. The three fermion generations are implicit with the upper case letters, for example $U \equiv (u, c, t)$. Q_L contains all up- and down-type left-handed quarks. Similarly, L_L contains all left-handed neutrinos and leptons (e,μ,τ).

When considering the same theory at low energies, the Higgs doublet H gets a non-negligible vacuum expectation value (VEV) and the $SU(2)_L \times U(1)_Y$ symmetry is spontaneously broken to the electromagnetism $U(1)_{em}$. This means that three of the four bosons gauging $SU(2)_L \times U(1)_Y$ acquire a non-zero mass and one combination of them remains massless: The photon. The Higgs VEV also gives mass to fermions. Fermions in $SU(2)_L$ doublets are not degenerate anymore and must therefore be considered separately e.g.

$$\begin{aligned} L_L &\to \begin{cases} N_L \\ E_L \end{cases}, \\ Q_L &\to \begin{cases} U_L \\ D_L \end{cases}, \end{aligned} \tag{1.2}$$

for all three generations (implicitly considered through upper-case letters) with $N_L = (\nu_e, \nu_\mu, \nu_\tau)$. The flavor symmetry between the different generations is also broken and the final Standard Model content in the unbroken gauge

$$SU(3)_c \times U(1)_{em} \tag{1.3}$$

is presented in table 1.2.

One can see that the model now contains only one real scalar boson, the Higgs boson h, whereas in table 1.1 there were four scalar degrees of freedom in the complex doublet H. The three missing degrees of freedom are absorbed when $W^\pm$ and Z acquire non-trivial masses and correspond to the spin 0 projection of the weak bosons that are not physical for massless vectors. [3]

3. A massless vector can have 2 different spin projections: ± 1. When it acquires a non-zero mass the spin projection 0 becomes physical and corresponds to a third degree of freedom.

Particle	Spin	$SU(3)_c \times U(1)_{em}$ representation
u_L	$1/2$	$(3, 2/3)$
u_R	$1/2$	$(3, 2/3)$
d_L	$1/2$	$(3, -1/3)$
d_R	$1/2$	$(3, -1/3)$
ν_{e_L}	$1/2$	$(1, 0)$
e_L	$1/2$	$(1, -1)$
e_R	$1/2$	$(1, -1)$
h	0	$(1, 0)$
g	1	$(8, 0)$
$W^{\pm}$	1	$(1, 1)$
Z	1	$(1, 0)$
γ	1	$(1, 0)$

Table 1.2 – Content of the Standard Model after the electroweak symmetry breaking i.e. at low energies. Only the first fermion generation is shown as the other ones are identical. The neutrino is the only fermion to have no right-handed counter-part in the SM.

The theoretical framework

The SM is a quantum field theory built to describe elementary particle interactions. A particle is described as a **relativistic quantum field** defined at each space-time point X such as

$$\hat{\Phi}(X) = \int_p \left(\hat{a}_p e^{-ipX} + \hat{a}_p^\dagger e^{+ipX} \right), \tag{1.4}$$

that is an excitation of the field $\hat{\Phi}$ at X, an infinite sum of oscillation modes with momentum p. Creation and annihilation quantum operators $\hat{a}$ and $\hat{a}^\dagger$ are similar to the ones defined in the quantum harmonic oscillator [26], creating or annihilating one particle of momentum p. The definition above corresponds to a real spin 0 particle for simplicity. All fields are Lorentz covariant, i.e. transform correctly under the Lorentz transformations of special relativity. These definitions allow us to describe relativistic particles which have very high energies while keeping track of their quantum nature. Then, the quantum vacuum with no particle is defined as the quantum state $|0\rangle$ and creation operators $\hat{a}_p^\dagger$ for example act on this state to produce another one, containing a particle. Namely

$$\hat{a}_p^\dagger |0\rangle \equiv |\Phi_p\rangle, \tag{1.5}$$

with $|\Phi_p\rangle$ the quantum state containing one particle Φ with momentum p. All the states defined in this way lie in a Fock space, also generalized from the simple quantum harmonic oscillator.

From a theory describing elementary particles we want to predict physical phenomena i.e. transitions between different quantum states. For example, a β-decay corresponds to a transition from an initial state $|n\rangle$ to a final state $|pe\bar{\nu}\rangle$ where we ignore momenta. From quantum mechanics, the transition amplitude noted $i\mathcal{M}$ is the product between the initial and final states:

$$i\mathcal{M}(n \to pe\bar{\nu}) \sim \langle pe\bar{\nu}|n\rangle, \tag{1.6}$$

and the transition probability P is then proportional to the squared amplitude, namely

$$P(n \to pe\bar{\nu}) \propto |i\mathcal{M}|^2, \tag{1.7}$$

which represents the theory prediction. Now that the framework is well-known the challenge for theoretical physicists is to calculate the so-called S-matrix elements such as $\langle pe\bar{\nu}|n\rangle$ in a given theory (the SM or beyond), a task known to be very hard and time-consuming. This calculation is based on the **Lagrangian formalism**. The Lagrangian $\mathcal{L}$ is a mathematical expression containing all the theory interactions and amplitudes such as $\langle pe\bar{\nu}|n\rangle$ are derived from it as presented in chapter 6.

In the SM the descriptions of electromagnetism and the strong nuclear force with quantum field theory are provided with **Quantum Electro-Dynamics** [27] (QED), and **Quantum Chromo-Dynamics** [28] (QCD) respectively. We will not present further fundamental aspects of quantum field theory as it requires dedicated books such as the Schwartz [29] or the more involved Weinberg trilogy [30–32].

1.2 Motivations to go beyond the Standard Model

In the previous section the Standard Model of particle physics whose last piece, the Higgs boson, was observed at the LHC in 2012 was briefly presented. By introducing some historical facts about its construction we discussed how theoretical predictions were important to guide experimental measurements at a time when the picture in figure 1.1 was incomplete. These predictions are always driven by inconsistencies in the theory or experiments. The neutrino was required to respect energy conservation in β-decays, the W-boson was then needed to build a consistent theory of these decays, the Higgs mechanism allowed physicists to explain particle masses, etc.

Now one may wonder if the Standard Model is complete or if we still miss some un-discovered particles and/or interactions. From our understanding of the underlying theory, we know for sure that the Standard Model does not describe all the phenomena we observe and that we must search for new physics. The question is therefore to know where the SM must be corrected or complemented. While most of the elementary interaction measurements are in very good agreement with the SM, there are still questions to which particle physicists must answer. Let us now present some inconsistencies we observe nowadays thanks to particle physics experiments, that could lead us to new discoveries in the twenty-first century.

1.2.1 Theoretical shortcomings

Neutrino masses

In the SM, fermion mass terms can be generated with a left- and a right-handed parts, namely

$$-m\left(\bar{\psi}_L\psi_R + \bar{\psi}_R\psi_L\right), \tag{1.8}$$

for a Dirac fermion ψ of mass m. The left- and right-handed parts of ψ are noted ψ_L and ψ_R respectively. In the SM, neutrinos have no right-handed parts and cannot get such a

mass term whereas we measured experimentally from neutrino oscillations [33, 34] that they have non-zero masses. This issue has however a very special feature because from the SM structure we know that a right-handed neutrino would not be measurable [4] as it would not interact with any other particle (i.e. it would be transparent to electromagnetism, weak and strong nuclear forces). It is moreover the most simple solution to this problem because it requires no additional hypothesis to make neutrinos different from the other fermions.

Another possibility is to consider a Majorana neutrino without right-handed counterpart that couples to itself such as

$$-\frac{m}{2}\left(\lambda_L^T i\sigma_2 \lambda_L + \lambda_L^\dagger i\sigma_2 \lambda_L^*\right), \tag{1.9}$$

where λ_L is a Majorana fermion of mass m and σ_2 the second Pauli matrix. Such a mass definition implies that the fermion is its own anti-particle. In the neutrino case, this would imply the possibility of neutrino-less double β-decay (often noted $0\nu\beta\beta$) as presented in figure 1.2. A β-decay is already a rare process, the neutrino-less double decay is therefore even less likely to happen (considering that neutrinos are Majorana particles) and has not yet been observed experimentally [35].

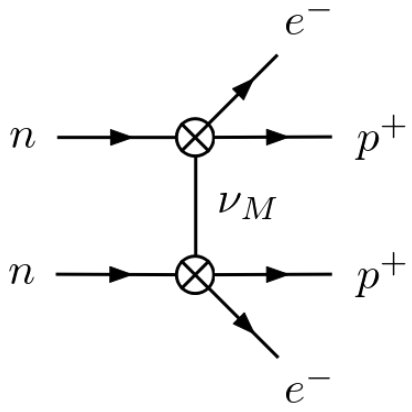


Figure 1.2 – Neutrino-less double β-decay which is possible if the neutrino is a Majorana fermion ν_M. Each vertex $\otimes$ represents a β-decay. The same neutrino can participate to both because it is its own anti-particle. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

Dark matter

From several cosmological measurements such as galaxy rotation curves or galaxy collisions, we know that there is in the Universe a large amount of matter that we cannot see i.e. that does not interact with light. The measurements of the Cosmic Microwave Background (CMB) by WMAP [36–40] and Planck [41, 42] also represent strong evidence for the existence of *dark matter*. In particular, the power spectrum of the CMB cannot

4. Strictly speaking this is not true as the right-handed neutrino still couples to its left-handed counterpart through the mass term and could be observed in couplings proportional to its mass with the Higgs boson for example. However, as neutrino masses are extremely small (< 1.1 eV [35] i.e. 500′000 times lighter that the electron) such processes are for the moment not measurable.

be explained without dark matter. It has been derived that dark matter is several times more abundant than the baryonic matter that we can observe in our telescopes. This matter can be of several kinds such as elementary particles, dark objects or primordial black holes. It is important to note here that dark matter may not be made of particles but as particle physicists we can investigate which extensions of the SM could provide a good dark matter candidate. In the following we therefore consider only the elementary particle alternative.

Similarly to the neutrino mass, this issue has the peculiar feature to be explained by very simple and non-observable scenarios. We know that if dark matter is composed of elementary particles they interact only weakly to ordinary matter, but it could also in principle not interact at all. Table 1.1 presented the high energy content of the SM, before the electroweak symmetry breaking. There is no theoretical constraint on the types of representations a model can contain. In particular, nothing prevents us to add pure gauge singlets i.e. with a representation $(1, 1, 0)$. Such fields would not interact with anything except through the gravitational force and would therefore behave like dark matter without being ever measurable in particle physics experiments.

As the study of non-observable theories is speculative and unscientific, search for dark matter in particle physics consists in the theoretical and experimental study of BSM models that provide a candidate for dark matter which still interacts with ordinary matter and is therefore observable.

1.2.2 Flavor anomalies

Experimental measurements of $b \to s$ transitions, presented in figure 1.3, present tensions with SM predictions in particular for $b \to s\mu^+\mu^-$ processes (see e.g. [43–48]). These tensions are called the **flavor anomalies**. If confirmed, they could be the sign of

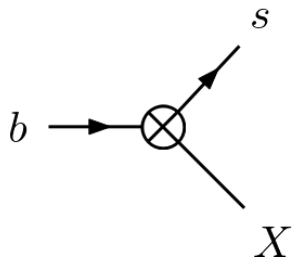


Figure 1.3 – Generic loop-level process for rare $b \to sX$ decays with X any electrically neutral final state. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

Lepton-Flavor Universality Violation (LFUV) stating that not all leptons behave in the same way (independently of their different masses).

Several BSM models are candidate to explain flavor anomalies:

- Z' **models** (see e.g. [49–51]) introduce a boson similar to Z but with non-vanishing tree-level flavor-changing couplings with quarks.
- **Composite Higgs models** (see e.g. [52–54]) can cause LFUV at the tree level. They postulate that the Higgs boson is not an elementary particle but composed of fermions.

- **Leptoquark models** (see e.g. [55–59]) define a direct lepton-quark-leptoquark coupling that would participate to flavor-changing currents. The new particles, the leptoquarks, can therefore be either scalar or vector fields.

1.2.3 The anomalous magnetic dipole moment of μ

The muon magnetic dipole moment can be physically interpreted as the coupling between the muon spin and the electromagnetic field. As the muon is electrically charged this interaction is non-zero and the muon produces a magnetic field. The Dirac equation predicts a muon dipole moment [35]

$$\vec{M} = g_\mu \frac{e}{2m_\mu} \vec{S}, \tag{1.10}$$

with the gyro-magnetic ratio $g_\mu = 2$, m_μ the muon mass, $\vec{S}$ its spin and finally e the universal electromagnetic coupling constant. However, this magnetic moment must be corrected by quantum loop effects such as the leading contribution presented in figure 1.4.

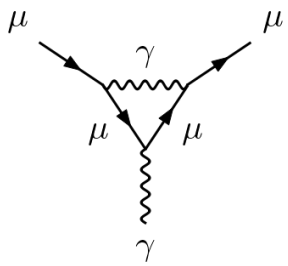


Figure 1.4 – QED contribution to the muon anomalous magnetic dipole moment in the SM. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

We parametrize the deviation from the tree-level value with the so-called **anomalous magnetic dipole moment**

$$a_\mu \equiv \frac{g_\mu - 2}{2}, \tag{1.11}$$

often noted $(g-2)_\mu$. The combined experimental average is [60]

$$a_\mu^{EXP} = 116\,592\,061(41) \times 10^{-11}, \tag{1.12}$$

while the value predicted by the SM is [61]

$$a_\mu^{SM} = 116\,591\,810(43) \times 10^{-11}. \tag{1.13}$$

The tension between experiments and the theoretical prediction is therefore

$$a_\mu^{EXP} - a_\mu^{SM} = (251 \pm 59) \times 10^{-11}, \tag{1.14}$$

that corresponds to a 4.2σ discrepancy. There are still some interrogations about the theoretical prediction from the Standard Model (see for example lattice QCD calculations [62]) but such a large tension could be a sign of new physics in the muon sector motivating the search for BSM scenarios shifting the value of a_μ.

1.2.4 The Hierarchy problem

The Higgs mass is measured to be [35]

$$m_h = 125.25 \pm 0.17 \text{ GeV}, \tag{1.15}$$

and is in agreement with the leading SM predictions from the electroweak spontaneous symmetry breaking. However, the Higgs mass gets quantum corrections from loop effects as presented in figure 1.5.

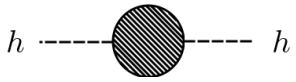


Figure 1.5 – Generic diagram representing loop corrections to the Higgs mass. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

These loop corrections depend quadratically on the energy cut-off used in the calculation i.e. the energy scale at which our description is not correct anymore. This cut-off is at most equal to the Planck mass

$$m_P = \sqrt{\frac{\hbar c}{G}}, \tag{1.16}$$

at which we need a theory of quantum gravity. In natural units $\hbar = c = 1$ and the Planck mass reads

$$m_P = G^{-1/2} \approx 1.22 \times 10^{19} \text{ GeV}. \tag{1.17}$$

This energy scale is much larger than m_h and a small correction to the Higgs mass cannot be explained. This is known as the **hierarchy problem** related to the enormous energy gap between the weak nuclear force at $\sim$ 100 GeV and the gravitation at $\sim$ 10^{19} GeV. Some physicists argue that there must be new physics between those scales to solve this issue, motivating the search for BSM models in this energy range. From the experimental point of view of particle colliders, we are for now able to reach energies of the order of 1 TeV = 1000 GeV in the center of mass frame.

Some BSM scenarios can theoretically solve the Higgs mass corrections issue:

- **Supersymmetric (SUSY)** models [63–66], such as the Minimal Supersymmetric Standard Model (MSSM), predict cancellations in the involved loop corrections between SM particles and their SUSY partners.
- **Composite Higgs models** [67]. As in these models the Higgs boson is no longer a fundamental scalar particle, its mass corrections are therefore not calculated in the same way and composite Higgs models are candidates to solve the Higgs mass correction issue.

1.3 Model building in particle physics

A particle physics model is built from several different elements. Such a model must be able to describe the different elementary particles that exist and their interactions. To build a model, the symmetries and the **fundamental interactions** must be defined first. Then, the **matter content** is constructed with all the different species and their couplings to the fundamental forces. Finally, **additional couplings** respecting the model symmetries can be postulated such as the Yukawa couplings in the Standard Model that couple the Higgs field to fermions and explain why fermions have non-zero masses through the Higgs mechanism. As a last step, one should check the model's consistency by ensuring that it is **anomaly-free** as we will discuss at the end of this section.

1.3.1 Symmetries

A model is symmetrical with respect to a transformation $\mathcal{T}$ if the Lagrangian $\mathcal{L}$ – containing all the kinetic, mass and interaction terms – is physically invariant under this transformation:

$$\mathcal{T}(\mathcal{L}) = \mathcal{L} + \partial_\mu V^\mu, \tag{1.18}$$

with $\partial_\mu V^\mu$ a possible divergence term that cancels out of any physical calculation.

Let us consider the free electron, a spin $1/2$ particle ψ of mass m. The corresponding Lagrangian reads

$$\mathcal{L} = \bar{\psi} i\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi. \tag{1.19}$$

This Lagrangian describes the free propagation of the electron and is symmetrical under several transformations such as

$$\begin{aligned} \psi &\mapsto \mathcal{T}_\alpha(\psi) = e^{i\alpha}\psi \\ \bar{\psi} &\mapsto \mathcal{T}_\alpha(\bar{\psi}) = e^{-i\alpha}\bar{\psi}, \end{aligned} \tag{1.20}$$

with α a real number. One can check that $\mathcal{T}_\alpha$ indeed leaves the Lagrangian invariant:

$$\mathcal{T}_\alpha(\mathcal{L}(\psi, \bar{\psi})) \equiv \mathcal{L}(\mathcal{T}_\alpha(\psi), \mathcal{T}_\alpha(\bar{\psi})) = \mathcal{L}(\psi, \bar{\psi}). \tag{1.21}$$

This is a **global symmetry**, meaning that the Lagrangian is invariant under a transformation with constant parameters α. If we promote α to be a function over the Minkowski space-time $\alpha(X)$, the Lagrangian is not invariant anymore because $\partial_\mu \alpha \neq 0$ and

$$\delta_\alpha \mathcal{L} \equiv \mathcal{T}_\alpha(\mathcal{L}) - \mathcal{L} = -\partial_\mu \alpha \bar{\psi}\gamma^\mu \psi, \tag{1.22}$$

where we used the Taylor development considering α as an infinitesimal parameter

$$e^{i\alpha} = 1 + i\alpha + \mathcal{O}(\alpha^2). \tag{1.23}$$

One can see that the derivative prevents the symmetry to be promoted to a **local symmetry**, i.e. valid for any $\alpha(X)$. This local symmetry can be achieved by adding a new field A_μ also transformation under $\mathcal{T}_\alpha$ following

$$A_\mu \mapsto \mathcal{T}_\alpha(A)_\mu = A_\mu + \partial_\mu \alpha. \tag{1.24}$$

With this new field, one can create an invariant term by introducing the covariant derivative D_μ

$$\bar{\psi} i\gamma^\mu D_\mu \psi, \tag{1.25}$$

with

$$D_\mu = \partial_\mu - iA_\mu. \tag{1.26}$$

Using the transformations defined above, this term now transforms as

$$\begin{aligned} \mathcal{T}_\alpha(\bar{\psi} i\gamma^\mu D_\mu \psi) &= \mathcal{T}_\alpha(\bar{\psi} i\gamma^\mu \partial_\mu \psi) + \bar{\psi}\gamma^\mu \mathcal{T}_\alpha(A_\mu)\psi \\ &= (-\partial_\mu \alpha + \partial_\mu \alpha)\, \bar{\psi}\gamma^\mu \psi + \bar{\psi} i\gamma^\mu D_\mu \psi \\ &= \bar{\psi} i\gamma^\mu D_\mu \psi. \end{aligned} \tag{1.27}$$

This connection between the electron ψ and the **photon** A_μ is therefore invariant under the **gauge transformation**. In general, any matter field transforming under an unbroken gauge symmetry is connected to the gauge boson (A_μ in this case) through an interaction of the type $\bar{\psi} A \psi$ like the one presented in figure 1.6.

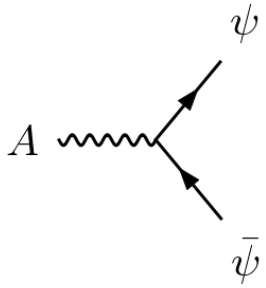


Figure 1.6 – General form of gauge interaction preserving the associated gauge symmetry. Two matter particles ψ interact with one gauge boson A.

The local symmetries of a model define therefore the gauge symmetries, i.e. the fundamental forces. The principle above can be generalized to arbitrary couplings and dimensions. The strong nuclear force mediated by the 8 gluons g_μ^A gauging the $SU(3)_c$ symmetry group has gauge couplings of the form

$$\bar{\psi}_i \gamma^\mu (D_\mu \psi)_i = \bar{\psi}_i i\gamma^\mu \partial_\mu \psi_i + i g_s \bar{\psi}_i \gamma^\mu g_\mu^A T_{ij}^A \psi_j, \tag{1.28}$$

with i the quark color indices, g_s the strong coupling constant, T_{ij}^A the tensor connection between the gluon g_μ^A and the matter field ψ_i ensuring that the $SU(3)_c$ gauge transformation leaves the Lagrangian invariant.

Global symmetries can also be defined in a particle physics model. They are not local, i.e. are not gauged and are not associated with a gauge boson or a fundamental force. They are the simple consequence of the empirical observation that some interactions, while allowed theoretically, do not exist. In the Standard Model for example, the baryon number B and the lepton number L are conserved. We associate with each of these conservation laws a $U(1)$ symmetry group (as the example above for the transformation $\mathcal{T}_\alpha$), $U(1)_B$ and $U(1)_L$. These symmetries are not local and the Lagrangian is only invariant

under a constant transformation α_L and α_B. (Anti-)quarks have a baryon number $(-)1/3$ and (anti-)leptons have a lepton number $(-)1$. Terms such as

$$\begin{aligned} &\phi^A \bar{l} \Gamma^A l, \\ &\phi^A \bar{q} \Gamma^A q, \end{aligned} \tag{1.29}$$

for a lepton l and a quark q conserve both symmetries but a coupling between one lepton and one quark is not possible as

$$\phi^A \bar{q} \Gamma^A l, \tag{1.30}$$

would not be invariant under the baryon or lepton number transformations. In the equations above, ϕ^A and γ^A are arbitrary boson and γ-matrix combination respectively.

To summarize what we discussed about symmetries, let us recall the three main features of symmetries:

- Any unbroken symmetry, local or global, is associated with a **conservation law**. In the Standard Model for example, the $U(1)_{em}$ symmetry is associated with the electric charge conservation and the global $U(1)_L$ symmetry is associated with the conservation of the lepton number.
- **Local / Gauge symmetries** are conserved locally, i.e. under a transformation with variable $\alpha(X)$ parameters. In order for the symmetry to be conserved, any dynamical field transforming non-trivially must couple to the **gauge boson** through a gauge interaction. These couplings are what we interpret as the fundamental forces of nature.
- **Global symmetries** are conserved only with constant parameters α. They do not require any additional particle or gauge interaction and are the mathematical occurrence of a conservation law in the Lagrangian.

1.3.2 Matter content

Once the gauge group has been defined, all postulated particles must be irreducible representations of this gauge group. [5] Table 1.1 presented the matter content of the SM as a set of particles, irreducible representations of the unbroken gauge group

$$SU(3)_c \times SU(2)_L \times U(1)_Y, \tag{1.31}$$

and table 1.2 presented the content in the final gauge group that we observe at low energies

$$SU(3)_c \times U(1)_{em}, \tag{1.32}$$

with the unbroken strong nuclear force $SU(3)_c$ with massless gluons, electromagnetic force $U(1)_{em}$ with a massless photon, and the broken weak nuclear force with massive vector bosons $W^{\pm}$ and Z^0.

The irreducible representation of a $U(1)$ group is a numerical charge such as the electric charge. For a non-abelian group such as $SU(2)$ or $SU(3)$, the representation is defined

5. See also chapter 5 for more details on irreducible representations.

by a dimension that is the number of independent degrees of freedom mixing under the transformation. For the spin group $SU(2)$ for example, spin $1/2$ fermions are the dimension 2 representation with spin up and down (2 possible states). This group is the same as $SU(2)_L$ from which is named the weak isospin that follows the same rules. In the case of $SU(2)_L$, the isospin up and down correspond for example to the electron neutrino ν_e and the left-handed electron e_L respectively that form a doublet (dimension 2 representation). A particle that does not feel the force associated to a group $\mathcal{G}$ is in the **trivial representation** of $\mathcal{G}$, of charge 0 for $U(1)$ and of dimension 1 for non-abelian groups.

There is no limitation in the matter content when one builds a BSM model, any particle in an irreducible representation of the unbroken gauge group can be added in the model.

What is a particle ?

One can see that there could be some ambiguity in what is called a particle. In table 1.1 for example, Q_L is defined as one particle while it actually contains all six left-handed quarks u_L, d_L, c_L, s_L, t_L and b_L. The Q_L notation is not just a shortcut to define multiple particles at once and reflects the way we would experimentally detect quarks at very high energies.

The lightest quark is u (up) with a mass around $2.5\ \mathrm{MeV}$ while the heaviest is t (top) with a mass about $173\ \mathrm{GeV}$. Particles of different masses are **distinguishable**. In the quark case, one can know whether an up-type quark is a u or a t depending on its mass. At very high energies however, well above the electroweak scale ($M_W \approx 100\ \mathrm{GeV}$) and the top quark mass scale ($m_t \approx 200\ \mathrm{GeV}$) the 6 quarks have negligible masses and they become **indistinguishable**. [6]

We therefore consider that **degenerate** quantum states with the same quantum numbers and interactions are simply different inner states of the same particle. This is the case for quark colors. All quarks have three independent color states conventionally called R, G and B but these states are not considered as different particles. Similarly, the gluon has 8 degenerate color degrees of freedom and is considered as a unique particle.

Therefore, there are in general much less particles than the number of independent **dynamical degrees of freedom** in a model. Counting one degree of freedom for each color or spin state, there are in the Standard Model

$$\underbrace{6\times3\times4}_{\text{quarks}}+\underbrace{3\times4}_{\text{charged leptons}}+\underbrace{3\times2}_{\text{neutrinos}}+\underbrace{8\times2}_{\text{gluon}}+\underbrace{2}_{\text{photon}}+\underbrace{3}_{Z}+\underbrace{3\times2}_{W^\pm}+\underbrace{1}_{Higgs}=118 \tag{1.33}$$

degrees of freedom, while we define 27 independent particles (counting anti-particles) for the interactions we observe at the typical energy scales of particle colliders and only 9 different particles in the limit of very high energies.

6. Different states still can be distinguished such as a spin measurement for an electron. The two spin states have the same mass, but one can measure the spin by imposing a magnetic field for example.

1.3.3 Spontaneous symmetry breaking

Inconsistent mass terms

In a theory without symmetry breaking, it is not possible to write down a gauge invariant mass term for gauge bosons or SM fermions, while we observe masses for the bosons $W^{\pm}$ and Z and all fermions.

For a gauge boson A_μ, a mass term such as

$$\mathcal{L} \ni \frac{1}{2} M_A^2 A_\mu A^\mu, \tag{1.34}$$

would not be gauge invariant. If we apply for example the transformation given in equation 1.24, we obtain

$$\delta\mathcal{L} = M_A^2 A^\mu \partial_\mu \alpha + \mathcal{O}(\alpha^2) \neq 0. \tag{1.35}$$

For a fermion ψ composed of Weyl fermions ψ_L and ψ_R, the mass term is

$$\mathcal{L} \ni -m(\bar{\psi}_L \psi_R + \bar{\psi}_R \psi_L). \tag{1.36}$$

For a fermion to have a mass term such as the ones we observe in nature, we must therefore couple the left- and right-handed parts together. However, as table 1.1 presented, left- and right-handed fermions have different gauge representations in the $SU(2)_L$ and $U(1)_Y$ groups. This has the simple consequence that the term above cannot be gauge invariant for any SM fermion.

We justified above the fact that mass terms that respect the gauge symmetry cannot be written in general for gauge bosons and SM fermions, that all transform under non-abelian gauge groups. As we observe massive particles, we need a gauge invariant procedure that can explain non-zero masses for SM particles. This procedure is the spontaneous symmetry breaking.

Principle

A symmetry is said spontaneously broken when the Lagrangian is invariant under this symmetry but that the **vacuum** is not. This mechanism is opposed to the explicit symmetry breaking, when the Lagrangian is not invariant. An example of such a symmetry can be seen in a quartic potential for a complex scalar boson ϕ:

$$V(\phi, \phi^\dagger) = -a\phi^\dagger\phi + \frac{b}{2}\left(\phi^\dagger\phi\right)^2, \tag{1.37}$$

where a and b are two positive constants. This is the so-called Mexican hat potential as presented in figure 1.7. The minimum of the potential is not located at $\phi = 0$ and consequently no quantum field theory calculation can be performed with ϕ as it is not a perturbation around the **ground state** i.e. the state that minimizes the potential. The minimum of $V(\phi, \phi^\dagger)$ can be found by finding the classical solution of

$$\begin{aligned} &\frac{\partial V(\phi, \phi^\dagger)}{\partial \phi^\dagger} = 0 \\ \Leftrightarrow &\left(-a + b\phi^\dagger\phi\right)\phi = 0, \text{ with } \phi \neq 0 \\ \Leftrightarrow &\phi^\dagger\phi = \frac{a}{b}. \end{aligned} \tag{1.38}$$

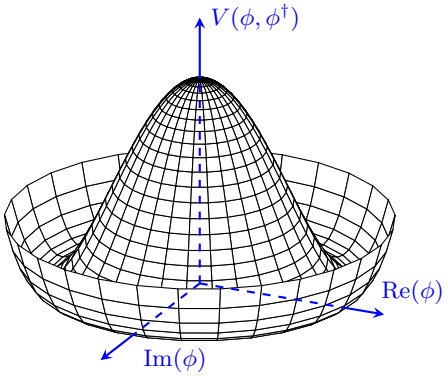


Figure 1.7 – Mexican quartic potential for a complex scalar field ϕ, expressed as a 2D function of the real and imaginary parts of ϕ.

The minimum is therefore any state for which the squared modulus of ϕ is equal to a/b, namely

$$\phi = \sqrt{\frac{a}{b}} e^{i\alpha}, \tag{1.39}$$

with α any angle in $[0, 2\pi[$. From the point of view of quantum mechanics, this minimum is interpreted as the **vacuum expectation value** (VEV) of the field ϕ, i.e. its average value, around which excited states can be measured. This is noted

$$\langle\phi\rangle = \sqrt{\frac{a}{b}} e^{i\alpha}. \tag{1.40}$$

The phase α is arbitrary and can be chosen, without loss of generality, to zero. Then, the field is expanded around its vacuum expectation value to define a new field $\tilde{\phi}$ that has a vanishing VEV:

$$\phi \equiv \sqrt{\frac{a}{b}} + \tilde{\phi}. \tag{1.41}$$

We can check that the new field has no VEV:

$$\begin{aligned} \langle\phi\rangle &= \left\langle \sqrt{\frac{a}{b}} + \tilde{\phi} \right\rangle \\ \Leftrightarrow \sqrt{\frac{a}{b}} &= \sqrt{\frac{a}{b}} + \langle\tilde{\phi}\rangle \\ \Leftrightarrow 0 &= \langle\tilde{\phi}\rangle. \end{aligned} \tag{1.42}$$

The potential in equation 1.37 can now be expressed as a function of $\tilde{\phi}$ and reads

$$V(\tilde{\phi}, \tilde{\phi}^\dagger) = \underbrace{\frac{a}{2}\left(\tilde{\phi}^\dagger + \tilde{\phi}\right)^2}_{=2a\,\mathrm{Re}(\tilde{\phi})^2} + \sqrt{ab}\tilde{\phi}^\dagger\tilde{\phi}\left(\tilde{\phi}^\dagger + \tilde{\phi}\right) + \frac{b}{2}\left(\tilde{\phi}^\dagger\tilde{\phi}\right)^2, \tag{1.43}$$

where constant terms have been dropped as they do not contribute to physical observables. We see 3- and 4-vertex interactions for $\tilde{\phi}$ but more importantly that the real part

of the new scalar field acquires a mass, contrary to the imaginary part, through the term $2a\,\mathrm{Re}(\tilde{\phi})$. This means that the two degrees of freedom are now different particles with distinct free dynamics. $\mathrm{Re}(\tilde{\phi})$ is the massive degree of freedom consequence of the symmetry breaking, and $\mathrm{Im}(\tilde{\phi})$ is the associated massless **Goldstone boson**. By defining

$$\tilde{\phi} \equiv \frac{h + iG}{\sqrt{2}}, \tag{1.44}$$

with h and G the real massive and Goldstone bosons respectively, the final potential is

$$V(h, G) = \frac{1}{2}m_h^2 h^2 + \sqrt{\frac{ab}{2}}\left(h^3 + hG^2\right) + \frac{b}{8}\left(h^4 + 2h^2G^2 + G^4\right), \tag{1.45}$$

with

$$m_h = \sqrt{2a}, \tag{1.46}$$

the mass of the real scalar boson h.

We described the potential behavior for the initial complex scalar ϕ and showed that the latter field has a non zero VEV. Once expanded around the potential minimum, the two degrees of freedom of ϕ are dynamically separated because one of them acquire a non zero mass, h, while the other remains massless, G. Combined with a gauge interaction, this mechanism can provide an explanation for the observation of massive vector bosons.

Mass for vector bosons

If the scalar field ϕ defined in the previous section is coupled to a gauge boson by gauge interaction, this boson also acquires a non zero mass. In equation 1.26 we defined the covariant derivative for a field charged under a $U(1)$ symmetry with a gauge boson A. Applying this to ϕ, its kinetic term reads

$$\begin{aligned} \mathcal{L}_{kin} \ni (D_\mu\phi)^\dagger D^\mu\phi &= (\partial_\mu\phi^\dagger + iA_\mu\phi^\dagger)(\partial^\mu\phi - iA^\mu\phi) \\ &= \partial_\mu\phi^\dagger\partial^\mu\phi + iA_\mu(\phi^\dagger\partial^\mu\phi - \phi\partial^\mu\phi^\dagger) + A^\mu A_\mu\phi^\dagger\phi, \end{aligned} \tag{1.47}$$

where we recover the standard interactions of scalar QED. The last term in the above equation is important because after symmetry breaking the field ϕ is expressed as

$$\phi = \sqrt{\frac{a}{b}} + \frac{h + iG}{\sqrt{2}}, \tag{1.48}$$

and the vector boson therefore acquires a mass:

$$A_\mu A^\mu\phi^\dagger\phi \ni \frac{a}{b}A_\mu A^\mu \equiv \frac{1}{2}M_A^2 A_\mu A^\mu, \tag{1.49}$$

with

$$M_A = \sqrt{\frac{2a}{b}}. \tag{1.50}$$

This principle can be generalized to an arbitrary gauged group provided that the scalar field is a non-trivial representation of it. In the SM, the Higgs mechanism uses a scalar

field H that is a doublet of $SU(2)_L$ and charged under $U(1)_Y$. H contains 4 degrees of freedom and is coupled to the 3 gauge bosons $W^{1,2,3}$ of $SU(2)_L$ and the one of $U(1)_Y$ named B. After spontaneous symmetry breaking, one H degree of freedom, the **Higgs boson**, becomes massive. The symmetry breaking also provides masses for the weak mediators. The covariant derivative for the H doublet is

$$(D_\mu H)_i = \partial_\mu H_i - igW^a_\mu \tau^a_{ij} H_j - \frac{1}{2} ig' B_\mu H_i, \tag{1.51}$$

with g and g' coupling constants, τ^a the $SU(2)$ generators and the $1/2$ factor is the hypercharge of H. The Higgs doublet H has a non zero VEV and is expanded following

$$H = \begin{pmatrix} G^+ \\ \frac{v+h+iG^0}{\sqrt{2}} \end{pmatrix}, \tag{1.52}$$

with G^0 and $G^\pm$ the three massless Goldstone bosons, h the massive degree of freedom and v the VEV parameter. By expanding the kinetic Lagrangian, one finds the following mass Lagrangian for electroweak bosons:

$$\mathcal{L}_{mass} \ni M_W^2 W^+_\mu W^{-\mu} + \frac{1}{2} M_Z^2 Z_\mu Z^\mu, \tag{1.53}$$

with

$$W^\pm = \frac{W^1 \mp iW^2}{\sqrt{2}}, \tag{1.54}$$

$$Z = \cos\theta_W W^3 - \sin\theta_W B, \tag{1.55}$$

where we defined

$$\tan\theta_W = \frac{g'}{g}, \tag{1.56}$$

$$M_W = \frac{1}{2} gv, \tag{1.57}$$

$$M_Z = \frac{M_W}{\cos\theta_W}. \tag{1.58}$$

One combination of W^3 and B remains therefore massless and it is the photon

$$A = \sin\theta_W W^3 + \cos\theta_W B. \tag{1.59}$$

It can be shown that, as the Higgs transforms under the gauge transformation $SU(2)_L \times U(1)_Y$, the three Goldstone bosons $G^\pm$ and G^0 are not physical and can be absorbed by a gauge transformation. These three degrees of freedom were however physical before the symmetry breaking and must appear in the low energy theory. As $W^\pm$ and Z acquire a mass, their spin 0 projections become physical and these bosons have 3 independent degrees of freedom while a massless vector boson only has the two possible projections ± 1. We say that the Goldstone bosons $G^\pm$ and G^0 are absorbed in the longitudinal polarizations (spin 0 projection) of $W^\pm$ and Z respectively. Among the 4 initial degrees of freedom of H, one becomes a real massive scalar h and three are absorbed by the three non-zero vector boson masses for W^+, W^- and Z.

Fermion masses

By coupling the scalar field ϕ to fermions in a gauge invariant way we can also generate fermion masses. These couplings are called **Yukawa** interactions and takes the general form for a fermion $\psi = \psi_L + \psi_R$:

$$\mathcal{L} \ni -Y\phi\psi_L^\dagger\psi_R + \text{h.c.}\,. \tag{1.60}$$

After the symmetry breaking, ψ also acquires a mass term and its mass m_ψ reads

$$m_\psi = Y\sqrt{\frac{a}{b}}. \tag{1.61}$$

The mass term above can explain SM fermion masses. The scalar field H is a $SU(2)$ doublet and allows us to write down a mass term such as

$$-Y_u\bar{Q}_L^i U_R\epsilon_{ij}H^{i\dagger} + -Y_d\bar{Q}_L^i D_R H^i + \text{h.c.}\,, \tag{1.62}$$

with i, j $SU(2)$ indices, ϵ_{ij} the fully anti-symmetric tensor and Q_L, U_R and D_R quark fields defined in table 1.1 (other indices are ignored in the argument). One can check from table 1.1 the the terms above indeed are gauge invariant. Under the right conditions, the spontaneous symmetry breaking can explain distinct and non-zero masses for u and d because of the fact that H is a $SU(2)_L$ doublets. This allows us to couple Q_L to U_R and D_R in a gauge invariant way. This procedure ends up producing masses for SM fermions by setting appropriate values for the Yukawa parameters Y_ψ in the initial Lagrangian.

By postulating the Higgs mechanism in the 60's [6], theoretical physicists had therefore a unique, elegant and gauge invariant way to explain the masses of weak bosons, $W^\pm$ and Z, and all SM fermions. The only missing piece to confirm this hypothesis was the remaining massive degree of freedom, the Higgs boson h, that was discovered about fifty years later [18, 19].

1.3.4 Additional interactions and anomalies

Any Lorentz and gauge invariant interaction can be added consistently to a particle physics model. There is however one limitation when building a new model: it must be **anomaly-free**. In quantum field theory, any symmetry classically conserved is said anomalous if it is violated by quantum loop corrections. If a gauge symmetry is anomalous it implies that the theory is not unitary. In the latter case, one could observe that energy is not conserved in some processes for example, or that when we sum the probabilities of an event's possible outcomes the result is not equal to 1. A reasonable theory must therefore be unitary and anomaly-free, i.e. all gauge symmetries must be conserved at the quantum level. We will not detail here how anomalies are derived from the quantum one-loop calculations. We nevertheless give the simple recipe allowing us to check that a theory is anomaly-free in the following.

Anomalies can be checked through triangle one-loop diagrams such as the one presented in figure 1.8 and it is known (see for example [29]) that it is proportional to

$$\sum_{f_L}\text{Tr}\left(T^A_{f_L}T^B_{f_L}T^C_{f_L}\right) - \sum_{f_R}\text{Tr}\left(T^A_{f_R}T^B_{f_R}T^C_{f_R}\right), \tag{1.63}$$

with f_L and f_R all the left- and right-handed fermions of the theory respectively and $T_f^{A,B,C}$ the generalized gauge couplings of the fermion f to the gauge bosons A, B and C. These couplings are equal to the charge for $U(1)$ groups and algebra generators for non-abelian symmetries.

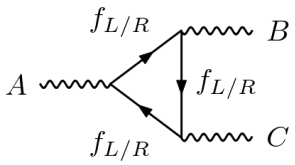


Figure 1.8 – Triangle loop-diagrams involved in the calculation of gauge anomalies in quantum field theory. The fermions in the loop are all the Weyl fermions of the theory. The anomaly must cancel for all A, B and C gauge bosons of the theory. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

In order to have an anomaly-free theory, equation 1.63 must vanish identically for all gauge bosons A, B and C. To illustrate the procedure, let us present two Standard Model examples of anomaly cancellations. Using the quantum numbers in table 1.1, we can calculate the anomalies for the diagrams $SU(2)_L^2 \times U(1)_Y$ and $U(1)_Y^3$. The anomaly for the first one reads

$$\begin{aligned} &\sum_{f_L} \mathrm{Tr}\Big(T^i_{f_L} T^j_{f_L} Y_{f_L}\Big) - \sum_{f_R} \mathrm{Tr}\Big(T^i_{f_R} T^j_{f_R} Y_{f_R}\Big), \\ &\quad = \mathrm{Tr}\left(\frac{\sigma^i}{2}\frac{\sigma^j}{2}\right) \underbrace{2}_{SU(2)\text{ doublets}} \times \Big(\underbrace{3 \times \frac{1}{6}}_{3\text{ colors}\times Y_{Q_L}} - \underbrace{\frac{1}{2}}_{Y_{L_L}} \Big) = 0, \end{aligned} \tag{1.64}$$

with $T_f^{i,j}$ the generators of $SU(2)_L$ and Y_f the hypercharge of the fermion f. As both L_L and Q_L are $SU(2)_L$ doublets, their generators are the Pauli matrices $\sigma^i/2$. For the $U(1)_Y^3$ diagram, there is no generator and the anomaly is simply a combination of hypercharge values:

$$\begin{aligned} &\sum_{f_L} Y^3_{f_L} - \sum_{f_R} Y^3_{f_R} = 2 \times Y^3_{L_L} - Y^3_{E_R} + 3 \times \Big(2 \times Y^3_{Q_L} - Y^3_{U_R} - Y^3_{D_R}\Big) \\ &\quad = 2 \times (-1/2)^3 - (-1)^3 + 3 \times (2 \times (1/6)^3 - (2/3)^3 - (-1/3)^3) = 0. \end{aligned} \tag{1.65}$$

It has been proven above that the SM is anomaly-free for two types of diagrams, $SU(2)_L^2 \times U(1)_Y$ and $U(1)_Y^3$. This must be done in general for all different types. When building a new BSM model it is crucial to check that it is anomaly-free before deriving any prediction as otherwise the model is simply inconsistent.

1.3.5 Discussion on Standard Model extensions

Beyond the Standard Model scenarios are particle physics models that **extend** the SM and lead to similar predictions at low energies. At higher energies however, they must describe new physics effects that are not predicted by the SM and possibly answer the questions that are not addressed by the SM. The typical mass scales of the Standard

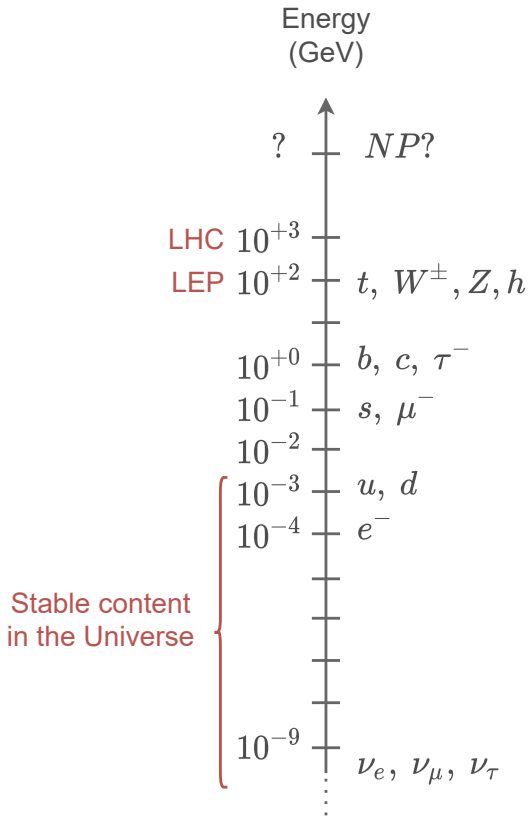


Figure 1.9 – Energy scales of Standard Model particles in GeV. BSM searches study New Physics (NP) effects at higher energy scales than the more massive particles in the SM.

Model particles are presented in figure 1.9. For now, the most powerful particle collider we have is the LHC, producing an effective energy [7] at the center of mass in the TeV range. A BSM scenario must reproduce all the physics we observe below the TeV scale and provide new physics effects at higher energies that answer questions raised by the SM.

The top-down approach

To build a BSM model, one can choose the top-down approach, i.e. defining a model at very high energies and deriving predictions at lower energies. This method is in general motivated by fundamental features of particle interaction and aims to provide a model that deepens our understanding of the laws of nature. Examples are Grand Unification Theories (GUT) that unify the three fundamental particle interactions (electromagnetism, weak and strong nuclear forces) at very high energies in one unique force from which all the SM can be derived. This approach has the advantage to generate elegant and enlightening models, also unifying matter particles. The drawback is that they are less

7. The center of mass energy at the LHC is 13 TeV but as we collide protons, the interacting partons (quarks and gluons) carry only a fraction of this energy.

connected to phenomenology as they predict NP effects at energy scales that we may not be able to reach with our current technology.

The bottom-up approach

The bottom-up approach is the inverse of the top-down method. It consists in describing empirically what we observe at low energies and try to understand what happens at higher energies but without necessarily providing a complete picture. Effective Field Theories (EFT) in particular follow this approach. As our knowledge is based on experiment iterations always at higher energies, we naturally follow a bottom-up approach. From a theoretical point of view, the bottom-up approach is minimal to describe NP effects.

In particular, it is possible to build BSM scenarios as simple SM extensions, adding only a few features to the model. One can for example postulate the existence of a new neutral massive vector boson X that couples to the Standard Model leptons and contribute, at the one-loop level, to $(g-2)_\mu$ that was introduced in section 1.2.3. This is presented in figure 1.10. Without providing a deeper explanation for the existence of X,

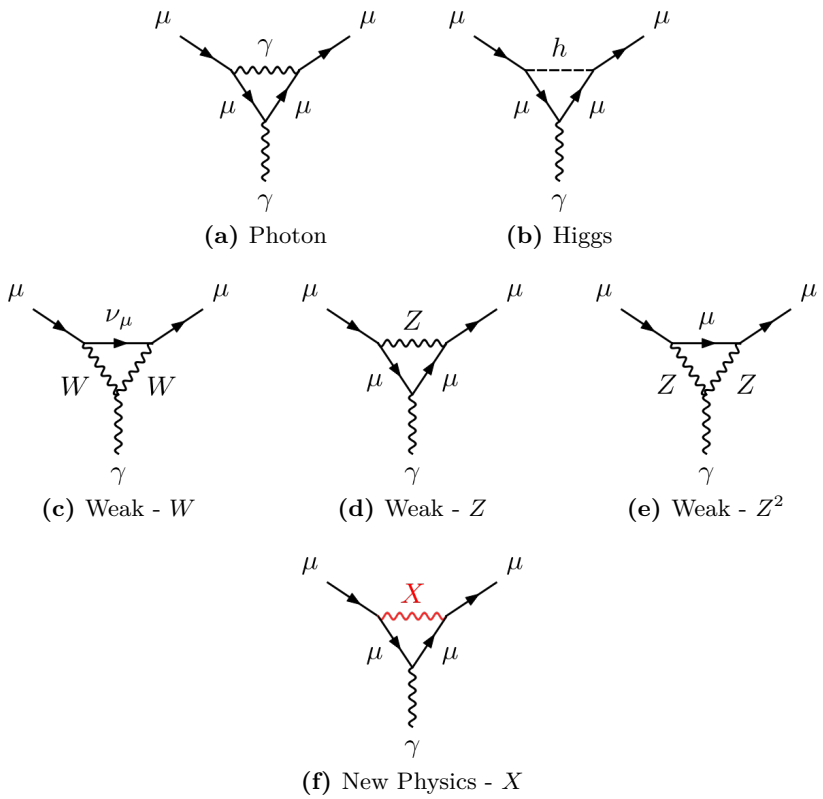


Figure 1.10 – One-loop diagrams to $(g-2)_\mu$ with the Standard Model contributions and one New physics diagram involving a massive neutral vector boson X. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

we can predict that such a particle would imply a correction to $(g-2)_\mu$ of the order

$$\delta(g-2)_\mu \propto \left(\frac{m_\mu}{M_X}\right)^2, \tag{1.66}$$

with M_X the mass of the new vector boson X. Then, one has to know what would be the other phenomenological implications of such vector boson, and what symmetry could explain such a particle. Let us here only consider the $(g-2)_\mu$ example. From the relative $(g-2)_\mu$ enhancement of 10^{-9} presented in equation 1.14 with respect to the electroweak scale at 10^2 GeV, we can predict an approximate mass for X if we suppose that it couples to μ in a similar way as weak bosons:

$$\begin{aligned} \frac{\delta(g-2)_\mu}{(g-2)_\mu} &\approx \frac{M_W^2}{M_X^2} \\ \Leftrightarrow M_X &\approx M_W\sqrt{10^9} \approx 10^6 \text{ GeV}, \end{aligned} \tag{1.67}$$

which is well above the LHC energy scale. This prediction is of course very incomplete and would have to be refined for all different coupling types and strengths. We presented this argument for pedagogical reasons and to show in a simple case the link between one experimental tension and a possible NP effect.

There is no fundamental argument to ensure that there are new particles at scales close to the electroweak scale, typically at the $\text{TeV} - 10\,\text{TeV}$ range. This is because we already know, from the SM, that it is possible to have large energy gaps between particles masses. We have in the SM

$$\frac{m_e}{m_\nu} > 5 \times 10^5, \tag{1.68}$$

which is an enormous gap. If a similar gap exists between the electroweak scale and new particles, it means that the new particles would have masses above 10^5 TeV which is not reachable with our current technology or even in the next decades.

However, particle physicists can discover very massive particles without having to reach the required energies to produce them. It is possible to use **high-luminosity** [8] experiments rather that **high-energy** ones. Thanks to quantum field theory, we are able to make extremely precise predictions from a model. As massive particles contribute in Feynman diagrams, it is possible to deduce their existence even from experiments at much lower energies. In the above example, we showed that the knowledge of $(g-2)_\mu$ at a relative precision of 10^{-9} can probe elementary particle physics effects up to the 10^5 TeV scale. In the limit of our quantum field theory validity domain, there is therefore no limitation in the energy scales we can probe provided that we have sufficiently precise measurements based on a large number of events.

8. The luminosity refers to the number of events in a collider. The higher is the luminosity, the more events we observe and the more precise is the statistics.

1.4 Need for automated calculations

The picture

After presenting the SM together with motivations to go beyond, let us discuss the way BSM scenarios are studied. First, it is important to distinguish two different families of new physics models:

- **Theoretically motivated models.** These are not necessarily based on a phenomenological motivation such as the ones presented in the previous section although they often try to address some. In particular, they do not extend the SM but explore a new sector to understand its properties.
- **Phenomenologically motivated models.** These scenarios extend the SM and try to address some specific questions raised by experiments while maintaining the same predictive power as the SM.

Although all the following arguments apply to a certain extent to theoretically motivated models, the demand is more important for phenomenology because one has in principle to perform a very large number of calculations in all the newly suggested models. The real limitation in these studies is to maintain "the same predictive power as the SM". These predictions have been calculated since decades and measured in different experiments. We now have a very large number [9] of SM predictions that have been compared to measurements. A new model should therefore be tested on all these observables but this requires to calculate all the associated predictions. We know from our SM experience that it is a very long task involving the high energy physics community for decades and we obviously cannot do it for the hundreds of models out there. Consequently, BSM scenarios cannot be studied in detail without automation.

The purpose

Among the hundreds of models that are candidates to solve phenomenological issues, it is likely that many of them are not relevant to describe nature [10] but we cannot realize it yet as we do not possess a systematic and general automated tool to derive predictions from all these scenarios.

Figure 1.11 presents one possible representation of a general tool to perform systematic BSM analyses. Question marks represent the main steps for which automation is required. To be able to provide such a tool, all the procedures must be **model-independent** i.e. not specialized for a given type of model.

There three main procedures to automate for general BSM phenomenology are:

9. This number is hard to estimate exactly. The Particle Data Group (PDG) [35] is reedited every year and enumerates the main experimental measurements in particle physics. It represents very well the state of our knowledge.

10. A BSM model can always be at least as relevant as the SM to describe nature by setting its parameters to very small values. However, search for BSM scenarios must be driven by SM deviations or explicit new phenomena. With a detailed analysis one could show for example that the new model cannot bring anything new to the picture without breaking other observables.

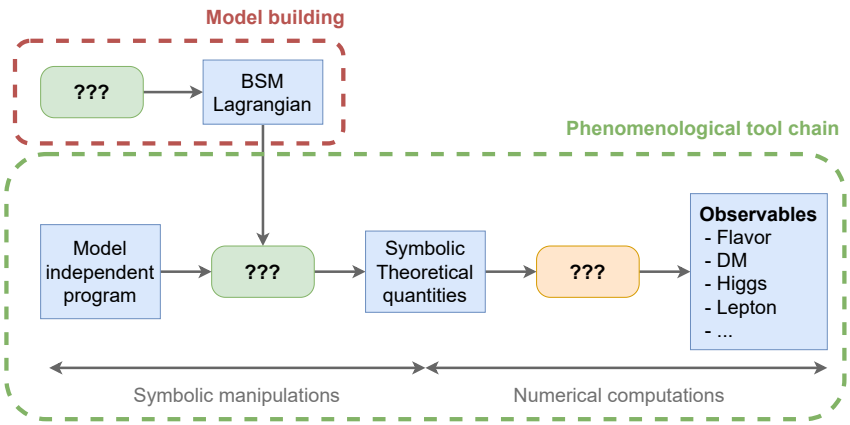


Figure 1.11 – Concept of a tool chain to automate systematic BSM phenomenology. From a BSM Lagrangian and a model-independent program, we want a tool chain able to predict observables in all domains of particle physics. Question marks represent places where automated tools are required.

- **Model building.** This part cannot be fully automated but is not a time-critical part. For example, one can decide to add a right-handed neutrino in the SM singlet gauge representation to create a new model. For practical purposes however, model building must be assisted by automated methods allowing us to derive easily the exact BSM Lagrangians.
- **Symbolic theoretical calculations.** These calculations, from a BSM Lagrangian, are model-dependent and must be performed analytically. They represent the steps that can take decades to perform and therefore are the main limitation to BSM phenomenology.
- **Numerical observable calculations.** From the analytically simplified theoretical quantities, observables can be calculated numerically in a model-independent way for all domains of particle physics. This last step in the prediction derivation is simpler to automate and as we will discuss several tools are already dedicated to it.

Theoretical calculations

Symbolic theoretical calculations from the Lagrangian are very time-consuming and prevent us to derive exhaustive predictions from BSM models. These calculations can be described using the so-called Feynman diagrams as demonstrated in figure 1.12.

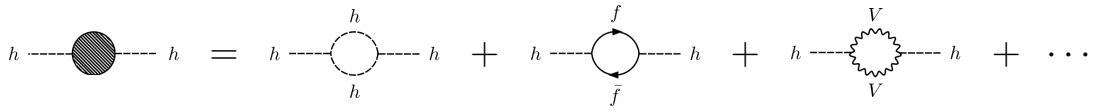


Figure 1.12 – Calculation principle of the 1-loop corrections to the Higgs mass. Several Feynman diagrams have to be calculated and the resulting amplitude is the sum of all contributions. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

In principle, the transition amplitude from an initial state i to a final state f can be

defined as follows

$$i\mathcal{M}(i \to f) = \sum_{N_L=0}^{+\infty} \sum_{d \in D_{N_L}(i \to f)} i\mathcal{M}(d), \tag{1.69}$$

with a sum over N_L the number of loops, $D_{N_L}(i \to f)$ the set of all possible diagrams with exactly N_L loops for the process $i \to f$ in the theory and $i\mathcal{M}(d)$ the amplitude of the diagram d. Each additional loop makes the contribution smaller and a diagram with N_L loops is proportional to

$$\left(\frac{\alpha}{4\pi}\right)^{N_L+E_0}, \tag{1.70}$$

with E_0 the exponent of tree-level diagrams, and α coupling constants typically of the order of $10^{-1}/10^{-2}$. Starting from tree-level diagrams, one generally obtains one or two more significant digits when considering the next loop level.

From the transition amplitude $i\mathcal{M}$ one can calculate more relevant quantities for phenomenology such as squared amplitudes $|\mathcal{M}|^2$ that require again more algebra or Wilson coefficients that are identified as complex-valued functions in front of particular structures in the amplitude. Squared amplitudes are crucial for collider phenomenology and Wilson coefficients are used in particular in flavor physics. More details about these different calculations are given in chapter 6.

For phenomenology we need one-loop calculations ($N_L = 1$) because they often represent the leading contributions such as in the motivations we discussed in the previous section: Flavor anomalies, the muon anomalous magnetic dipole moment and Higgs mass corrections are example of observables that require one-loop calculations. Unfortunately, calculating an amplitude at the one-loop level is already a very hard task for one process. If one wants to perform a detailed phenomenological analysis of a BSM model, hundreds of processes have to be calculated and it is simply not possible to do it by hand. This is why we need automated software programs able to perform analytical theoretical calculations from general BSM Lagrangians, at tree-level and one-loop.

1.5 Existing packages

In the following we present general purpose software programs automating theoretical calculations from the Lagrangian or an equivalent description. Codes that are mostly dedicated to SM observables are therefore not discussed here. There are two main types of codes performing this sort of calculations:

- **Mathematica packages.** Mathematica [68] is a commercial and closed software for symbolic manipulations. As the theoretical calculations must be performed analytically, several physics programs are based on this computer algebra system to implement high energy physics calculations.
- **Open-source solutions.** Other programs are free of Mathematica and use their own symbolic computation tools to carry out the calculations or hard-code some identities and equations to derive specific observables.

1.5.1 Mathematica packages

Mathematica is a powerful computer algebra system, packages written with it therefore logically provide more general features than the open-source codes which will be presented in the next section.

FeynRules

FeynRules [69] calculates the expression of Feynman rules in BSM models from their Lagrangian. From the vertices it can straight-forwardly provide $1 \to 2$ decay rates by squaring the interaction and efforts are made for loop-level calculations [70]. A main strength of FeynRules is that it can provide Feynman rules as input to other computer programs such as FeynArts, CompHEP, MadGraph5_aMC@NLO, ..., that we present in the following. These interfaces are possible thanks to the Universal Feynman rule Output (UFO) [71].

FeynArts / FormCalc

FeynArts [72] and FormCalc [73] have been developed jointly to provide a powerful tool for general BSM phenomenology. The role of FeynArts is to initialize amplitude expressions by finding all possible diagrams, drawing them and applying Feynman rules that must be given as input, for example from FeynRules. Then, FormCalc uses FORM [74] and Mathematica to fully simplify the amplitudes at tree-level or at one-loop in a large variety of BSM models. These models include for example general $SU(N)$ gauges and not only the SM gauge. FormCalc can also square amplitudes for collider physics. It is written in a very general way and provides a solution to the issue presented in section 1.4.

FormFlavor

FormFlavor [75] is based on FeynArts / FormCalc and calculates Wilson coefficients at the one-loop level. The Wilson coefficients are straight-forward to obtain at the tree-level once the amplitude can be derived. As it will be discussed later in section 6.5, the real difficulties come at the one-loop level because additional simplification procedures must be implemented to extract the coefficients. Although FormFlavor could in principle be used to calculate general BSM Wilson coefficients, it has been developed specifically to derive them in MSSM scenarios with non-minimal flavor violation.

SARAH

SARAH [76] is closely related to the Fortran code SPheno [77,78] and mostly addresses the phenomenology of SUSY models. It is however able to handle also other BSM scenarios and provide several methods to calculate specific quantities such as one-loop Renormalization Group Equations (RGE), two-loop RGE in SUSY models, two-body decays at the one-loop level, ... This code is written in a less general way than FormCalc for example. This makes it more suitable for SUSY-specific analysis but cannot be used as a general tool for BSM phenomenology.

1.5.2 Open-source solutions

LanHEP

LanHEP [79] is a Feynman rule calculator i.e. derives expressions for vertices in the theory directly from its Lagrangian. It is a C program implementing its own symbolic manipulation routines. From the vertices calculated by LanHEP, other packages such as CompHEP, CalcHEP or MadGraph5_aMC@NLO, that are discussed next, are able to calculate automatically squared amplitudes in models that LanHEP can build in particular, in the Standard Model gauge group presented in equation 1.3 or beyond.

CalcHEP and CompHEP

CalcHEP [80] and CompHEP [81] are open-source Lagrangian-level tools using their own symbolic computation frameworks to calculate tree-level squared amplitudes analytically. They rely on a vertex calculator such as LanHEP that provides the Feynman rules for the diagram calculations. The automation of a large number of calculations is also more difficult in CompHEP as it has been originally designed with a graphical user interface to be very simple to use for few processes.

MadGraph5_aMC@NLO

MadGraph5_aMC@NLO [82] is a python / C++ / Fortran code automating the calculation of cross-sections that can be used directly for event generation in the SM and BSM scenarios at tree-level and at one-loop. It can take as input Feynman rules from UFO files generated by vertex calculator such as FeynRules or LanHEP in order to extend some of its capabilities to BSM models. MadGraph5_aMC@NLO is very powerful for the calculation of SM processes. [11] Although it has several capabilities for BSM models, it has not been designed originally for this purpose and is therefore limited. In particular, it does not rely on a computer algebra system and calculations are restricted to the quantities that have been explicitly implemented by the developers, unable to provide general symbolic results for cross-sections.

1.5.3 Limitations

As we discussed in this section, several packages exist automating different theoretical calculations. Before going further let us recall the scope of our discussion. Our purpose is to design a tool for general BSM phenomenology as presented in figure 1.11. While codes that are specific to SM or SUSY calculations are very valuable because we still need to investigate further these models, we consider here a broader scope. In particular, the SM will probably have to be extended because of future experimental breakthroughs and SUSY models are not the only options. We are therefore looking for a software program able to manage all different types of models and calculations in a very general way.

11. See the online interface <http://madgraph.phys.ucl.ac.be/>

Specialization

Usually, computing packages are built first from common cases and then extended for more general purposes. This implies a lot of work for developers that can only provide a finite number of features because they have to be hard-coded to some extent. One-loop calculations in particular are often considered in the SM case. Without a general implementation on all aspects it is great challenge to generalize such calculations to BSM scenarios.

`FormCalc` on the other hand is written in a very general way by using all the symbolic manipulation abilities of `FORM` and Mathematica. Models are not much constrained and calculations are performed using general quantum field theory relations and not limited to specific processes.

Mathematica

The most effective tools for general BSM phenomenology are written with Mathematica. This is in particular the case of FeynRules and FormCalc. Codes not relying on Mathematica such as `CompHEP`, `CalcHEP` or `MadGraph_aMC@NLO` are not able to provide a full one-loop support as it requires high-performance analytical calculations. One is therefore forced to use Mathematica in order to obtain relevant quantities for BSM phenomenology. This implies multiple drawbacks:

- **Closed software.** Mathematica is closed meaning that the source code is not public and we can only use it as a black-box. For scientific purposes, this is very inconvenient.
- **Commercial software.** Mathematica is expensive, in particular each active session of this program must correspond to a purchased license. Even if the Mathematica packages are themselves free and open-source, one has to pay for a Mathematica license to use them.

It is preferable for the high energy physics research not to rely on this software and to provide its own free and open-source tools, even for general BSM phenomenology.

Wilson coefficients

No computer program, even using Mathematica, is for now able to provide Wilson coefficients for general BSM scenarios at the one-loop level. As we discussed in section 1.2, these quantities are required in particular to study the impact of new physics on flavor anomalies, one of the most promising experimental hints that the Standard Model is incomplete.

Model building

Model building is not really considered in general and the standard procedure to define a model is to write explicitly the whole Lagrangian by hand. This feature has some unfortunate consequences:

- Giving the explicit Lagrangian of BSM models is a very tedious and error-prone task as the number of terms grows rapidly and their exact form depends heavily

on conventions that can be inconsistent between different references. In the unconstrained MSSM for example, there are about 10^4 interaction terms. [12] This is a problem considering that results derived from the model strongly depend on the correctness of the different vertices.

- (B)SM models are better described at high energies, with all symmetries preserved. For a newly constructed BSM model, one therefore has to derive the low energy Lagrangian by calculating the different steps of symmetry breaking. This is again another possible way to introduce mistakes.

A solution to this issue is to provide assistance for model building through automated routines as it was suggested in figure 1.11.

1.6 MARTY

1.6.1 Presentation

Motivated by the arguments introduced in section 1.2 we present MARTY [83–85], a C++ program published under the terms of the GNU GPLv3 license, that complements the ecosystem of existing packages listed in section 1.5 by providing unique features which were lacking in the previously published computer programs. More information on MARTY can be found in the website [13] on which the user manual [86] and the documentation [87] can be found. The basic principle of MARTY in the context of systematic BSM analyses is shown in figure 1.13.

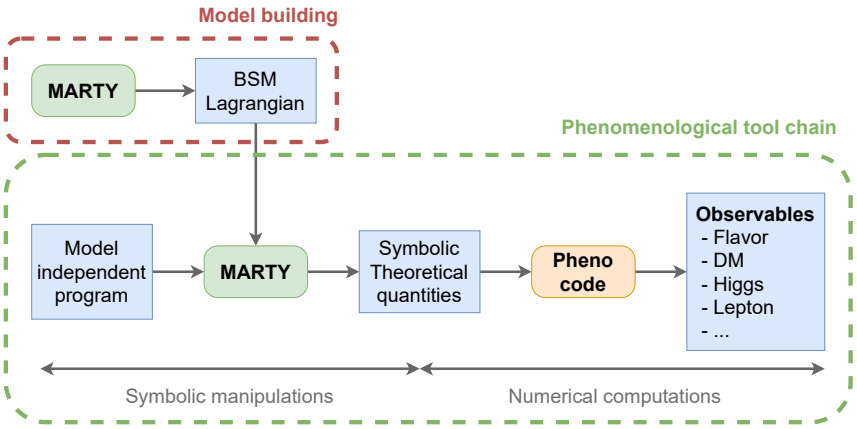


Figure 1.13 – The complete version of the tool chain in figure 1.11 filled in. The existing and open-source codes for phenomenology can perform the numerical computations and MARTY takes the responsibility to do the theoretical symbolic calculations and to provide assistance for model building.

MARTY stands for **M**odern **AR**tificial **T**heoretical ph**Y**sicist and is dedicated to symbolic theoretical calculations for general beyond the Standard Model scenarios, at tree-level and at one-loop. MARTY's main features are the following:

12. The exact number depends also on conventional choices.
13. See <https://marty.in2p3.fr>.

- It is **independent** of any other external program and in particular does not rely on Mathematica. In particular, MARTY enters the category of open-source programs.
- Thanks to its CSL module, MARTY is a **symbolic computation program**. It can therefore manage all kinds of mathematical expressions and calculations. Although it is a part of MARTY, CSL has a separate user manual [88] and its own documentation [89].
- Implementations are **very general**, not specifically adapted to any kind of model. In particular, any arbitrary combination of **semi-simple Lie groups** can be used as the gauge group for a model (not limited to $SU(N)$ in particular, see chapter 5 for more details).
- Amplitudes, squared amplitudes and Wilson coefficients can be derived in **all models, at tree-level and at one-loop**.
- All the calculation mentioned above are performed **automatically**, **symbolically** and **without approximation**. [14]
- **C++ numerical libraries** are generated automatically and contain functions implementing the symbolic formulas derived for scalar quantities such as squared amplitudes or Wilson coefficients.
- **Model building** is supported by a lot of built-in functions that automate modifications of the Lagrangian such as replacements, field rotations, diagonalization, symmetry breaking, ...
- **Spectrum generation** with general mixings is supported in all models in MARTY. In particular, a dedicated C++ spectrum generator for any given model can be generated automatically in the same C++ library that contains the results.

1.6.2 Limitations

MARTY is very general but still has some limitations. Besides the ones implicitly given in the features list, the main limitations are:

- **Dimension 4 space-time.** While dimensional regularization at the one-loop level promotes calculations to D-dimensions, models must be defined in $4D$.
- **Spin 0, 1/2 and 1 particles.** A spin $1/2$ particle can be a Weyl, Dirac or Majorana fermion. Scalar and vector particles are also supported. Higher spin fields such as spin $3/2$ or 2 cannot be described yet.
- **Complete NLO description**. All one-loop quantities can be calculated, and the complicated theoretical calculations are taken care of automatically. However, for well-defined NLO quantities more work is needed to combine the relevant tree-level and one-loop calculations.

These constraints could be lifted by extending MARTY to the relevant calculations. As it is a very general and modular code with a complete symbolic manipulation support, it is possible to implement the missing procedures.

14. Except of course the perturbative expansion on which quantum field calculations are based.

1.6.3 Connection to phenomenology

The scope of MARTY is the theoretical side of phenomenology, namely all the long and error-prone steps from model building to final symbolic theoretical quantities. As shown in figure 1.13, MARTY does not participate beyond the derivation of these quantities. For example, squared amplitudes at tree-level and one-loop in general BSM models can be calculated but cross-sections must be derived by users. As discussed in section 6.4, the complicated theoretical calculation stops when the squared amplitude is fully simplified analytically. Then, a simple numerical program can derive the cross-section as we present later in chapter 7. Therefore, unlike programs such as MadGraph_aMC@NLO or CompHEP, MARTY delegates numerical calculations as they are already performed by other reliable and open-source packages.

SuperIso [90–93] for example can derive flavor observables in a model-independent manner from the values of Wilson coefficients, and SuperIso Relic [94–96] calculates the dark matter relic density together with direct and indirect detection rates from squared amplitudes of $2 \to 2$ processes in a given model. Interfaces with these two software programs are already a work in progress.

In general, the way to obtain a phenomenological tool chain such as the one illustrated in figure 1.13 is to implement interfaces with codes on the phenomenology side. In this way, the full chain can be automated and allows one to perform easily detailed analysis of any new BSM model. An interface with GAMBIT [97] together with its different modules [98–103] could be very interesting as it provides general tools to compare BSM models to experimental data and the SM in all domains of particle physics, starting from theoretical quantities which MARTY can deliver also in a very general manner.

An overview of MARTY is presented in chapter 2, introducing in particular the software requirements and its two independent modules: CSL and GRAFED. Chapters 3, 4 and 5 will present MARTY in more details through its quantum field, particle physics model and group theory implementations respectively. Then, specific calculation procedures for BSM phenomenology and their impact on the code development are detailed in chapter 6. As a demonstration of MARTY's ability to automate tree-level and one-loop calculations for general BSM scenarios, a selection of results will be presented in chapter 7. Finally in chapter 8, new analytical results we have obtained at the loop level with MARTY in Non-Minimal Flavor Violating (NMFV) MSSM scenarios are presented.

CHAPTER 2

MARTY – An open-source solution

MARTY [83] is the solution we propose to the technical issues that have been presented in the previous chapter. The basic principle is to provide an independent computer program able to calculate theoretical predictions automatically for Beyond the Standard Model scenarios, starting from the Lagrangian. This requires symbolic manipulations as those calculations must be done analytically. MARTY is a free and open-source all-in-one package, independent from any external program, implementing its own C++ Symbolic computation Library (CSL). This chapter introduces the solution first by presenting the requirements of MARTY then by explaining what are symbolic computations and finally how it is done by CSL and used in MARTY.

2.1 Requirements

Requirements are the set of conditions a program must fulfill. The obvious one, that is always implicit, is that the program must work. Then one has to specify what strengths the program must have (simplicity, performance, development time, etc). This section is about the main specifications of MARTY that have driven its development, with each time the costs that are associated with them.

2.1.1 Generality

MARTY must be as general as possible, not specifically designed for a few BSM scenarios or use cases. The advantage is that the code can be used in many different physical studies, which is a very powerful feature. There are two main costs when writing a general purpose code:

- Simple cases are more difficult to optimize. In the case of a calculation program such as MARTY, it means that simple problems are solved using a general solution that is not particularly adapted. Results are then harder to obtain and can be less simplified mathematically.
- Development time. Writing a general code requires time to first understand the

global picture and take into account all edge cases to finally implement the relevant software architecture.

If the costs above are affordable, generality is a great reward. In the case of MARTY there are two main tasks to implement in a general way, model building and calculations.

Models

Before making any prediction, MARTY must be able to store and manipulate theories Beyond the Standard Model. For phenomenological purposes, the most important features concern particle types and group theory [1]. To describe most of BSM theories, one must handle

- Spin 0 particles, Lorentz scalars.
- Spin 1/2 particles with Weyl, Dirac and Majorana fermions.
- Spin 1 particles, vector bosons.

In terms of group theory, most models are based on $SU(N)$, $SO(N)$ and $U(1)$ gauge groups. In MARTY we decided to implement a general description of semi-simple Lie groups, including

- The abelian $U(1)$ group, used for example to describe electromagnetic interactions.
- $SU(N)$ groups, describing the strong and weak nuclear interactions in the Standard Model.
- $SO(N)$ groups.
- $Sp(N)$ groups.
- The 5 exceptional groups E_{6-8}, F_4 and G_2.

This is probably more general than what is required solely by phenomenology but the formalism of semi-simple Lie algebras allows us to describe all these groups at once, and this generality could extend the range of MARTY's applications to more theoretical capabilities.

The main limitations of models that can be built in MARTY are:

- The space-time, limited for now to a $4D$ Minkowski space.
- Particles spins, limited to spin 1. Spin $3/2$ or spin 2 (such as the hypothetical graviton) can also be interesting phenomenologically but cannot be constructed in this framework.

Calculations

The main purpose of MARTY is to automate calculations Beyond the Standard Model for all scenarios that can be defined and to provide routines for amplitudes, squared amplitudes and Wilson coefficients at the tree and one-loop levels. Such calculations are already a challenge by hand and they represent an even bigger difficulty to automate. Being general at the calculation level requires to implement simplification procedures

1. Particle types (a.k.a. spins in MARTY) can also be described with group theory as the spin corresponds to the Lorentz group representation. Spin is however most of the time considered independently.

that are context independent, i.e. the model, particles involved, number of diagrams, expressions for the vertices, etc. This may be the biggest challenge in the development of MARTY for the two following reasons:

- One has to understand all aspects of perturbative calculations to build a general procedure. This includes Dirac algebra, group theory, loop calculations, etc.
- By definition, a general procedure must not fail in any case meaning that one must build a program that can adapt to any calculation without destroying performance, a very difficult task in symbolic computations when manipulating large expressions. [2] The challenge is then to find the minimal set of instructions that can solve the general problem.

2.1.2 Performance

The problem that MARTY addresses is performance-critical for several reasons. Symbolic manipulation is not a task that fits well computer architectures, it requires highly dynamical programs known to be slow. Furthermore, one-loop calculations generate very large expressions and add more difficulty to the initial task. Finally, as we discussed in the previous section, generality has a high structural performance cost. For all these reasons C++ has been chosen as the programming language for MARTY and many efforts have been dedicated to performance, being for the execution time or the amount of memory the programs need to run.

There are two main challenges to optimize the execution time for such a program. The first one is to have a very good knowledge of C++ to take the best advantage of it. Secondly, one must maximally reduce unnecessary calculations, a very complex task in a general code relying on symbolic manipulations that must simplify correctly all possible expressions.

The amount of memory is also a main concern as expressions can have very large sizes and require many optimizations to be stored in the memory of a laptop. The first key to reduce memory consumption is obviously to have light-weight symbolic objects, carrying the absolute minimum of information with them. This is however not enough to perform general calculations beyond the Standard Model and one needs to introduce relevant [3] abbreviations in expressions to compress the final results. For example, two expressions

$$x = 1 + (a+b)(a-b),$$
$$y = 3 + \frac{(a+b)}{(a-b)},$$

2. A good example of this issue is expansion. An ultimate way to be fully general in perturbative calculations is to expand the entire amplitude. However, this cannot be done because it would generate gigantic expressions in many cases, that could not be handled by any software.

3. The more the abbreviation is used in other expressions the more relevant it is.

can be compressed with the following definitions:

$$x = 1 + Ab_1 \cdot Ab_2,$$
$$y = 3 + \frac{Ab_1}{Ab_2},$$

with

$$Ab_1 = a + b,$$
$$Ab_2 = a - b.$$

This principle allows us to store only once an expression in the entire program and use its abbreviation several times, which is much lighter than its encapsulated expression. This method saves time, space, and produces more compact results. It is therefore crucial in general for the performance of MARTY and any computer algebra system manipulating large expressions.

Performance optimizations have, as any optimization, a cost. In general and in the particular case of MARTY, a better performance means a more complex code, harder to develop and maintain. A common piece of advice in software engineering is to never optimize any code that is not performance critical because one will have to pay the costs without getting the benefits.

2.1.3 Software engineering standards

MARTY is designed for the members of the high energy physics community, i.e. users and developers. It must therefore comply to high quality standards for the user interface and the internal code structure.

User interface

The user interface must be simple and not demand a deep C++ knowledge. This means that most of the features must be encapsulated in simple interface functions hiding all object types and data transfer. As an example of interface, let us consider the (squared) amplitude calculation within a model `toyModel` for a $\mu \to \mu\gamma$ process:

```
auto amplitude = toyModel.computeAmplitude(
    Order::OneLoop,
    {Incoming("mu"), Outgoing("mu"), Outgoing("A")}
); // calculating mu -> mu A at one loop
Display(amplitude);
Show(amplitude);

auto squaredAmplitude = toyModel.computeSquaredAmplitude(ampl);
Display(squaredAmplitude);
```

Using correctly old and modern C++ features one can hide most of the implementation details and simply ask users the minimal required quantity of information, i.e. the order of development (`TreeLevel` or `OneLoop`) and field insertions. In this example, the `auto`

keyword (C++11) allows the compiler to automatically deduce the returned type, users thus do not have to know it. The amplitude is returned in a variable of a given type, and used in other interface functions such as `Display()` (writing symbolic results in standard output), `Show()` (displaying Feynman diagrams in GRAFED) or the squared amplitude calculation (squaring the result). A more advanced user will know that the amplitude is of type `mty::Amplitude`, that the squared result is a simple `csl::Expr` and how to customize the output. In most MARTY use cases this is however not required and a basic knowledge of the user interface is enough to perform all types of calculations.

Code quality

MARTY is also meant to be developed and extended by the community. As any software package designed to be used during a long time by several people, MARTY must comply to general software engineering standards:

- **General code readability.** The computer program, considered as a pure text file for the moment, must be readable. This means that style conventions must be fixed [4] and be followed during the whole development. Rules are often partially arbitrary because they are subjective. There are tools [5] that provide a way to define clear conventions and even to apply them automatically on any C++ source file.
- **Specific code readability.** Once the code is a readable text, it must be a readable computer program. In order to improve code quality, a same problem may be expressed differently in C++ depending on the context as we discussed in sample code 1.
- **Separation of concerns.** This principle is probably the one with the biggest impact on the overall code structure. It states that two different problems should be handled by two separate logical units. This applies at all abstraction levels: one instruction per line, one irreducible task per function, one purpose per object. To give an example at the level of functions, a common piece of advice is to limit their size to 20 lines maximum. This may seem arbitrary but it is true that most of the time a bigger function can be separated in several more fundamental tasks.
- **Modularity.** This principle is the separation of concerns applied at the biggest scale of the code. In MARTY the main modules are CSL for the symbolic manipulation, GRAFED to draw Feynman diagrams and finally the physics core of MARTY. In particular CSL and GRAFED do not contain any physics consideration and can be used as standalone. This is modularity, i.e. trying to maximally decouple large parts of the code that do not fundamentally need to be entangled together.
- **Simplicity.** From the point of view of software engineering, the best solution to solve a problem is not the most impressive or elegant one, but the **simplest** one. A simple solution is easier to write, read, maintain, and is the best investment one can make.

4. See the example presented on this web page <https://named-data.net/doc/ndn-cpp-dev/0.4.0/code-style>.
5. See the online clang-formatter <https://zed0.co.uk/clang-format-configurator/>.

Sample code 1: Simplicity in C++

We consider the iteration through a container of integers:

```
std::vector<int> vec {1, 2, 3, 4, 5};
```

If no index is needed, the range-for loop should be preferred:

```
for (int x : vec) {
    std::cout << x << '␣'; // >> 1 2 3 4 5
}
```

If an index is needed however, this loop becomes more difficult to understand:

```
size_t i = 0;
for (int &x : vec) {
    x = 2*i + 1;
    ++i;
}
```

In this case, the classic for loop is simpler:

```
for (size_t i = 0; i != vec.size(); ++i) {
    vec[i] = 2*i + 1;
}
```

Note Compiler optimizations guarantee that equivalent implementations will have no measurable difference in terms of performance, letting us express better our intent when writing a piece of code.

Different kinds of complexity

Let us balance the concept of simplicity with the difference between essential and accidental complexity in software engineering [6] introduced by Frederick Phillips Brooks Jr., *"There is no silver bullet"* [104]. The essential complexity is a fundamental consequence of a problem's structure whereas the accidental complexity is a side effect coming from the implementation choice. In terms of development best practices this has two implications:

- Accidental complexity should be maximally reduced at all development steps. The choice of technology, the architectural design of the code, the separation of concerns and finally micro-optimizations must be adapted.
- Essential complexity cannot be reduced and must be identified early in the development for two reasons. First, no implementation can reduce it and resources should not be spent in that purpose. Then, identifying it allows a developer to have a deeper understanding of the problem and consequently to avoid accidental complexity.

In this section we have presented general coding standards that we followed to develop MARTY. It is important to mention that coding standards and code requirements are often in conflict. To give an example in the case of MARTY, **generality** is always achieved at

6. Although it has been formulated in the context of software development, these concepts can be applied to any kind of problem solving tasks including physics modeling for example.

the cost of **simplicity**, because the more different cases there are the more complex MARTY must become to handle all of them, even if most of the use cases are simple. This is one consequence of the fact that there is no such thing as a perfect software program. Even at the level of micro-developments there is almost never one best solution but rather solutions that are better than others for particular purposes. Knowing that, the software developer must always find a good balance when choosing a particular solution and learn to compromise at all stages of software development.

2.2 Symbolic manipulations

A symbolic manipulation software has the ability to store and manipulate abstract mathematical expressions. In programming, an expression like `2*a + b` is generally a numerical calculation, `a` and `b` having definite values, and the result can be computed immediately. A symbolic manipulation software can manipulate such mathematical expressions without having to know any value for the variables, keeping their symbols all along in full generality. This requires a whole system to store internally symbolic expressions and manipulate them in a mathematically consistent way. This section presents core principles of symbolic manipulation programs and how CSL applies them.

2.2.1 Internal representation of an expression

Expressions can naturally fit in trees. They are in general arbitrary functions of others, $\cos(x)$ for example can be a valid mathematical expression whatever is x (variable, sum, another function, etc). The expression

$$A\left(1+\cos\frac{2\pi t}{T}\right), \tag{2.1}$$

can be represented in a tree as figure 2.1 shows.

Expressions can be composed with each other arbitrarily. It means that one particular node may be any expression. The tree representation is therefore of much relevance because one node can be added, changed or removed without having to modify or copy the rest of the expression. A linear representation (a character string for example) would not provide such an advantage as a modification in the middle would in general require to move contiguous elements forward or backward even if they have not been modified.

2.2.2 Dynamic programming and polymorphism

When summing two symbolic expressions, the result's type cannot be known by the compiler as it can be any type of expression: $a+b$ is a sum, $a+a=2a$ a product, $a-a=0$ a number, etc. In terms of C++ software engineering this is very challenging because object types cannot be determined at compile-time, meaning that the code must

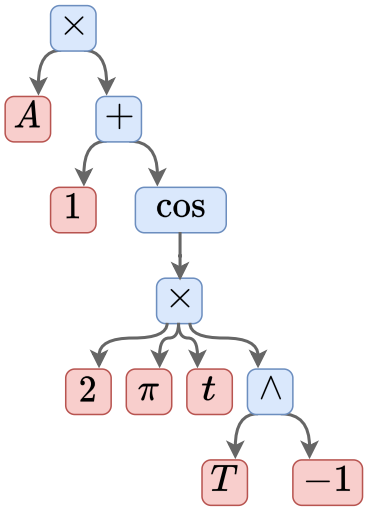


Figure 2.1 – Tree representation of $A\left(1+\cos\frac{2\pi t}{T}\right)$. Red nodes (leafs) are building blocks, and intermediate nodes are functions of others.

be highly polymorphic [7] and perform a large number of memory allocations / deallocations (known to have a high performance overhead). Dynamic programming is at the center of any symbolic manipulation system to store arbitrary expressions in a computer program [8]. Once their storage has been set properly, the system must provide a way to modify and simplify them. Some simplifications must be done automatically to keep expressions manageable in a computer program. This is the role of expression canonicalization presented in the following section.

2.2.3 Canonical forms of expressions

In CSL and most of the existing computer algebra systems, expressions are always kept in a canonical form when created. Canonicalization is the set of fundamental simplifications applied to all expressions created in the program to make them fit into precise definitions. It is mandatory for symbolic computations to obtain not only correct, but readable and manageable expressions in space (memory) and time (execution speed). Those simplifications are performed every time an expression is created, and must therefore be as fast as possible.

Here are some of the canonicalization rules of CSL:

- A term of a sum cannot be itself a sum ($a+(b+c)\mapsto a+b+c$).
- An element of a product cannot be itself a product ($a\cdot(bc)\mapsto abc$).
- If a sum or product contains only one element it is converted into this element.

7. A polymorphic algorithm is defined as a unique piece of code that can be used for different types of objects. Symbolic manipulations require this behavior at all time as, for example, the + operation for two operands A and B ($A+B$) must be valid for all possible A and B whatever their type is.

8. Polymorphism and dynamic programming are not a design choice to solve this particular problem. They are the fundamental consequence of analytical calculations in mathematics and cannot be bypassed: any solution to this problem should have these two properties one way or another.

- The repetition of an operation is transformed into the appropriate operation: $a + 2a \mapsto 3a$, $2a \cdot 4a^2 \mapsto 8a^3$.
- Special values of sums, products or functions are applied immediately ($0x \mapsto 0$, $1x \mapsto x$, $\cos(0) \mapsto 1, \ldots$).
- There are no subtraction or division operations in the program (except integer fractions): $a - b \mapsto a + (-1) \cdot b$ and $a/b \mapsto a \cdot b^{-1}$.

Example of canonical expression Let us illustrate the canonicalization of expressions with an example: The Taylor development of $\cos(\pi/2 + \omega t)$ around $t = 0$ up to the order t^2 included. The standard calculation by hand gives:

$$\begin{aligned}\cos(\pi/2 + \omega t) &\approx \cos(\pi/2) - \omega t \sin(\pi/2) + \mathcal{O}\left(t^3\right) \\ &= -\omega t + \mathcal{O}\left(t^3\right)\end{aligned}$$

This takes two steps if we need one to evaluate special values of cos and sin functions. Let us see what a naive symbolic computer program would do without canonicalization. The rules followed by the program to differentiate general expressions with respect to a variable t are the following:

$$\begin{array}{lll}\text{sum:} & \frac{d}{dt}\sum_i f_i(t) & = \sum_i \frac{df_i}{dt}, \\ \text{product:} & \frac{d}{dt}\prod_i f_i(t) & = \sum_i \left(\prod_{j<i} f_j(t) \cdot \frac{df_i}{dt} \cdot \prod_{k>i} f_k(t)\right), \\ \text{composition:} & \frac{d}{dt}(f \circ g)(t) & = \frac{dg}{dt} \cdot \left(\frac{df}{dt} \circ g\right)(t).\end{array}$$

We note $T_n^f(x)$ the coefficient of order n in the development for a function f. It can be calculated recursively for efficiency, we therefore obtain intermediate steps $C_n^f(x)$ and evaluate the derivative part at the end:

$$\begin{aligned}C_n^f(x) &= \frac{d}{dx} C_{n-1}^f(x) \quad \text{for } n > 0, \\ C_0^f(x) &= f(x), \\ T_n^f(x) &= \frac{(x - x_0)^n}{n!} \cdot C_n^f(x_0).\end{aligned}$$

Applying these rules on $\cos(\pi/2 + \omega t)$ gives without canonicalization

$$\begin{aligned}&\cos(\pi/2 + \omega t) \approx \cos(\pi/2 + \omega \times 0) + \left(\frac{t}{1}\right)(0 + (0 \times t + \omega \times 1)) \times (-1 \times \sin(\pi/2 + \omega \times 0)) \\ &+ \left(\frac{t}{2}\right)\left(\frac{t}{1}\right) \times [0 + ((0 \times t + 0 \times 1) + (0 \times 1 + \omega \times 0)) \times (-1 \times \sin(\pi/2 + \omega \times 0)) \\ &+ (0 + ((0 \times t) + \omega \times 1)) \times (0 \times \sin(\pi/2 + \omega \times 0) \\ &+ (-1) \times ((0 + (0 \times t + \omega \times 1)) \times \cos(\pi/2 + \omega \times 0)))] .\end{aligned}$$

The result is mathematically exact but is two orders of magnitude larger [9] than the mathematically equivalent $-\omega t$. One can easily understand that such a naive symbolic manipulation system would be useless.

9. The size used for this measure is the number of tokens, i.e. the number of building blocks (numbers, variables), mathematical function (cos, sin) and mathematical operations (+, ×, ^).

Applying arithmetic rules on 0 and 1 simplifies drastically the result but it is still not acceptable. Letting parenthesis around sums or products that contain one element only we obtain

$$\begin{aligned}\cos(\pi/2+\omega t) \approx \cos((\pi/2)) + (t)(\omega)(-1\times\sin((\pi/2))) \\ + \frac{t}{2}\cdot(t)\left[(\omega)(-1\times((\omega)\times\cos((\pi/2))))\right](\omega).\end{aligned}$$

With canonicalization of sums and products with one element the result is better but not satisfying

$$\cos(\pi/2+\omega t) \approx \cos(\pi/2) + (-1)\times\omega t\cdot\sin(\pi/2) + \frac{-1}{2}\omega^2 t^2\cos(\pi/2).$$

Finally special values of functions can be applied to recover our calculation by hand:

$$\cos(\pi/2+\omega t) \approx (-1)\omega t + \mathcal{O}\left(t^3\right).$$

This example demonstrates the importance of always keeping expressions in canonical forms: it corresponds to steps of calculation that we neither explicitly nor consciously do by hand but that must be done by a computer program in order to produce manageable results.

2.2.4 Automatic ordering of expressions

Automatic ordering of expressions has been integrated in CSL in order to have human-readable expressions and to better simplify them. It allows to order different expressions by simplicity. For example we have

$$\begin{array}{rll} x & \text{is simpler than} & y, \\ xy & \text{is simpler than} & yx, \\ 1 & \text{is simpler than} & \cos(x)^2+\sin(x)^2, \\ x & \text{is simpler than} & \frac{x^2}{x+y^2/z}+\frac{x\cdot y^2/z}{x+y^2/z}. \end{array} \tag{2.2}$$

The ordering rule noted $<$ must be total, i.e. if E is the set of CSL mathematical expressions, the rule must respect the following conditions:

$$\begin{array}{l} \forall x,\ y,\ z\in E: \\ \textbf{if } x\leq y \textbf{ and } y\leq z \textbf{ then } x\leq z \text{ (transitivity)}, \\ \textbf{if } x\leq y \textbf{ and } y\leq x \textbf{ then } x=y \text{ (antisymmetry)}, \\ x\leq y \textbf{ or } y\leq x, \text{ (connexity)}. \end{array} \tag{2.3}$$

Once the order has been defined, expressions can always be kept sorted to improve readability and more importantly enable simplifications that would not be possible otherwise. CSL follows closely the rules given in [105] that provides a total order for mathematical expressions. Applying this order, we have for example:

$$tyx+1+\cos(x+1) \xrightarrow{\text{becomes}} 1+txy+\cos(1+x).$$

Table 2.1 shows some of the rules for CSL expressions. The specific rules referenced in this table are defined in the following:

- $\mathcal{O}_1$ for two **integers**: Arithmetic comparison of values of u and v.
- $\mathcal{O}_2$ for two **products**: Compares arguments from the end of the two products (last arguments). When the first non-equivalent arguments are encountered, the result is returned. Otherwise $u < v$ if u has less arguments than v.
- $\mathcal{O}_3$ for two **pow** objects: Compares arguments from the beginning of the two pow objects. When the first non-equivalent arguments are encountered, the result is returned. Returns false otherwise.
- $\mathcal{O}_4$ for two **sums**: Equivalent to $\mathcal{O}_2$.
- $\mathcal{O}_5$ for two **functions**: If the types of functions are different, returns the corresponding type order in `csl::Type`. Otherwise returns the order of the arguments of both function: $\mathrm{Arg}(u) < \mathrm{Arg}(v)$.
- $\mathcal{O}_6$ for two **indexed tensors** follows three steps.
  - If the two tensors have different names, returns the alphabetical order of them.
  - Otherwise if one of the tensors is complex conjugated and not the other, returns true if the right operand v is complex conjugated.
  - Otherwise returns the alphabetical order of the tensor index structures.

$u_\downarrow < v_\rightarrow$ table	Integer	$\cdot$	$\hat{}$	$+$	Function	Indexed tensor
Integer	$\mathcal{O}_1(u,v)$	true	true	true	true	true
$\cdot$	false	$\mathcal{O}_2(u,v)$	$u < \cdot v$	$u < \cdot v$	$u < \cdot v$	$u < \cdot v$
$\hat{}$	false	$\cdot u < v$	$\mathcal{O}_3(u,v)$	$u < v^1$	$u < v^1$	$u < v^1$
$+$	false	$\cdot u < v$	$u^1 < v$	$\mathcal{O}_4(u,v)$	$u < +v$	$u < +v$
Function	false	$\cdot u < v$	$u^1 < v$	$+u < v$	$\mathcal{O}_5(u,v)$	false
Indexed tensor	false	$\cdot u < v$	$u^1 < v$	$+u < v$	true	$\mathcal{O}_6(u,v)$

Table 2.1 – Sample of ordering rules in CSL. The rows correspond to u and the columns to v.

In case of doubt using CSL, one can at all time test the order of two expressions using `operator<()`, `operator<=()`, `operator>()`, `operator>=()`. This is presented in sample code 2.

Ordering is not only a cosmetic feature as it is of great importance for automated simplifications. The cost of always keeping expressions sorted is not negligible but allows us to have much better and simpler simplification algorithms. To illustrate this point let us consider two complicated cases of expression comparison that become trivial when expressions follow a total order, multi-argument functions and indexed tensor comparisons:

$$a + xb - 3 \stackrel{?}{=} -3 + bx + a, \tag{2.4}$$

$$A_{ji}A_{jk} \stackrel{?}{=} A_{kj}A_{ij}, \tag{2.5}$$

with $A_{ij} = A_{ji}$ a symmetric 2-dimensional tensor. Without ordering, one must write respectively $\mathcal{O}(N^2)$ and $\mathcal{O}(m^2 \cdot N^2)$ algorithms with N the number of different terms

Sample code 2: Examples of ordering test

```
#include <csl>
#include <iostream>
using namespace std;
using namespace csl;

int main() {
    Expr a = constant_s("a");
    Expr b = constant_s("b");
    Expr x = variable_s("x");
    Expr y = variable_s("y");

    cout << boolalpha;
    cout << (a < b) << endl; // >> true
    cout << (y >= x) << endl; // >> true
    cout << (cos_s(x) < sin_s(x)) << endl; // >> true
    cout << (1 + b*x > 1 + a*y) << endl; // >> false
}
```

and m the maximum number of indices for one tensor. By sorting expressions in sums, products, and indices inside (anti-)symmetrical tensor permutations the above equations become

$$-3 + a + bx \stackrel{?}{=} -3 + a + bx, \tag{2.6}$$

$$A_{ij}A_{jk} \stackrel{?}{=} A_{ij}A_{jk}, \tag{2.7}$$

which are trivially solvable with respectively $\mathcal{O}(N)$ and $\mathcal{O}(m \cdot N)$ algorithms that are also much simpler to write and maintain. While this argument seems to highlight a trade-off between sorting or comparison times, the choice is simpler than it looks like. Sorting is performed during expression construction that also requires comparisons (in particular for canonicalization, see section 2.2.3) while comparison is a pure read-only algorithm that does not create or modify expressions. In other words, performance while creating expressions is limited by both sorting and comparison so an equal trade-off between the two has no impact [10] while for comparison only, the performance gain is very important if sorting is enabled.

2.2.5 Limitations of symbolic computations

Computers are not smart

While it is often possible to automate some computations, one should keep in mind that the automation of symbolic computations come along with strong limitations. In

10. The trade-off is not equal as one could expect and is actually in favor of the 'sorting first' method. When comparing two linear collections for example, sorting and comparing is $\mathcal{O}(N \log N)$ while the unsorted comparison goes in $\mathcal{O}(N^2)$.

particular, all clever tricks a human may find to simplify complicated mathematical expressions are almost impossible to automate. A deterministic algorithm [11] can only test a predefined set of simplifications, but the addition of more tricks will increase the computational complexity much faster than the number of problems the system can solve. Considering the two following expressions

$$A = x^2/x, \tag{2.8}$$
$$B = x, \tag{2.9}$$

one can wonder if they are equivalent. While it is trivial for us to know that A and B are equal, a computer program will not in general know it if A is not simplified. One could write a more complex function to compare expressions, however symbolic manipulation have a high performance cost and thus require to have the least number of systematic operations possible. In this first example, canonicalization is enough because

$$x^2/x \mapsto x^2 \times x^{-1} \mapsto x^{2-1} \mapsto x^1 \mapsto x. \tag{2.10}$$

Considering a more complicated example

$$A = a \cdot (1 + x), \tag{2.11}$$
$$B = a + ax, \tag{2.12}$$

it is again clear that $A = B$ mathematically. However, no symbolic manipulation system would expand or factor automatically expressions as this would represent a catastrophic performance cost overall. Without extra help or particular care, a symbolic manipulation program will therefore answer that $A \neq B$, or avoid to give any answer in the best case scenario. A boolean mathematical statement for a computer program will not necessarily mean the same thing as for a mathematician. In particular, $A = B$ has different meanings in both. While a mathematician tries to know if the two objects are fundamentally identical, the computer program will stick to a structural comparison.

Take away This limitation makes computer algebra systems much more useful to perform a lot of relatively simple calculations than fewer and more complicated ones.

Computers are meant to use numbers, not symbols

Our human brains are better to treat abstract objects than numbers. The operation $x + y$ for example is easier to understand for us than $3.23543 + 43.4321$. One the other hand a numerical addition such as $3.23543 + 43.4321$ corresponds for a computer to only one or few core operation while creating and storing the symbolic representation $x + y$ will require several memory allocations and probably between 10^2 and 10^3 core operations for a decent implementation. This represents an important contrast between our intuitive representation of analytical calculations and the way they are implemented inside a computer.

Any user of a symbolic computation program should be aware of this contrast in order to apprehend correctly performance measurements and good practices using such

11. Artificial intelligence could have a role to play in this matter.

software. As demonstrated in section 2.3.5, writing algorithms using symbols like we write them using numbers is a very bad habit, in particular for CSL. An algorithm as simple as a sum for example must be adapted to the symbolic world to avoid terrible performance costs.

2.3 CSL

This section presents a summary of major CSL features and guiding principles. For more explanations and details on the different mathematical objects of CSL see the user manual [88] and the documentation [89].

2.3.1 Philosophy

The C++ language CSL is developed in C++ based only on the standard library (C++17 standard). The reasons for the language choice are multiple. One of the main considerations is to choose a language that is modern and commonly used in the high energy physics community. This leaves Python and C++ as possible technologies. C++ has been chosen for two main reasons:

- **Performance:** Symbolic expressions cannot benefit from Python's high performance features, i.e. array manipulations. The tree representation presented in the previous section requires a specific implementation, that is much more efficient [12] in C++, a compiled language, than in Python.
- **Structure:** For such a big computer program, constraints coming with a less permissive language like C++ (data types, memory / life-time management) force the code to be better structured. It requires thus more efforts at the first stages of the development but will naturally yield more robust and reliable software.

Independence CSL is built from scratch in order to have a free and independent program. The cost is substantial in terms of developing work compared to the direct use of an existing library such as Mathematica [68], a symbolic manipulation software with its own language (Wolfram language), or SageMath [106] a free python library. These libraries provide very high-level features to manipulate mathematical expressions. It is of course very tempting to use them and it could be perfectly justified. However, if it is feasible to develop only what is required from scratch, for high energy physics purposes, the effort is rewarding. The physics community then have a total control on the program, even its deepest features, while being independent of any bug or update of an external software used as a black box.

Minimal solution for high energy physics With CSL we do not pretend to have a better program than professional computer algebra systems. Its only purpose is to provide a self-made mathematical basis for MARTY and physicists using it. CSL is designed for

12. For similar instructions, C++ can be between 10 and 1000 times faster than Python depending on the problem.

high-energy physics, even if no physics appear at this stage. It must be fast and powerful enough to manipulate expressions from the theory to experimental predictions, but will not be a complete general purpose symbolic manipulation library. For example, while it is a common feature of computer algebra systems, no equation resolution algorithm is provided. In general, features that are not used in perturbative calculations are absent or limited.

2.3.2 C++ basics

This section presents the basics of C++, its philosophy, main features and guideline examples.

History, philosophy

C++ is a compiled language developed during the 80's by Bjarne Stroustrup. The basic idea was to take the C language and add object oriented features and classes. The name was initially *C with classes*. Its first standardization dates from 1998 (C++98 standard), and a new standard came in 2003 (C++03) with minor improvements (mostly bug fixes).

Similarly to the C language, C++ is low level and very efficient. It means that writing C++ will be harder than a higher-level language (especially learning it), the same tasks will require more lines of code but there is a gain in efficiency and control on all parts of the code (provided that the developer has a good knowledge of the language).

While the initial idea of C++ was *C with classes*, it is today way more than that. C++ evolves, in particular since the 2011 standard (C++11) that represent kind of a revolution for the language. Since then, a new standard is released every 3 years, together with better compilers. C++11, C++14, C++17, and now C++20 represent what is called *Modern C++*. Besides new language features and extensions of the standard library, new ways of writing and thinking C++ are developed based on the following principles (not exhaustive):

- **Safety**. Modern C++ helps developers to write safe code without too much effort, in particular avoiding the so called *Segmentation fault* crash, extremely common in C or C++ programs.
- **Performance**. C and C++ are compiled and very efficient languages. New features coming with modern C++ must absolutely not make C++ less efficient. [13]
- **Readability**. New features are often introduced to clarify C++ code and make the intent explicit. Modern implementations are easier to understand, for a developer with a good knowledge of C++, by simply reading it without needing additional comments.

Compile-time vs. run-time

As we said, C++ is a compiled language. This means that before being able to run anything, a **compiler** must read the code and produce an executable file. This is **compile-**

13. C++ is not less efficient than C but requires a bigger knowledge to reach the same performance.

time, translating human-language to computer-language. Then, the executable can be ran, this is **run-time**. Interpreted languages like Python do not have a compile-time, the code is read and executed on the go by an external program.

Compile-time is a blessing for developers. Nowadays, compilers are very efficient and detect numerous errors, often specifying how to correct them before running the first instruction of the program. C++ aims to find as many errors as possible at compile-time. In general, the effect of an error must be detected early after the cause. The longer is the time passed between the bug's cause and its visible consequences, the more difficult debugging is. There are two solutions to address this issue. The first is to catch errors early at run-time by checking regularly that variables are valid. This has run-time overhead (time to check), and is not perfect.

The best solution when it can be implemented is to detect the error at **compile-time**. These errors are the simplest to debug because they are decoupled from any program state, and as the checking takes place at compile-time the execution suffers of no overhead.

C++ compilers usually have good reasons to forbid a developer do to something. Compiler errors and warnings may be annoying at first but they are the very best debugging tools we have. Compiler warnings and errors must be understood in the following way:

- [Warnings] *'You may know what you are doing but I doubt it.'*,
- [Errors] *'This will not work, I cannot let you do it.'*.

This is like having a C++ professor telling the developer what is wrong and very often how to make it work.

Modern compilers are also very good to generate a better machine code than what is written by the developer. This feature has a nice consequence allowing developers in most cases to write clearer code with better abstractions, while generating the same machine code as the equivalent, minimal but less human readable code.

2.3.3 C++ good manners

In this section we introduce two features of C++ and the way in which they must be used, namely memory allocations and the `std::vector`. C++ core guidelines [107] or books such as [108] are dedicated to C++ good manners and provide more insights on how to maximally benefit from C++ features.

Memory allocation

Memory allocation is used in many places in `CSL` and the standard library. One must not avoid it at all costs but be aware that its use must be minimal. If it is not mandatory, there is no reason to allocate a variable on the heap rather that on the stack (static variable). The overheads of heap allocation are the following:

- **Time.** Static allocation is basically changing the stack pointer value, and possibly call a constructor. Heap allocation is asking to the operating system a memory address where a certain quantity of contiguous blocks are available, allocating this

memory and returning the pointer to it. This may take time, especially in programs that allocate a lot of memory.

- **Risk of pointers.** By allocating variables dynamically, one manipulates pointers with all the risks associated with them (null pointer in particular).
- **Life-time management.** Dynamic memory allocation requires deallocation, i.e. managing an object's life-time. This problem is however almost completely solved by C++11 smart pointers that take full responsibility in this matter.

C++ vectors

`std::vector` is the main general purpose container in C++. It is very efficient provided it is used correctly. Pieces of advice to properly use this container are presented through an example in sample code 3 and sample code 4.

Sample code 3: C++ vector good manners, part 1

Here is an example to demonstrate the value of reserve. We overload operators `new` and `delete` in order to count memory allocations / de-allocations. We also create an object that counts the number of times it is copied. The setup is the following:

```
#include <iostream>
#include <vector>
using namespace std;

size_t alloc  = 0;
size_t dealloc = 0;
size_t copy   = 0;

void* operator new(size_t sz) {
    alloc+= 1;
    return malloc(sz);
}
void operator delete(void* ptr) noexcept {
    dealloc+= 1;
    free(ptr);
}
struct A {
    int value;
    A()           : value(0)       {}
    A(int t_value): value(t_value) {}
    A(A const &a) : value(a.value) { ++copy; }
};
```

See the test in sample code 4.

Sample code 4: C++ vector good manners, part 2

Here is the test of the setup presented in sample code 3 (without and with reserve of 1000 elements).

First method, really bad

```
size_t N = 1000;
std::vector<A> vec1;
for (size_t i = 0; i != N; ++i)
    vec1.push_back(A(7));
std::cout << "Copies␣:␣" << copy << std::endl; // >> 2023
std::cout << "Alloc␣␣:␣" << alloc << std::endl; // >> 11
std::cout << "Dealloc:␣" << dealloc << std::endl; // >> 10
```

Second method, using reserve

```
std::vector<A> vec2;
vec2.reserve(N);
for (size_t i = 0; i != N; ++i)
    vec2.push_back(A(7));
std::cout << "Copies␣:␣" << copy << std::endl; // >> 1000
std::cout << "Alloc␣␣:␣" << alloc << std::endl; // >> 1
std::cout << "Dealloc:␣" << dealloc << std::endl; // 0
```

Third method, using reserve and emplace_back

```
std::vector<A> vec3;
vec3.reserve(N);
for (size_t i = 0; i != N; ++i)
    vec3.emplace_back(7);
std::cout << "Copies␣:␣" << copy << std::endl; // >> 0
std::cout << "Alloc␣␣:␣" << alloc << std::endl; // >> 1
std::cout << "Dealloc:␣" << dealloc << std::endl; // 0
```

One can see the importance of reserving the memory space. Using `emplace_back` instead of `push_back` allows the vector to create the object in place (giving arguments of construction directly to the function) instead of creating it and copying it into the container.

Note All three counters must be set to 0 before each test, even the first one, as starting a C++ program may imply memory (de-)allocation.

2.3.4 The Expr type

The most important things to know about `Expr` are summarized in the following statements:

- It is the type representing a mathematical expression in CSL.
- There is no other general purpose expression type.
- It is a pointer-type, in particular a `std::shared_ptr`. An expression may be shared between others, and its destruction does not depend on the user or the developer but on the standard library implementation.

`Expr` is a pointer-type because as we said, the underlying expression may be anything. Anything means any type, and in C++ it is not possible to handle different types in the same container. It is however possible to manipulate pointers of the same type, but pointing to objects of different types. This is "polymorphism".

A very simplified version of the inheritance hierarchy in CSL is presented in figure 2.2. Mathematical expressions all inherit from an abstract base class, Abstract, and specialize step by step up to the final objects one may encounter in expressions. An important feature is that no intermediate abstract class may be built. This prevents any user to build objects that are not specialized enough to have a meaning mathematically.

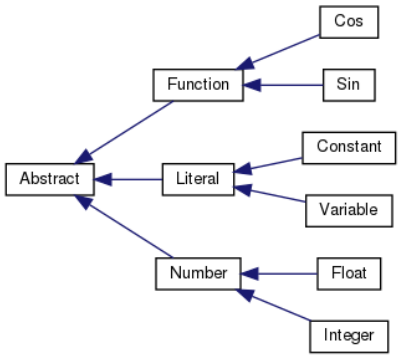


Figure 2.2 – Simplified inheritance hierarchy of mathematical expressions in CSL. One can find the full diagram in the documentation [89].

The Expr class is a `std::shared_ptr<Abstract>` with more interface. This pointer does not point to a pure Abstract object, but to one specialization that is a particular mathematical expression.

The `Expr` type has few subtleties in its use for the user. As each copy of a shared pointer increases its reference count, it means that more operations are done than a simple copy of pointer. `Expr` object are then always passed by reference in functions. This reference is const if the function does not modify the expression (`Expr const&`), non-const otherwise (`Expr &`). Examples are presented in sample code 6. Arithmetic and comparison operators are defined with `Expr` object, as shown in sample code 7.

Sample code 5: Basics on Expr

Using `std::shared_ptr<Abstract>` features of `Expr`, using a dot

```
Expr expr = functionReturningExpr();
// Displaying the reference count of the shared ptr
cout << expr.use_count() << endl;
// Getting the raw pointer from the shared ptr
Abstract *raw_ptr = expr.get();
```

Using features of the underlying `CSL` object, using the arrow, de-referencing, or interface functions

```
Expr expr = functionReturningExpr();
// Directly
cout << expr->getType() << endl; // may display "Cos" for ex
cout << (*expr).getType() << endl; // may display "Cos" for ex
// Using an interface function
cout << GetType(expr) << endl; // may display "Cos" for ex
```

Note When calling `expr->getType()`, C++ gets to know the exact type of the pointed expression and calls the corresponding virtual function of that type.

Sample code 6: Functions with Expr objects

Full example showing how to pass and return `Expr`

```
Expr func( // Replaces B by A+1 and returns the old value of B
    Expr const &A,
    Expr &B
    )
{
    Expr B_copy = B;
    B = A + 1;

    return B_copy;
}
```

Sample code 7: Operators with Expr

Arithmetic and comparison operators work on `Expr` as expected:

```
Expr a = constant_s("a");
Expr x = variable_s("x");
Expr y = 4 * (a + x)/(a - x);
cout << (a == y) << endl; // >> 0
cout << (a != y) << endl; // >> 1
cout << (a <= y) << endl; // >> 1
```

See also Section 2.2.4 for the `operator<()` acting on expressions (simplicity ordering).

2.3.5 CSL good manners

This section is dedicated to a user knowing already the basics of C++ and CSL. In the following some pieces of advice to use CSL appropriately are presented. CSL provides its own objects that use memory allocations and de-allocations. In order to have dynamic and arbitrary mathematical expressions, a lot of memory management must be done. At each intermediate step of calculation, expressions are created on the heap and destroyed (such as during the canonicalization procedure discussed in section 2.2.3). Manipulating a sum of ten CSL expressions for example (copying, modifying) will be **much slower** than ten integers in a `std::vector`. **CSL expressions are not just containers.**

The Expr type

While `Expr` is still heavier to copy than a raw pointer like `int*`, it is very simple and in particular does not imply the copy of the underlying expression. In the following example, the expression $2 + \exp(3)$ is stored once in memory, `a` and `b` pointing to it:

```
Expr a = 2 + exp_s(3);
Expr b = a; // Just pointer copy
```

As `Expr` is a pointer, it can be invalid. There is nothing fundamentally wrong about writing something like

```
Expr x = nullptr;
Expr y;
```

It is possible to do it, in particular to express the fact that the expression may be invalid. But **in no circumstance a null `Expr` should enter a CSL expression**. CSL takes for granted the validity of `Expr` objects. No test is done to check if pointers are valid because they are meant to be, and it would represent an important run-time overhead. Giving a null `Expr` to CSL will certainly cause a bug but the exact behavior is not defined. Because of the fact that `Expr` is a pointer type, **an un-initialized Expr is null**. To express that an expression is invalid, one should consider to use `std::optional<Expr>` or the undefined CSL constant `CSL_UNDEF` instead.

Avoidable allocations

Sums and products are dynamic containers, i.e. can store an arbitrary number of arguments. Therefore, they rely on memory allocation and must be used carefully. There are several ways to build multi-functions like sums in CSL that are not equivalent in terms of performance. In particular, writing symbolic algorithms in the same way as numerical instructions has often a high performance cost. This is summarized in sample code 8.

Sample code 8: Good habits with symbolic manipulations

Product of few elements, a bad solution:

```
std::cout << x * x * x * x * x << std::endl;
// ~40 allocations/deallocations of pure Expr
// >> x^5
```

Product of few elements, a good solution:

```
std::cout << prod_s({x, x, x, x, x}) << std::endl;
// ~15 allocations/deallocations of pure Expr
// >> x^5
```

The chained operators correspond in this case to do $x \times (x \times (x \times (x \times x)))$ which requires many more intermediate steps than the second solution.

In the following we consider a function `f()` returning an `Expr`.
Sum of many elements, a very bad solution:

```
Expr sum = 0;
for (int i = 0; i = 1000; ++i) {
    sum += f(); // Performs simplifications 1000 times
}
```

Sum of many elements, a good solution:

```
std::vector<Expr> terms(1000);
for (int i = 0; i = 1000; ++i) {
    terms[i] = expr; // Pointer copy, no symolic machinery
}
Expr sum = sum_s(terms); // Simplify only once, at the end
```

The first solution that invokes symbolic summations at each iteration can have a disastrous impact on performance for large expressions and must always be avoided.

Heavy interface functions

Heavy modifiers, interface functions or algorithms (i.e. that (re-)allocate many expressions) should be used only when they are necessary. In particular:

- `DeepCopy()` re-allocates the entire expression ensuring that no object is shared between the old and new expressions. If repeated this may take time in the program.
- `DeepRefresh()` performs a `DeepCopy()` but also applies simplification rules to ensure the canonical form of the resulting expression.
- Repeated calls to modifier functions in general must be avoided when possible such as `DeepExpand()`, `DeepFactor()`, `Replace()`, ...
- The `ForEachNode()` algorithm should be preferred in general to `Transform()`, because it does not refresh the expression on the go. If the result does not need to be refreshed the second option has an unnecessary run-time overhead.

2.4 CSL as a module of MARTY

MARTY is a very general code for high-energy physics. From model building to theoretical calculations at one-loop, many calculations can be performed by MARTY for BSM. Amplitudes, squared amplitudes and Wilson coefficients may be calculated in full generality in a very large variety of BSM scenarios. The idea behind that code is to be able to build a new BSM model easily and perform a detailed phenomenological study within a few lines of C++ code and in very little time, greatly accelerating the research Beyond the Standard Model.

Such a complex code must encapsulate as many features as possible and provide a simple user interface. Most of the features can be accessed through a single function call and parameters are simplified as much as possible in order to ask to users the minimal quantity of information. A sketch of how MARTY works internally is presented in figure 2.3. One may see in particular that there is no direct communication between the user and the calculation modules of MARTY. This is the consequence of the fact that calculations are fully automated. All CSL capabilities can still be used to modify the results as they are symbolic expressions stored in Expr variables. [14]

A standard MARTY program always goes through the same steps.

- **Model loading.** The model can be ready to be used or need some calculation steps to be derived by MARTY.
- **Setting of options.** Before doing calculations, one may want to change some options to customize the output.
- **Calculations.** Once the model is loaded and options are set, one can launch the calculation(s) that MARTY must perform. Details are given in chapter 6.
- **Library generation.** After calculating a theoretical quantity analytically, MARTY generates the corresponding C++ code to evaluate the results numerically depending on the model parameters. This procedure is also automated.

All the steps detailed above are typically very simple to implement and a MARTY program rarely takes more than a hundred lines of code when written correctly.

Once the numerical C++ code is generated, the resulting values can be used in a phenomenological code to scan the model parameters, detect interesting scenarios with respect to the SM, perform comparisons with experimental data, etc. The ultimate tool would be a direct interface between MARTY and such codes. It is already on the way for SuperIso [90–92] in flavor physics, and SuperIso Relic [94–96] in Dark Matter. With such interfaces, a complete phenomenological analysis for a BSM model can be automated at tree-level or one-loop by providing only a BSM Lagrangian.

14. Results are rarely composed only of symbolic expressions because there is more information to return to the user. One may however always get to symbolic expressions and use CSL to modify them.

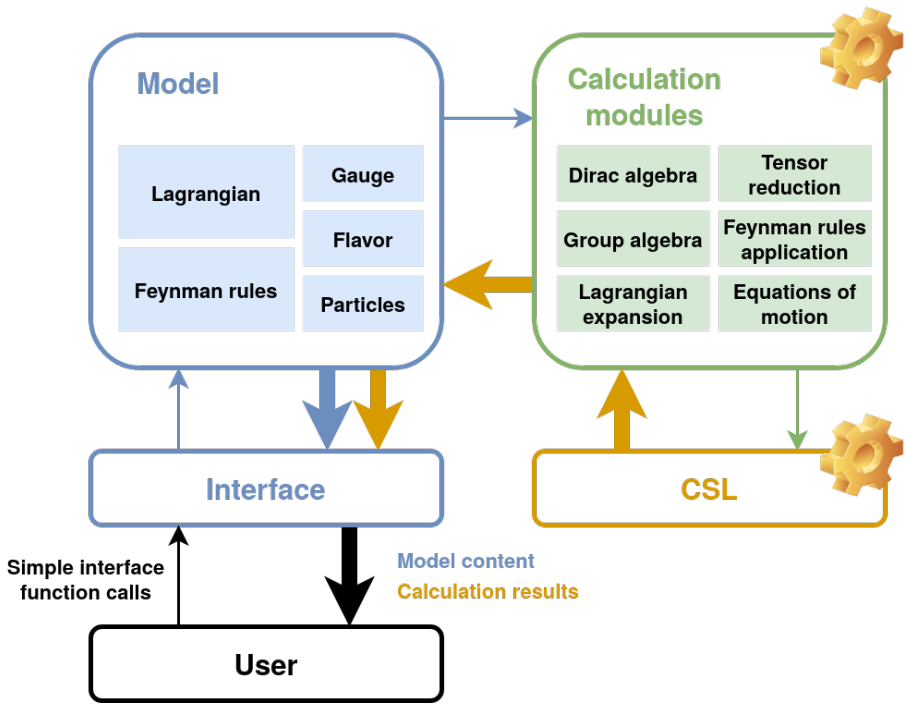


Figure 2.3 – Sketch of MARTY's basic principle. The user communicates (in C++) with the high-energy physics model to get / modify its content, and to perform calculations. These calculations are done internally by MARTY (using CSL) and are completely separated from the user interface.

2.5 GRAFED

GRAFED is a **G**enerating and **R**endering **A**pplication for **FE**ynman **D**iagrams, a Linux/Mac desktop application to generate and create any type of Feynman diagrams. It is a module of MARTY but can be used as standalone to create publication-quality diagrams. In particular, all the diagrams presented in this thesis have been generated automatically or created with GRAFED.

GRAFED has been developed with the non-commercial version of Qt [109], a C++ framework to create desktop applications. We developed GRAFED for three main purposes:

- **Design.** We wanted MARTY to have the ability to display nice Feynman diagrams on screen.
- **Validation.** It is important when developing or using the code to know quickly what diagrams contribute to a calculation in order to understand what MARTY calculates.
- **Diagram creation.** GRAFED provides a simple and comprehensive interface allowing users to edit, create and save Feynman diagrams. They can then be exported as `.png` image files or LaTeX source code for features supported by the TikZ-Feynman package [110]. Examples of edition are shown in figure 2.4.

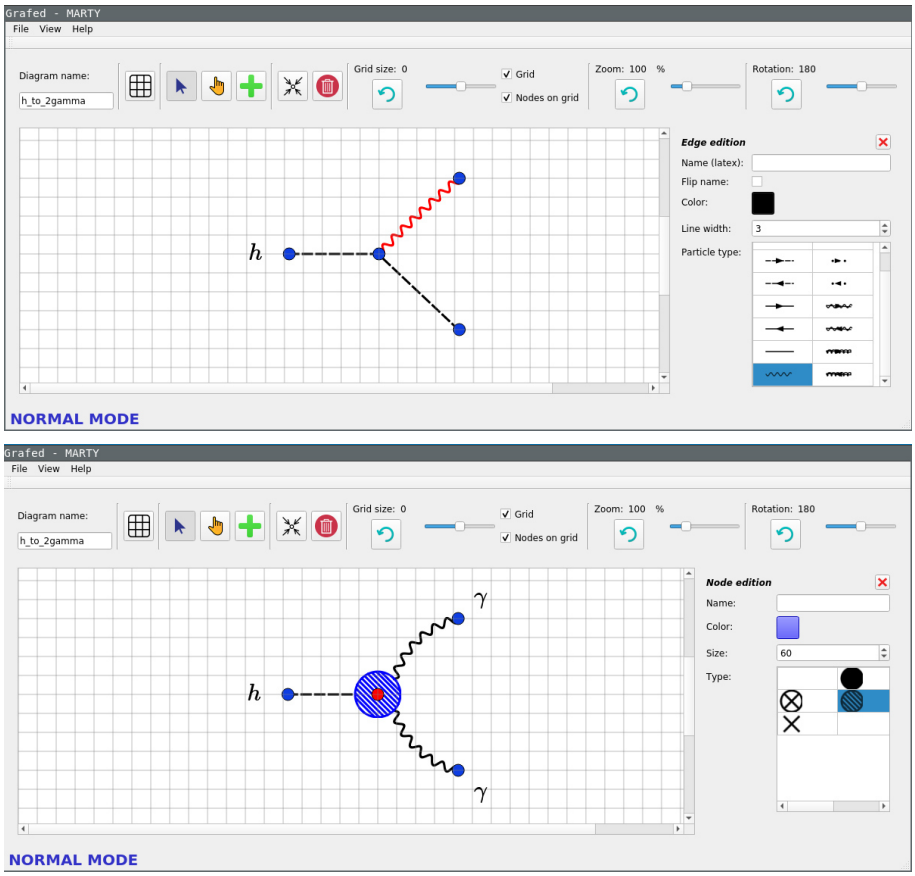


Figure 2.4 – Screen shots of the GRAFED application on Linux while creating a $h \to \gamma\gamma$ diagram. The two screen-shots present two different steps of edition with the edge-specific edition panel (top) and the node-specific panel (bottom).

The algorithm

Automated generation of Feynman diagrams requires to derive their layout, i.e. the vertex positions. This is not an easy task as there exists many different topologies and the purpose is to have an algorithm that generates nice diagrams which correspond to standard conventions in high energy physics. Another possibility could be to work with topologies, for example defining one layout for each different topology. We decided however to implement one unique and general algorithm that matches better MARTY's design principles.

For a given graph $G = (V, E)$ with vertices V and edges E, we define a layout of G as a set of vertex positions in the 2D plane namely

$$\ell_G \equiv \{(x_v, y_v)\ \forall v \in V\}\,. \tag{2.13}$$

We associate to each layout ℓ_G of G a unique energy $E(\ell_G)$ that can be defined in general as

$$E(\ell_G) \equiv \sum_i \alpha_i \cdot e_i(\ell_G), \tag{2.14}$$

with $e_i(\ell_G)$ fundamental energy costs of the layout ℓ_G and α_i layout-independent coefficients. The different $e_i(\ell_g)$ have precise definitions. For example we defined an energy cost for edge lengths that must be close to a given distance d_0:

$$e_{d_0}(\ell_G) \equiv \sum_{(v_i,v_j)\in E} \left(d_0^2 - d_{ij}^2\right), \tag{2.15}$$

where

$$d_{ij}^2 \equiv (x_{v_i} - x_{v_j})^2 + (y_{v_i} - y_{v_j})^2. \tag{2.16}$$

We defined several fundamental energies for distances, relative and absolute edge angles around the vertices, crossing edges, ... Once the energy function E has been fully defined, a minimization algorithm can be used to find the best layout $\bar{\ell}_G$ minimizing the energy given by

$$\bar{\ell}_G = \text{argmin}_{\ell_G}(E(\ell_G)), \tag{2.17}$$

for all graphs G in a given process. [15] Following the definition of ℓ_G in equation 2.13 one can see that the minimization is done on $2|V|$ real parameters with $|V|$ the number of vertices in G. [16] The energy functions e_i together with values of the coefficients α_i have to be tuned empirically to generate nice layouts. This algorithm has the nice feature of being unique and not limited to any topology.

Most of the diagrams we show in this thesis have be created with GRAFED for pedagogical purposes. There are nevertheless diagrams that are direct outputs of the layout generation algorithm presented above such as those in figure 7.1.

15. The algorithm must only be ran once for each topology, the same layout is then used for similar diagrams that simply use different particles.

16. To avoid local minima with crossing edges, the minimization is first promoted in a 3D-space with $3|V|$ parameters to allow the graph to expand and is then projected on a relevant 2D-plane to terminate the minimization.

CHAPTER 3

Quantum fields

3.1 Introduction

A brief overview of MARTY has been presented in the previous chapter, discussing in particular the challenge that symbolic manipulations represent in general and the way in which this project fits into the context of BSM phenomenology. We are now interested in the way MARTY can automate the theoretical calculations from general BSM Lagrangians. In order to do so, the first step is to introduce the main ingredient of elementary particle physics: The particles. Then, chapter 4 will present how (B)SM models are described in MARTY. Finally, after a discussion on group theory implementations in chapter 5, automated calculations for BSM phenomenology will be addressed in chapter 6.

Quantum fields can represent both the abstract concept of particle e.g. the electron in the SM and objects entering mathematical expressions such as $e_\alpha(X)$ or $e^\dagger_\beta(Y)$. We present how these features are implemented in MARTY through a description of the corresponding user interface. More details on particle types and properties can be found in the user manual [86] or the documentation [87]. If needed the CSL user manual [88] and documentation [89] present in details tensor fields and their properties in general.

The `QuantumField` object inherits from `TensorFieldElement` in CSL. A tensor field is a tensor with a space-time point:

$$A_{\mu\nu} \to A_{\mu\nu}(X), \tag{3.1}$$

and a quantum field is a tensor field with more properties. Figure 3.1 presents a sketch on how quantum fields are represented inside MARTY following the same principle as in CSL. The abstract field, a `QuantumFieldParent`, has one unique copy in the entire program and contains all the particle's intrinsic properties (spin, mass, gauge representation, ...) whereas `QuantumField` objects entering expressions are multiple and carry only their extrinsic properties (indices, space-time point) and a reference to their parent to access their fundamental features.

As in the case of simple tensors, the parent object is not directly the quantum parent, but in this case a `Particle`. It is a shared pointer to `QuantumFieldParent`, ensuring a well-managed life-time for the parent. MARTY's user interface always takes `Particle` objects, and the `QuantumFieldParent` interface can be accessed simply using `->` on a `Particle`.

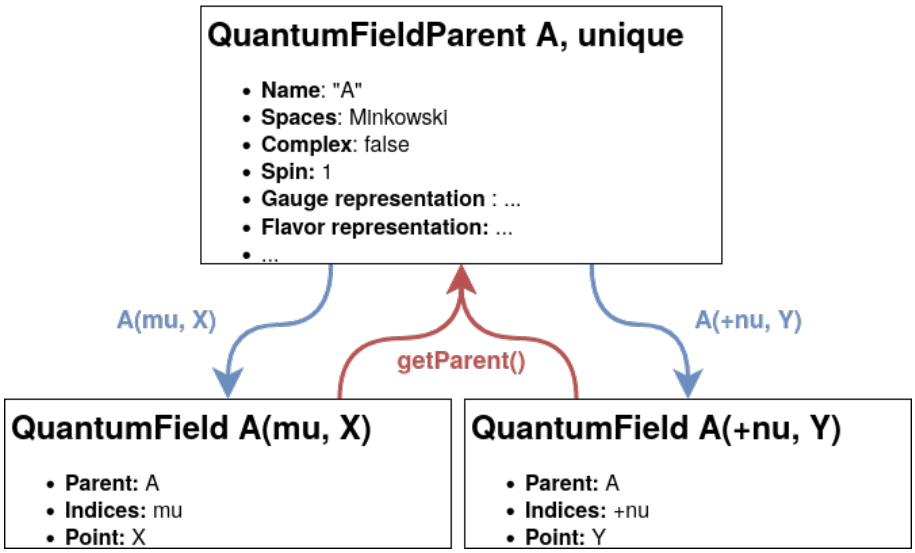


Figure 3.1 – Working principle of quantum fields in `MARTY`. The parent (`QuantumFieldParent`) is unique in the program, contains all its intrinsic properties, and can generate symbolic expressions (`QuantumField`) given some indices and a space-time point. The + sign indicates that an index is up.

3.2 Different types of quantum fields

All possible quantum fields in `MARTY` inherit from `QuantumFieldParent` as shown in figure 3.2. The base class is not constructible, one manipulates a (shared) pointer to this base class. This shared pointer is encapsulated in a `Particle` object and can then reference a Dirac fermion, a vector boson, etc. This is exactly equivalent to the `Expr` interface pointing to a `Abstract` object presented in section 2.3.4.

A particle in `MARTY` is one specialization of `QuantumFieldParent`: [1]

- `ScalarBoson`: Trivial representation of the Lorentz group.
- `WeylFermion`: Chiral 4-component fermion. They are always projected on left or right chiralities in amplitudes. They may be paired to form a Dirac fermion.
- `DiracFermion`: 4-component fermion. Can represent either a Dirac particle or a Majorana particle if the fermion is self-conjugate.
- `VectorBoson`: Spin 1 particle A_μ, associated with a field strength $F_{\mu\nu}$. A vector boson can also have associated ghost and Goldstone bosons.
- `GaugeBoson`: Specialization of `VectorBoson` that keeps a reference to the group of which it is the gauge boson. In a non abelian gauged group (different from $U(1)$) a gauge boson has a predefined `GhostBoson`.
- `FieldStrength`: Field strength object for a vector boson (gauge or not) $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$. This object is intentionally does not contain covariant derivative terms that are treated independently.

1. `BaseVectorBoson` is not a specialization that can be created, only a common interface that is used for `FieldStrength` and `VectorBoson`.

- `GhostBoson`: Ghost bosons defined in non abelian gauged groups. They are non-physical anti-commuting bosonic fields introduced to quantize correctly the theory and appear only at the one-loop level in diagrams.[2] They are linked with their `GaugeBoson` through gauge fixing.
- `GoldstoneBoson`: Goldstone bosons can be defined to link a scalar boson of the theory as the Goldstone of a vector boson in order to have gauge invariant quantities when using massive vector bosons.

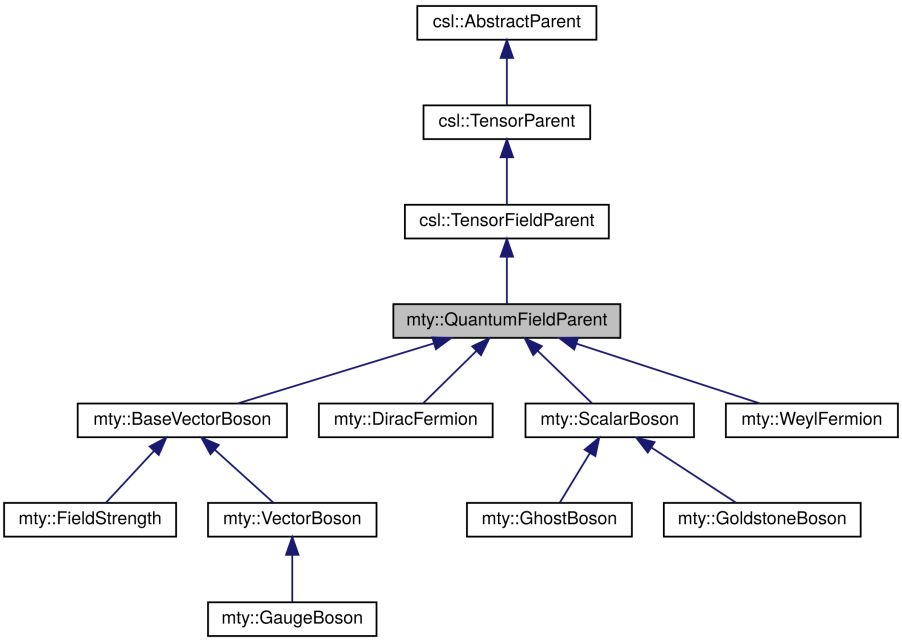


Figure 3.2 – Inheritance tree for the `QuantumFieldParent` object. It first inherits from `CSL` tensors, and then each particle type is a different specialization of `QuantumFieldParent`.

3.2.1 Particle types

Some particles may be defined automatically by `MARTY`, such as the gauge bosons and ghosts, when defining the gauge group of the theory. For most phenomenological purposes one has to define a custom particle content,[3] using built-in functions that create all the required particles described above. All particles are built in the same way. The particle name, the model in which it lives and additional arguments specific to the particle must be given to the relevant builder function. In all sample codes presented in the following, `model` is assumed to be a valid `Model` object, whose gauge is already initialized (see the model building chapter of the user manual [86] for more details).

2. They can also appear in external legs replacing gauge bosons in order to respect gauge invariance while using a simple polarization sum for the vector.

3. Only in a model building context. When using a built-in model, there is no need to define anything before using the model.

3.2.2 Fermions

Three types of spin 1/2 particles can be built in MARTY, namely Weyl, Dirac, and Majorana fermions. Majorana fermions do not have their own builder function, as they do not have their own class either. To create a Majorana particle one must first build a Dirac fermion, and then specify that it is self-conjugate. Doing so will enable non-trivial contractions in diagrams such as $\langle\psi\psi\rangle$ and $\left\langle\bar{\psi}\bar{\psi}\right\rangle$, whereas for Dirac fermions only $\left\langle\psi\bar{\psi}\right\rangle$ and $\left\langle\bar{\psi}\psi\right\rangle$ do not vanish. Sample code 9 presents a summary on how to build fermions.

Sample code 9: Creating fermions

Dirac fermion

```
Particle e = diracfermion_s("e", model);
```

Weyl fermion

```
Particle muL = weylfermion_s("mu_L␣;␣\\mu_L", model, Chirality::Left);
Particle muR = weylfermion_s("mu_R␣;␣\\mu_R", model, Chirality::Right);
```

Majorana fermion

```
Particle maj = diracfermion_s("M", model);
maj->setSelfConjugate(true);
```

Note One can give different names for the particle in the program (its identifier) and in diagrams (displayed as Latex code on the screen) separated by a semi-colon. Spaces around the character ; are ignored.

3.2.3 Vectors

Spin 1 particles are often built by default in MARTY, as they are in general gauge bosons that are created automatically when the gauge group is defined. Most of high-energy physics models e.g. the SM or even BSM scenarios do not have additional spin 1 particles. It is still possible in MARTY to create other spin 1 fields, that are not gauge bosons, as shown in sample code 10.

Sample code 10: Creating vector bosons

Vector boson

```
Particle A = vectorboson_s("A", model);
```

Field strength

```
Particle F_A = A->getFieldStrength();
```

Note Vector bosons are real by default. To create a complex vector one has to use `A->setSelfConjugate(false);`.

3.2.4 Scalars

Spin 0 particles are very simple to create as they have no specific property due to their spin. The procedure to create a scalar boson is presented in sample code 11.

Sample code 11: Creating scalars

```
Particle phi = scalarboson_s("phi ; \\phi", model);
```

Ghosts and Goldstone bosons can be created explicitly as shown in sample code 12, but in general will be handled automatically by MARTY during the model construction. As those particles are tied to a given VectorBoson, there is no need to define any property. Only the associated vector and an optional name have to be given.

Sample code 12: Creating ghosts and Golstones

Considering a vector boson A as in sample code 10.

Ghost

```
Particle ghost_A = ghostboson_s("c", A);
// Or with name chosen by marty
Particle ghost_A = ghostboson_s(A);
```

Goldstone

```
Particle goldstone_A = goldstoneboson_s("c", A);
// Or with name chosen by marty
Particle goldstone_A = goldstoneboson_s(A);
```

3.3 Using and modifying a Particle

Once a particle has been built following prescriptions of section 3.2, the interface is almost always identical for all the different types of particles. This section presents how to perform basic manipulations on particles. The documentation of Particle and QuantumFieldParent [87] contains all the interface methods presented in this section.

3.3.1 Obtaining particles from a model

Gauge-related particles

We saw in the previous section how to create new particles. Before going further, it is necessary to explain how to get particles that MARTY creates on its own. Figure 3.3 presents the objects that are automatically created with gauged groups, including naming conventions, in MARTY. All default names can be changed by the user. It is however important to know the initial convention to be able to access all objects created automatically.

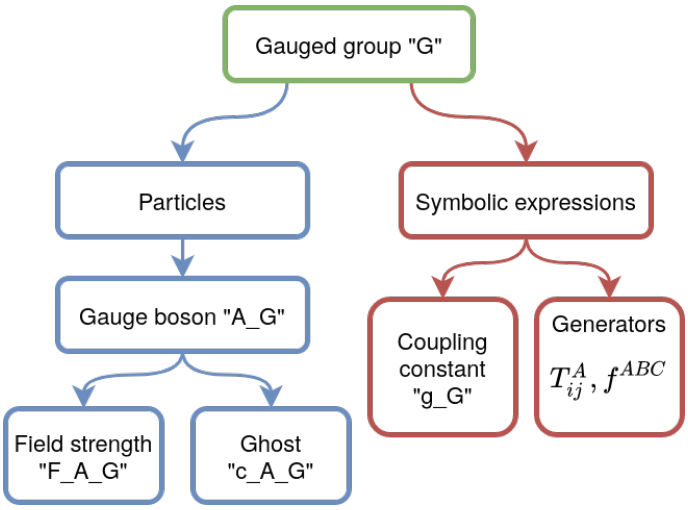


Figure 3.3 – Sketch of the main objects a MARTY gauged group creates automatically. No ghost particle or generators are created for the abelian $U(1)$ group.

The way to get different kinds of particles from a model is demonstrated in sample code 13.

Sample code 13: Getting a particle from a model

```
Particle e_L = model.getParticle("e_L"); // Weyl fermion
Particle A = model.getParticle("A_G"); // Vector boson of group "G"
Particle F_A = model.getParticle("F_A_G"); // Field strength
// Or get the FieldStrength from the VectorBoson
// Particle F_A = A->getFieldStrength();
Particle c_A = model.getParticle("c_A_G"); // Ghost boson
```

Dirac fermion embedding

When creating a Dirac fermion, MARTY automatically creates its left-handed and right-handed parts. The three generated particles can talk to each other, and in particular a user can navigate in the triangle they define through simple function calls. This is presented in figure 3.4 and concrete examples in a MARTY program are shown in sample code 14.

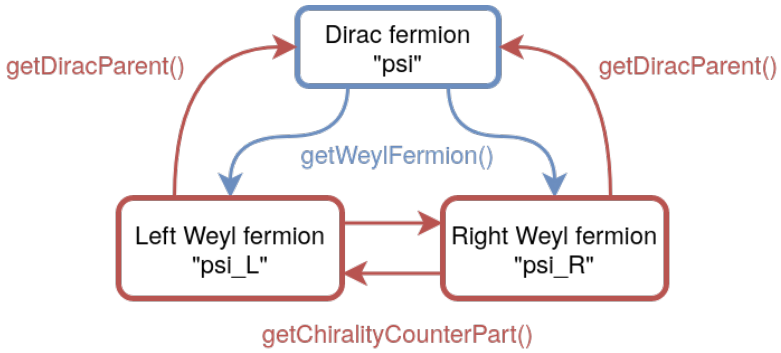


Figure 3.4 – Sketch of the relations between the different parts of a Dirac fermion embedding $\psi \equiv \psi_L \oplus \psi_R$. The `getWeylFermion()` function takes as argument a `Chirality` value.

Sample code 14: Dirac fermion embedding

Creating a Dirac fermion

```
Particle psi = diracfermion_s("psi", model);
```

Navigating in the triangle

```
Particle psi_L = psi->getWeylFermion(Chirality::Left);
Particle psi_R = psi_L->getChiralityCounterPart();
Particle other_psi = psi_R->getDiracParent();
```

3.3.2 Simple particle properties

Simple particle properties do not include gauge and flavor representations, treated separately in section 3.3.3. The main properties of quantum fields are presented in table 3.1.

Property	Type	Getter	Setter
Name	`string`	`getName()`	`setName()`
Latex name	`string`	`getLatexName()`	`setLatexName()`
Spin dimension	`int`	`getSpinDimension()`	
Mass	`Expr`	`getMass()`	`setMass()`
Width	`Expr`	`getWidth()`	`setWidth()`
Self-conjugation	`bool`	`isSelfConjugate()`	`setSelfConjugate()`
Physicality	`bool`	`isPhysical()`	`setPhysical()`

Table 3.1 – List of properties for quantum fields with the type, getter and setter functions when relevant. For setter functions, one must give an argument of the specified type.

Name

Names of particles have two different purposes apart from identifying their owners in expressions. First, they must uniquely define particles to allow a user to identify a particle in a model. This name has to be short and simple to make the program readable. Another use of names is in Feynman diagrams, where we prefer in general to see ν_{μ_L} rather than num. The problem is that latex expressions are complicated, in this case one should write `"\\nu_{\\mu_L}"` each time referring to the muon neutrino.

This is why regular and latex names are separated in `MARTY`. If not specified, the latex name will be identical to the regular one. To avoid multiple calls (`setName()` and `setLatexName()`), it is possible when creating a particle to give both names in the same string literal separated by a `;` (spaces around it are ignored). For example, one can define a neutrino with

```
Particle num = weylfermion_s("num␣;␣\\nu_{\\mu_L}", Chirality::Left);
```

For identification purposes `"num"` will then have to be used, but Feynman diagrams will display ν_{μ_L}.

Spin

The spin can of course not be changed, as it would require to change the particle type. The value of spin can be accessed at anytime with the `getSpinDimension()` function. As this function returns an integer, one does not obtain the spin but the spin dimension

$$d = 2j + 1, \tag{3.2}$$

for a particle of spin j. Examples of property usage are presented in sample code 15.

Self-conjugation

The self-conjugation property is in general the possibility for the field to contract with itself, i.e. without being complex conjugated. For a self-conjugate field, contractions such as $\langle\phi\phi\rangle$ and $\left\langle\phi^\dagger\phi^\dagger\right\rangle$ are enabled in diagrams whereas only $\left\langle\phi\phi^\dagger\right\rangle$ and $\left\langle\phi^\dagger\phi\right\rangle$ are for other fields, as shown in figure 3.5. This is a general statement. For integer spin particles it is simpler because the self-conjugate property implies that fields are real

$$\begin{aligned} \phi^\dagger &= \phi \text{ for scalars bosons,} \\ A_\mu^\dagger &= A_\mu \text{ for vector bosons.} \end{aligned} \tag{3.3}$$

There is in this case only one possible contraction.

For fermions one must be more careful because the self-conjugate Dirac field is not real. It has 2 degrees of freedom instead of 4 but this is not equivalent to $\psi^\dagger = \psi$, at least not in all realizations. For spin 1/2 particles, the self-conjugate property reads

$$\psi^c \equiv C\bar{\psi}^T = \psi, \tag{3.4}$$

with the conjugation matrix

$$C \equiv -i\gamma^0\gamma^2, \tag{3.5}$$

in the Weyl realization for γ-matrices that is the one used in `MARTY`. As the relation is not as simple as $\psi^\dagger = \psi$, we keep both ψ and $\psi^\dagger$ in the Lagrangian, and the four possible contractions presented in figure 3.5 are possible replacing ϕ by ψ. Examples of property usage are presented in sample code 15.

(a) Standard contractions **(b)** Self-conjugate contractions

Figure 3.5 – Possible field contractions. For non self-conjugate particles only contractions (a) do not vanish. For self-conjugate particles, all four contractions are non-zero. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

Mass and Width

A particle's mass and width arise in propagators when calculating an amplitude. They may be set to any expression, not only constants or zero. One may for example give a mass

$$M_G = \sqrt{\xi} M_A \tag{3.6}$$

to a Goldstone boson related to a vector A, with gauge fixing parameter ξ.

The mass M and width Γ of a particle appear in propagator denominators

$$\frac{1}{p^2 - M^2 + iM\Gamma}. \tag{3.7}$$

The mass also appears in numerators for fermions and vector bosons but we will not detail theses dependencies here. The width is by default zero for all particles to lighten the symbolic results. Depending on the kinematics, one should add non trivial widths for particles produced around their resonance peak e.g. the Standard Model Z boson at 90 GeV. Examples of property usage are presented in sample code 15.

Sample code 15: Quantum fields properties

Building particles for the example

```
Particle W = vectorboson_s("W", model);
Particle c_W = ghostboson_s(W);
Particle psi = diracfermion_s("psi", model);
```

The basics

```
cout << c_W->getSpinDimension() << "␣" << psi->getSpinDimension()
<< "␣" << W->getSpinDimension();
// >> 1 2 3
```

Mass and width

```
Expr xi = constant_s("xi");
Expr M_W = constant_s("M_W");
Expr G_W = constant_s("G_W");
W->setMass(M_W);
W->setWidth(G_W);
c_W->setMass(sqrt_s(xi) * W->getMass()); // m_{c_W} = sqrt(xi) M_W
cout << c_W->getMass() << endl;
// >> xi^(1/2)*M_W
```

3.3.3 Gauge and Flavor representations

Gauge and flavor representations are a central part in MARTY's model building features, users should thus have a good knowledge of this aspect.

Gauge representations

MARTY can handle all irreducible representations of semi-simple Lie groups. These representations are uniquely defined by Dynkin labels [111] that are positive integers. The number of labels required to define the representations of a given group corresponds to the rank ℓ of the algebra. Table 3.2 presents the link between gauged groups and their corresponding algebra. Dynkin labels for common representations used in high-energy physics are written in table 3.3.

The abelian $U(1)$ group must be treated independently as there is no Dynkin label in this case. $U(1)$ representations are defined by (possibly fractional) charges. In order to unify notations when doing model building for $U(1)$ or non abelian gauged groups, a fractional charge is treated as a pair of Dynkin labels, one for the numerator and one for the denominator. For example in the SM, the electron has Dynkin labels $(-1, 1)$ for the electromagnetic $U(1)$ gauge whereas the up quark has labels $(2, 3)$. More details on irreducible representations are given in chapter 5.

By default, all representations are trivial[4]. Sending the gauge group name and the

4. Trivial representations are dimension 1 for non abelian gauged group and representations of charge 0 for $U(1)$ groups.

Group	Algebra	Rank
$SU(N)$	A_ℓ	$\ell = N - 1$
$SO(2N+1)$	B_ℓ	$\ell = N$
$Sp(2N)$	C_ℓ	$\ell = N$
$SO(2N)$	D_ℓ	$\ell = N$
E_6	E_6	$\ell = 6$
E_7	E_7	$\ell = 7$
E_8	E_8	$\ell = 8$
F_4	F_4	$\ell = 4$
G_2	G_2	$\ell = 2$

Table 3.2 – Link between semi-simple Lie groups and their algebras. The rank ℓ of the Lie algebra corresponds to the number of Dynkin labels defining uniquely irreducible representations.

Group	Algebra	Dynkin labels	Dimension
$SU(2)$	A_1	(1)	2 (doublet)
$SU(2)$	A_1	(2)	3 (triplet)
$SU(3)$	A_2	$(1,0)$	3 (triplet)
$SU(3)$	A_2	$(0,1)$	$\bar{3}$ (anti-triplet)
$SU(3)$	A_2	$(1,1)$	8 (octet)
$SU(3)$	A_2	$(2,0)$	6 (sextet)
$SO(4)$	D_2	$(1,0)$	2 (left spinor)
$SO(4)$	D_2	$(0,1)$	$\bar{2}$ (right spinor)
$SO(4)$	D_2	$(1,1)$	4 (vector)

Table 3.3 – Common representations in high energy physics, with their group, algebra, Dynkin labels and dimensions. Trivial representations (dimension 1) have always labels equal to zero.

Dynkin labels between curly braces to `setGroupRep()` will automatically change the particle representation. An example in a $SU(2)_L \times U(1)_Y$ gauge is given in sample code 16.

Sample code 16: Setting gauge representations

Considering a gauge composed of one $SU(2)$ group named `"L"` and one $U(1)$ group named `"Y"`, one can set the representation of $e_L : (2, -1)$ and $u_R : (1, 2/3)$ writing

```
Particle e_L = weylfermion_s("e_L", model, Chirality::Left);
e_L->setGroupRep("L", 1); // curly braces not required for one value
e_L->setGroupRep("Y", -1); // denomimator = 1 omitted
```

for the electron and

```
Particle u_R = weylfermion_s("u_R", model, Chirality::Right);
u_R->setGroupRep("Y", {2, 3}); // fractional charge 2/3,
                               // more than one value -> curly braces
```

for the quark.

Note This example does not exactly correspond to SM conventions for simplicity.

Note When the denominator of a fractional charge is 1 it may be omitted when giving the representation.

See also Chapter 5 for more details on representations.

Flavor representations

A flavor symmetry indicates that several particles have the same interactions in a given theory, without being related to a gauge symmetry i.e. a fundamental force of nature. Flavor symmetries, like gauge symmetries, can hold at high energies and be broken at lower energies. In the Standard Model the three fermion generations are described with a flavor symmetry. It is broken at low energies and the masses of e.g. the electron, muon and tau particles [5] are different. Flavor representations in `MARTY` are for now limited to two types:

- **Complex flavors.** Mixes N complex fields that are considered as a fundamental representation of a $SU(N)$ flavor group. The SM flavor symmetry is a $SU(3)$ group.
- **Real flavors.** Mixes N real fields that are considered as a vector representation of a $SO(N)$ flavor group.

The flavor machinery could be extended in the future if needed. We may consider for example different representations of a same flavor group, with tensors mixing different representations. However, these cases are very rare and there is no support in `MARTY` to do it automatically. Work-arounds are nevertheless already possible as interactions

5. In the SM, the electron, muon and tau particles are exactly identical at high energies (typically above 1 TeV) but the symmetry is broken by the Higgs potential and they acquire different masses.

can be user-defined in a very general way, introducing custom tensors coupling between different fields, as demonstrated in the next section. Sample code 17 shows how to define flavor representations for particles in a MARTY model.

Sample code 17: Setting flavor representations

In the following we consider a complex flavor $SU(3)$ named "C" and one real $SO(4)$ named "R".

Creating the fields

```
// Real field for the real SO(4) flavor
Particle phi = scalarboson_s("phi", model);
phi->setSelfConjugate(true);

// Complex field for the complex SU(3) flavor
Particle psi = diracfermion_s("psi", model);
```

Setting the representations

```
phi->setFundamentalFlavorRep("R");
psi->setFundamentalFlavorRep("C");
```

Note As MARTY may be extended for non fundamental representations in the future, one has to precise that the representation is fundamental through the function name.

3.4 Quantum Fields in expressions

This section presents how to handle quantum fields in symbolic expressions. This is important when doing model building in MARTY as a BSM Lagrangian requires in general custom interactions other than gauge couplings (automatically provided by MARTY). This section is about how to obtain or create all objects required to build interaction terms, and how to write the couplings in a way that MARTY understands. When the model is already provided, the features presented in the following are not necessary to use MARTY.

3.4.1 Indices

To create symbolic quantum field objects, one must first define indices for the particles as explained in figure 3.1. Minkowski, Dirac and gauge indices can be obtained simply from the user interface. There is several ways to get indices, the simplest one is presented in sample code 18. Group indices are more complicated to obtain. There is one vector space per irreducible representation and per gauged group. The model will generate group indices given:

- The number of different indices that must be generated.
- The model.
- The group that can be specified with its name.

- A particle name, the model returns indices in the vector space corresponding to the representation of the particle in this particular group.

In the unbroken Standard Model for example there are 7 independent index families:

- **Minkowski** indices for space-time and vector bosons generally noted $\mu, \nu \ldots$
- **Spinor** indices in Dirac space generally noted $\alpha, \beta \ldots$
- **Flavor** indices for the three fermion generations generally noted $I, J \ldots$
- **Gluon** color indices (octet representation of $SU(3)_C$) generally noted $A, B \ldots$
- **Quark** color indices (triplet representation of $SU(3)_C$) generally noted $a, b \ldots$
- **Weak boson** indices (triplet representation of $SU(2)_L$) generally noted $A, B \ldots$
- **Weak indices for matter particles** (doublet representation of $SU(2)_L$) generally noted $i, j \ldots$

Sample code 18: Generating indices

Minkowski indices

```
auto mu = MinkowskiIndices(10); // 10 indices
```

Dirac indices

```
auto alpha = DiracIndices(10); // 10 indices
```

Gauge indices, for the representation of the particle `"phi"` in the gauged group `"g"`

```
auto A = GaugeIndices(10, model, "g", "phi"); // 10 indices
```

Flavor indices, for the flavor `"f"`

```
auto I = FlavorIndices(10, model, "f"); // 10 indices
```

Indices can then be used simply using the subscript operator `[]`

```
mu[i]; // is a Minkowski index for i in [0, 10[
alpha[i]; // is a Dirac index for i in [0, 10[
A[i]; // is a Group index for i in [0, 10[
I[i]; // is a Flavor index for i in [0, 10[
```

Note The `auto` keyword allows the user to not care about the type of the index collection i.e. `std::vector<csl::Index>`.

3.4.2 Space-time point

Quantum fields are tensor fields. They need then a space-time point to live, and `MARTY` for now is limited to the Minkowski space[6]. One simply generates a space-time point (vector in Minkowski space) following the prescription given in sample code 19.

6. This statement does not mean that tensor fields can only live in the Minkowski space. `CSL` is general and there is no such limitation. `MARTY` calculations are limited to the Minkowski space-time for now, so are quantum fields.

Sample code 19: Generating space-time points

The easy way

```
Tensor X = MinkowskiVector("X");
```

The alternative way

```
Tensor X("X", &Minkowski);
```

Note The second way is fully general and can be used for any vector space (`Minkowski` here).

When building a Lagrangian, there is no support in `MARTY` for non-local terms i.e. not located on the same space-time point. `MARTY` thus allow users to omit this detail and give only indices when building a Lagrangian, basically replacing in interaction terms

$$\phi_I(X) \mapsto \phi_I, \tag{3.8}$$

where the space-time point X is added by default automatically.

3.4.3 Creating an expression from a Particle

We saw in section 3.4.1 how to generate indices for quantum fields and in section 3.4.2 how to generate (or omit) space-time points, the two required ingredients to create symbolic quantum fields from particles.

Let us consider the example of a $SU(3)_C \times SU(2)_L$ gauge with a fermion Q in the fundamental representation of both groups, and a vector W in the adjoint representation of the $SU(2)_L$ group. In addition, we introduce a complex $SU(3)$ flavor named 'F' for the fermion in order to have a complete example. Sample code 20 shows how to create a gauge interaction term as the following

$$\mathcal{L} \ni ig\bar{\psi}\not{W}\psi = ig\bar{\psi}^{Iai}_{\alpha}(X)\gamma^{\mu}_{\alpha\beta}W^{A}_{\mu}(X)T^{A}_{ij}\psi^{Iaj}_{\beta}(X), \tag{3.9}$$

with μ a Minkowski index, (i, j) indices in the doublet representation of $SU(2)_L$, a an index in the triplet representation of $SU(3)_C$, A an index in the triplet representation of $SU(2)_L$ and I an index in the 3-dimensional complex flavor for the fermion. This example is complicated on purpose because it shows how to create arbitrary interaction terms in `MARTY`, introducing all the necessary user interface to do so. One may notice that all indices are explicit in `MARTY`. We chose to have one unique and general treatment for all indices rather than having a mix between explicit and implicit indices. The interface could be improved in the future allowing for example to omit indices in bi-linear diagonal couplings (like $SU(3)_C$ and the flavor indices in equation 3.9), but for now all indices must be given.

It is important to know the order of indices defined for quantum fields. If not given in order, `MARTY` will raise an error and stop the program. The order is the following, to give from left to right:

- **Flavor indices** of non trivial representations, in the same order than when adding the corresponding flavors to the model.

- **Gauge indices** of non trivial representations in the same order than when adding the corresponding gauged groups to the model.
- **Space-time indices** for spin 1/2 particles (Dirac index) and spin 1 (Minkowski index).

The term presented in equation 3.9 and in sample code 20 is a gauge interaction, that is of course given automatically by MARTY. Knowing how to build it by hand is nevertheless a good starting point to create general BSM interactions.

Sample code 20: From particles to symbolic expressions

Consider a gauge formed by a $SU(3)$ group `"C"` and a $SU(2)$ `"L"`, with `psi` in the fundamental representation of both groups and in an additional $SU(3)$ flavor `"F"`, and finally `W` in the adjoint of the $SU(2)$.

Generating indices and space-time point

```
auto I = FlavorIndices(1, model, "F");
auto a = GaugeIndices(1, model, "C", "psi");
auto A = GaugeIndices(1, model, "L", "W");
auto i = GaugeIndices(2, model, "L", "psi");
auto mu = MinkowskiIndices(1);
auto al = DiracIndices(2);
Tensor X = MinkowskiVector("X");
```

Getting the two additional tensors we need

```
Tensor gamma = DiracGamma();
Tensor T = GetGenerator(model, "L", "psi");
```

Creating the expression

```
Expr g = constant_s("g");
Expr term = CSL_I * g
* GetComplexConjugate(psi({I[0], a[0], i[0], al[0]}, X)),
* W({A[0], +mu[0]}, X)
* T({A[0], i[0], i[1]})
* gamma({mu[0], al[0], al[1]})
* psi({I[0], a[0], i[1], al[1]}, X);
```

Note Here as all fields have the same point X, one can omit it and the default space-time point of MARTY will be introduced automatically.

Note There is no need for γ^0 as in MARTY $\psi^\dagger$ is defined as $\bar{\psi} = \psi^\dagger\gamma^0$. This saves a lot of unnecessary calculations.

See also Sample code 27 for more details on how to get generators from a model.

3.4.4 Type system

CSL type system does not include MARTY objects, and in particular quantum fields. MARTY extends the type system to generalize it to any type, even types that may be later user-defined [7]. While CSL type system allows one to find out the type of an expression (number, tensor, sum, etc), the extended type system allows one to compare the expression to any given type (including MARTY types) and convert back to it (if the type is correct). The example given in sample code 21 shows how to, from a CSL expression containing a quantum field, recover the field inside the expression and its Particle parent.

Sample code 21: From symbolic expressions to particles

We consider that sfield is an expression (Expr) containing a quantum field.

Using only CSL, we cannot obtain particle properties

```
cout << IsIndicialTensor(sfield) << endl;
// >> 1
cout << GetPrimaryType(sfield) << endl;
// >> Indicial
cout << GetType(sfield) << endl;
// >> TensorFieldElement
```

Using the extended type system in MARTY, obtaining the QuantumField and the Particle

```
if (IsOfType<QuantumField>(sfield)) {
    QuantumField field = ConvertTo<QuantumField>(sfield);
    Particle particle = field.getParticle();
}
```

Warning If one wants to modify the quantum field and the symbolic expression at the same time, one must get a pointer to the field:

```
if (IsOfType<QuantumField>(sfield)) {
    QuantumField *field = ConvertToPtr<QuantumField>(sfield);
    // Here modification of field will affect sfield also
    Particle particle = field->getParticle();
}
```

3.4.5 Polarization field

Polarization fields are another quantum field type object in expressions. They represent momentum space fields with explicit spin indices. This gives

$$\phi(X) \to \phi(p) \text{ for scalars,} \tag{3.10}$$
$$A^{\mu}(X) \to \epsilon^{\mu}_{\lambda}(p) \text{ for vectors,} \tag{3.11}$$
$$\psi_{\alpha}(X) \to u_{\alpha\sigma}(p) \text{ for fermions,} \tag{3.12}$$

7. This generalization has a cost which is to be slightly less simple and optimized.

with λ a spin index for the vector and σ for the fermion. The relation between position- and momentum-space fields for fermions is:

$$\psi_\alpha(X) = \int \frac{d^3p}{(2\pi)^3} \frac{1}{\sqrt{2p^0}} \sum_{\sigma=\pm 1/2} \left(u_{\alpha\sigma}(p) a_{\sigma,p} e^{-ipX} + v_{\alpha\sigma}(p) b^\dagger_{\sigma,p} e^{+ipX} \right), \tag{3.13}$$

with $a_{\sigma,p}$ and $b_{\sigma,p}$ quantum annihilation operators. While $\psi(X)$ and $\psi(Y)$ do not commute because of quantum properties, $u(p)$ and $v(p)$ do because they are simply (possibly complex) coefficients. For scalars the spin-0 object $\phi(p)$ given in 3.10 is not defined in general because it is trivial, but allows `MARTY` to keep track of external fields in the result and is therefore used in practice.

Spin indices are important when squaring an amplitude as spin sum rules must be applied:

$$\sum_\lambda \epsilon^\mu_\lambda(p) \epsilon^{*\nu}_\lambda(p) \to -ig^{\mu\nu} \tag{3.14}$$

for the photon for example, and

$$\sum_\sigma u_{\alpha\sigma}(p) \bar{u}_{\beta\sigma}(p) \to \left(\not{p} + m \right)_{\alpha\beta} \tag{3.15}$$

for fermion particles.

All the statements given for quantum fields in sections 3.4.3 and 3.4.4 are also valid for polarization fields, with two main differences:

- The type of expression is not the same, `QuantumField` must then be replaced by `PolarizationField` when testing the type or converting a symbolic expression (see sample code 21).
- Polarization fields are generated also by the same particle, giving a polarization index separately before the other indices. An example is presented in sample code 22.

Sample code 22: Polarization fields

Taking the same example as in sample code 20, we create here the same term but with `PolarizationField` objects instead of `QuantumField` objects (useless here but for pedagogical purpose)

Creating polarization indices

```
auto pol = Euclid_R3.generateIndices(3);
```

Creating the expression

```
Expr g = constant_s("g");
Expr term = CSL_I * g
* GetComplexConjugate(psi(pol[0], {I[0], a[0], i[0], al[0]}, X)),
* W(pol[1], {A[0], +mu[0]}, X)
* T({A[0], i[0], i[1]})
* gamma({mu[0], al[0], al[1]})
* psi(pol[2], {I[0], a[0], i[1], al[1]}, X);
```

Note Vector spaces for polarization indices do not matter. They can be built from any vector space in the program independently of the particle, such as the built-in 3 dimensional space `Euclid_R3` for example.

Warning For scalar bosons one must also give a polarization index, even if the particle has no spin.

This chapter presented the different types of fields that MARTY provides i.e. particles of spin 0, $1/2$ and 1 which are the basic components of BSM models. A particle is contained in a model and is in particular an irreducible representation of the gauge group. To describe particle interactions, we therefore need to embed these particles into a high energy physics model with a gauge group that represents the fundamental interactions of the theory. Model structures and how they are related to quantum fields and gauge groups in MARTY are introduced in the next chapter. Abstract group and representation theory implementations will be later detailed in chapter 5.

CHAPTER 4

Models for high energy physics

4.1 Introduction

Models for high energy physics are the main ingredients in BSM phenomenology. When searching for new physics, we must compare theory predictions and experimental measurements in order to understand which BSM scenarios are compatible with nature. The set of models that MARTY is able to build therefore represents the domain of elementary particle physics that we can describe, in particular among the phenomena that have not been observed yet. Models in MARTY lie in a 4-dimensional Minkowski space-time. The gauge group (i.e. the fundamental forces) can be any combination of semi-simple Lie groups that are discussed in chapter 5. However, interactions are not limited to fundamental gauge couplings and can be defined in a very general way as this chapter demonstrates.

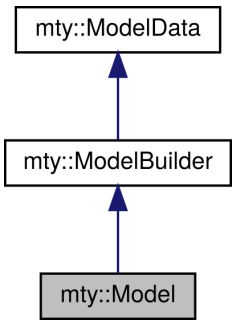


Figure 4.1 – Inheritance tree for the `Model` object. The three parts composing a model are the data container (the most fundamental, on the top), the model builder, and the final interface with calculation features.

In MARTY, `Model` objects represent beyond the Standard Model theories and contain the necessary methods for model building and calculations. The interface for models is divided into three different parts, `ModelData`, `ModelBuilder` and `Model`, for which the inheritance hierarchy is presented in figure 4.1.

- `ModelData` contains almost all the content of the model: Lagrangian, particles, gauge,

flavor, etc. It also provides methods to get and modify the different elements.

- `ModelBuilder` implements all model building features such as gauge symmetry breaking, particle replacement, diagonalization, etc. This interface is not presented in the following for simplicity but can be found in the dedicated chapter of the user manual [86].
- `Model` has methods to perform physical calculations in BSM scenarios. Feynman rules, amplitudes, squared amplitudes and Wilson coefficients can be calculated from this class. This interface is detailed in chapter 6 through the presentation of calculation procedures.

Class `ModelData` contains all the basic interface to store and manipulate the content of a high-energy physics model. Table 4.1 presents the main attributes of `ModelData` giving their types and roles in the program. One may not access them directly but only through different class methods. More details on how to manipulate these attributes are given in the next sections.

For the class methods that take a `Particle` as parameter, one can often give different objects, not only a `Particle`. Functions are templates, taking any argument that may be given to the `getParticle()` function. In other words, when `MARTY` asks the user a `Particle` object, any object may be given that is either directly a `Particle` or an object `MARTY` can use to find a particle through its `getParticle()` methods. [1] This simplifies the interface for users, as basically any object representing a field can be given to the template methods. This allows one to replace code like

```
model.doSomeThingWithParticle(model.getParticle("phi"));
```

by

```
model.doSomeThingWithParticle("phi");
```

In particular, one can give a name `"phi"` for example instead of searching the actual variable containing `phi`.

The same principle holds for gauge and flavor groups and the `getGroup()` methods. This means in particular that a method taking a group as parameter can use a group name instead of a `Group` object.

4.2 Adding / Removing particles

Particles may be added to or removed from a Model, in particular during model building. Once a particle is added in the model its gauge and flavor representations must not be changed as some default gauge interactions are introduced automatically by `MARTY` at that moment. Kinetic and mass terms are also introduced automatically, but as explained in section 4.4, masses of particles may still be changed later on. It is also possible to forbid `MARTY` to introduce any Lagrangian term by giving a boolean when adding the particle. A particle may also be removed from a model. In this case, the particle and

1. Such as a name, a `QuantumField` object, a `QuantumFieldParent` variable or an expression (`Expr`) if it represents a quantum field.

all the Lagrangian terms containing it are removed from the theory. All the procedures described above are presented in sample code 23.

Sample code 23: Adding / Removing particles

Creating particles

```
Particle psi = diracfermion_s("psi", model);
// Setting psi representation ...
Particle phi = scalarboson_s("phi", model);
// Setting phi representation ...
```

Adding the particles

```
model.addParticle(psi); // Adds psi with default interactions
model.addParticle(phi, false); // Adds phi without any Lagrangian term
```

Removing a particle

```
model.removeParticle(psi); // psi and all its interactions are removed
```

Attribute name	Type	Purpose
L	Lagrangian	Lagrangian of the model, contains all interaction terms. See section 4.4 for more details.
spaceTime	Space const*	Space-time of the theory, for now always Minkowski.
gauge	unique_ptr<Gauge>	Gauge group, containing all gauged groups.
flavor	unique_ptr<Flavor>	Flavor, containing all flavor groups
particles	vector<Particle>	List of all particles in the model.
quantumNumbers	vector<QuantumNumber>	List of quantum numbers in the model.
scalarCouplings	vector<Expr>	List of all scalar couplings, in particular gauge couplings.
tensorCouplings	vector<Tensor>	List of all tensor couplings.
gaugeLocked	bool	Tells if the gauge is fully initialized (true) and if particle content may be added.

Table 4.1 – Attributes of the ModelData class with their type and purpose.

4.3 Obtaining / Defining couplings

It is important to have access to the couplings present in the model, in case for example one wants to extend the model using them. As shown in table 4.1 there are two kinds

of couplings in a MARTY model:

- **Scalar couplings.** Gauge couplings are automatically added in this category when created.
- **Tensor couplings.** Initially empty, can contain any tensor.

One can obtain expressions for scalar couplings (Expr objects, typically constants) and tensors for tensor couplings (Tensor objects) from a model, given the name of the coupling. Initially the model only contains gauge couplings, but one can add couplings (scalars or tensors) to the model, and retrieve them later on. As we saw in figure 3.3, gauge couplings are defined initially with a name "g_<group-name>". This procedure is summarized in sample code 24.

Sample code 24: Managing couplings

Getting gauge couplings

```
// Getting the couplings of two gauge groups "Y" and "L"
Expr g_Y = model.getScalarCoupling("g_Y");
Expr g_L = model.getScalarCoupling("g_L");
```

Adding couplings

```
Expr e = constant_s("e");
model.addScalarCoupling(e);
// Defining our own gamma matrix
Tensor my_gamma("gamma", {&Minkowski, &dirac4, &dirac4});
model.addTensorCoupling(my_gamma);
```

Getting tensor couplings

```
Tensor my_gamma_2 = model.getTensorCoupling("gamma");
```

Note The first example is the most important to remember because there is no other simple way to get default gauge couplings from a model.

4.4 Lagrangian

The Lagrangian is almost entirely encapsulated by the ModelData class. In other words, a user will probably not directly interact with it, but through the interface functions of the model. Section 4.5 shows for example how to add interaction terms to the Lagrangian, using methods of the ModelData class. In general, the Lagrangian is just a container of symbolic expressions, and all modifications to it are ordered by Model classes.

4.4.1 Lagrangian in MARTY

The Lagrangian contains all kinetic, mass, and interaction terms of the theory. An InteractionTerm in MARTY may contain any of the three kinds of terms. The Lagrangian is therefore a collection of such objects, divided in three parts:

- **Kinetic terms.** Purely informative. No physics in `MARTY` depend on them, but they can be used to check if the user-defined model building prescriptions are consistent.
- **Mass terms.** For built-in models, they do not have any impact on the calculations either. However during model building `MARTY` uses this part of the Lagrangian to determine masses of the particles and matrices to diagonalize.
- **Interaction terms.** They determine the physics and vertices used in Feynman diagrams. They are used to derive Feynman rules (see section 6.2.2 for more details).

Amplitude calculations depend on two main objects for a given model, propagators and vertices. As we saw in section 3.3.2, the mass and width of a particle to insert in the propagators are taken from the particle itself, not the model. Changing a mass explicitly as shown in sample code 15 will not change the mass Lagrangian. Mass terms are meant to be used during model building but will not prevent a user to change masses and widths for any particle before running a calculation.

Kinetic terms could in principle affect physics because they determine the free equations of motions i.e. the propagators. A scalar Lagrangian such as

$$\mathcal{L} = \frac{1}{2}\partial_\mu\phi\partial^\mu\phi - \frac{1}{2}m^2\phi^2 \tag{4.1}$$

implies the following equation of motion for ϕ

$$(\Box + m^2)\phi = 0, \tag{4.2}$$

which in turns gives a propagator of the type

$$\frac{1}{\Box + m^2} \to \frac{1}{-p^2 + m^2}. \tag{4.3}$$

Kinetic terms could be used to determine the propagators in `MARTY`, but would require unnecessary algebra. Instead all propagators are fixed with a denominator as in equation 4.3, letting the possibility to define custom propagators.

Feynman rules are calculated from the interaction Lagrangian. Together with particle propagators (masses, widths, ...), they are the building blocks of quantum field theory calculations.

4.4.2 Interaction terms

The `InteractionTerm` object is used for the three kinds of terms in the Lagrangian. It contains a mathematical expression corresponding to a term in a Lagrangian such as

$$ig\bar{\psi}_{i\alpha}\gamma^\mu_{\alpha\beta}W^A_\mu T^A_{ij}\psi_{j\beta}. \tag{4.4}$$

For model building procedures, the `InteractionTerm` class can easily provide the relevant properties of an interaction e.g. the particle content.

`InteractionTerm` also keep track of all different index contractions in the interaction. When calculating Feynman rules, the Lagrangian expansion is done explicitly using interaction terms. The Wick theorem and most of the algebra is done with a generic expression, with only the field content, and the `InteractionTerm` class recovers all factors and

index symmetries in the final result. Taking the example of equation 4.4, the Feynman rule calculation injects in the Wick theorem a generic term such as

$$\bar{\psi}_{i\alpha} W^A_\mu \psi_{j\beta} \tag{4.5}$$

with only free indices, and the `InteractionTerm` object is asked to recover, in the final result, the initial index structure and factors of the Lagrangian term.

4.5 Adding Lagrangian terms

There are three ways to add a Lagrangian term in MARTY. The first one is automated, just by adding a new particle in the model. In this case MARTY initializes automatically gauge interactions, default kinetic and mass terms. The second way is to use built-in functions to add common mass terms for different kinds of particles. Finally, as presented in section 3.4.3, it is possible to build general interaction terms building explicitly the corresponding mathematical expression.

4.5.1 Built-in interaction terms

There are for now only three types of interaction terms in MARTY one can easily add to a model. These are bosonic mass terms

$$\mathcal{L} \ni -s\eta \cdot m^2 \phi^\dagger \phi, \tag{4.6}$$

Dirac or Majorana mass terms

$$\mathcal{L} \ni -\eta \cdot m \bar{\psi}\psi, \tag{4.7}$$

and Weyl mass terms

$$\mathcal{L} \ni -m \left(\bar{\psi}_R \psi_L + \bar{\psi}_L \psi_R \right). \tag{4.8}$$

In the above equations, η is 1 for complex fields, and $\frac{1}{2}$ for real ones, $s = +1$ for scalars and $s = -1$ for vectors. In the Dirac mass terms $\eta = \frac{1}{2}$ corresponds to a Majorana mass. The procedure to add such mass terms is presented in sample code 25. The η and s factors are determined automatically by MARTY.

Sample code 25: Adding mass terms explicitly

For a boson B (scalar or vector)

```
Expr M = constant_s("M");
model.addBosonicMass("B", M);
// Or
model.addBosonicMass("B", "M");
```

For a Dirac or Majorana fermion F

```
Expr m = constant_s("m");
model.addFermionicMass("F", m);
// Or
model.addFermionicMass("F", "m");
```

For a pair of Weyl fermions (L + R) F_L and F_R

```
Expr m = constant_s("m");
model.addFermionicMass("F_L", "F_R", m);
// Or
model.addFermionicMass("F_L", "F_R", "m");
```

Note When adding a particle to a model, a mass term will be defined automatically if it has a non zero mass. This procedure is useful when one wants to add a mass explicitly during model building for example.

4.5.2 General interactions

To add general interactions one must provide explicitly the expressions to MARTY. It is more involved but completely general and allows us to build any BSM Lagrangian. The procedure to build these expressions was introduced in section 3.4.3 with an example.

An unusual fermion bilinear definition

Before presenting the main ingredients to build general interactions, let us discuss the conventions and notations used for spin $1/2$ particles. In MARTY the hermitic conjugate of a fermion, $\psi^\dagger$, is defined as being equal to the usual $\bar{\psi} = \psi^\dagger\gamma^0$ to avoid to deal with too many unnecessary γ^0 matrices in vertex definitions and calculations. This means that a term such as

$$\bar{\psi}\gamma^\mu\chi = \psi^\dagger\gamma^0\gamma^\mu\chi, \tag{4.9}$$

is instead defined following

$$\tilde{\psi}^\dagger\tilde{\gamma}^\mu\tilde{\chi}, \tag{4.10}$$

i.e. by removing γ^0 and redefining fields $\tilde{\psi}$ and $\tilde{\chi}$. This is possible when considering only fermion bilinears, [2] with the redefinition of the hermitic conjugation of γ-matrices. With explicit γ^0 one has for example

$$(\gamma^\mu)^\dagger = \gamma^0\gamma^\mu\gamma^0, \tag{4.11}$$

2. Fermion interactions for BSM physics are always defined using fermion bilinears.

implying

$$\begin{aligned}\left(\psi^\dagger\gamma^0\gamma^\mu\chi\right)^\dagger &= \chi^\dagger\gamma^0\gamma^\mu\psi \\ &= \bar{\chi}\gamma^\mu\psi,\end{aligned} \tag{4.12}$$

where we used $(\gamma^0)^2 = 1$. Keeping γ^0 implicit and defining

$$\left(\tilde{\psi}^\dagger\right)^\dagger \equiv \tilde{\psi}, \tag{4.13}$$

one must define

$$(\tilde{\gamma}^\mu)^\dagger = \tilde{\gamma}^\mu \tag{4.14}$$

to obtain an equivalent property. With these definitions we indeed recover the initial behavior of our bilinear:

$$\left(\tilde{\psi}^\dagger\tilde{\gamma}^\mu\tilde{\chi}\right)^\dagger = \tilde{\chi}^\dagger\tilde{\gamma}^\mu\tilde{\psi}. \tag{4.15}$$

For a general matrix Γ, if $\Gamma^\dagger \equiv \Gamma'$ with explicit γ^0 one has to define

$$\tilde{\Gamma}^\dagger \equiv \gamma^0\Gamma'\gamma^0 \tag{4.16}$$

to perform consistent calculations with fermion bilinears without using γ^0 anywhere.[3] For automated calculation purposes this has a great impact on performance. The only consequence at the user level is the implicit use of γ^0 when writing the hermitic conjugate of a fermionic field $\psi^\dagger \equiv \bar{\psi}$.

Ingredients to build a general BSM Lagrangian

Let us review here the main ingredients for building interaction terms from scratch.

- **Particles.** One must have the `Particle` objects, required to generate quantum fields in expressions. They can be user-defined or obtained from a model, see sample code 13.
- **Indices.** Indices are necessary to generate symbolic expressions from particles. Sample code 18 shows how to get all the necessary indices and sample code 20 how to use them.
- **Space-time point.** This ingredient is not necessary for interaction terms. Users may omit it and `MARTY` will automatically place all fields at the space-time point defined for the rest of the Lagrangian. Sample code 19 gives more details on how to generate space-time points if needed.
- **Gauge couplings.** One may have to define new interactions depending on the gauge couplings of the model. The procedure to get gauge couplings has been developed in sample code 24.
- **γ-matrices.** They are built-in and may be accessed simply as shown in sample code 26.

3. γ^0 matrices in equation 4.16 can always be simplified using commutation and contraction identities as demonstrated for $\Gamma = \gamma^\mu$.

- **Generators.** One may have to define new interactions depending one the group generators (T^A_{ij}, f^{ABC} etc) of the model. All generators can be obtained through interface function calls as depicted in sample code 27.

- **Vector spaces.** To create a new custom tensor, one has to obtain the vector spaces (`csl::Space`) corresponding to all the tensor indices. This is presented in sample code 28.

- **CSL.** The CSL manual [88] contains all the details and allow the users to write interactions, building new couplings and tensors. Terms can be defined in a very general way and are not limited to particular forms.

With these ingredients it is possible to write any unreasonably complicated Lagrangian starting from the example in sample code 20.

Sample code 26: Getting γ-matrices

The easy way, using the interface

```
Tensor gamma = DiracGamma(); // gamma matrix
Tensor gamma5 = DiracGamma5(); // gamma5 matrix
Tensor sigma = DiracSigma(); // sigma matrix
Tensor P_L = DiracPL(); // left projector
Tensor P_R = DiracPR(); // right projector
Tensor C = DiracCMatrix(); // Conjugation matrix
```

Sample code 27: Getting group generators

Taking the Standard Model example, there is a `"Q_L"` particle in the doublet representation of a $SU(2)$ `"L"` and a gluon `"G"` in the adjoint representation of a $SU(3)$ `"C"`. One can obtain the generators for these representations.

Obtaining the generators from the model

```
Tensor T_SU2_2 = model.getGenerator("L", "Q_L");
Tensor f_SU3 = model.getGenerator("C", "G");
```

Getting the generators from the interface

```
Tensor T_SU2_2 = GetGenerator(model, "L", "Q_L");
Tensor f_SU3 = GetGenerator(model, "C", "G");
```

Sample code 28: Vector spaces

Taking an example with a `"C"` gauged group and an `"F"` flavor group.

From the interface

```
auto gaugeVectorSpace = GetVectorSpace(model, "C", "phi");
auto flavorVectorSpace = GetVectorSpace(model, "F", "phi");
```

From the model

```
auto gaugeVectorSpace = model.getVectorSpace("C", "phi");
auto flavorVectorSpace = model.getVectorSpace("F", "phi");
```

Note The `auto` here deduces the type `Space const*` that is a pointer to a constant `CSL` vector space.

Creating a new tensor from vector spaces

```
Tensor T("T", {&Minkowski, gaugeVectorSpace, flavorVectorSpace});
// T has one index in Minkowski, one gauge and one flavor index
```

Note Minkowski is a built-in `Space` object, the `&` symbol must thus be used to get a pointer whereas the spaces returned from the interface are already pointers (no need for `&`).

4.6 Fermion number violating interactions

4.6.1 Definition

This section is about fermion number violating interactions. Such vertices require a particular care from `MARTY`, but also from the user. Figure 4.2 presents examples of interactions that may violate the fermion number.

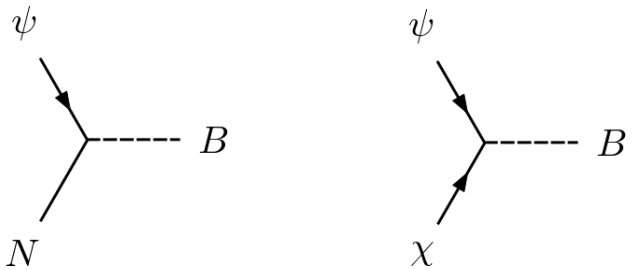


Figure 4.2 – Examples of interactions that can lead to fermion number violating processes. ψ and χ are regular spin $1/2$ particles, N a Majorana and B a boson (scalar or vector). These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

There is a calculation difficulty with these vertices in processes such as those depicted in figure 4.3 e.g. in SUSY models with SM fermions (ψ), neutralinos (N) and charginos (χ). A naive use of Feynman rules leads to unusual fermion bilinears because the fermion

number is not conserved along the line while we always want to have expressions such as

$$\bar{\psi}\Gamma\chi, \tag{4.17}$$

with Γ a combination of gamma matrices with indices flowing from left to right. The reason bilinears must be ordered is not only to have standard expressions, but mostly to be able to simplify them further using well-known identities. To solve this issue, we follow prescriptions similar to those in [112].

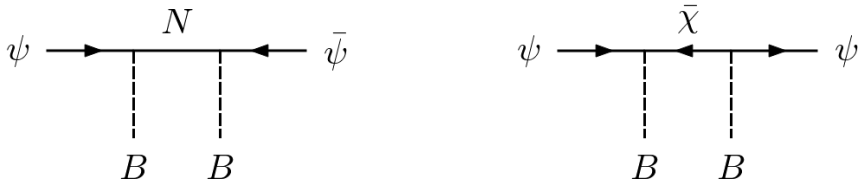


Figure 4.3 – Examples of fermion number violating (locally at least) processes. ψ is a regular spin 1/2, N a Majorana and B a boson (scalar or vector). These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

4.6.2 The conjugation matrix

The conjugation matrix C depends on the $\gamma-$matrix realization. In the Dirac realization a fermion is expressed as $\begin{pmatrix} \psi_L \\ \psi_R \end{pmatrix}$. In this basis, one has

$$C = -i\gamma^0\gamma^2. \tag{4.18}$$

In particular, $C = C^* = -C^T = -C^\dagger = -C^{-1}$. A charge conjugated fermion is then defined by

$$\psi^c \equiv C\bar{\psi}^T. \tag{4.19}$$

A Majorana fermion N is its own charge conjugated particle and the previous equation reads

$$N \equiv N^c = C\bar{N}^T. \tag{4.20}$$

Regarding fermionic external legs, the conjugation matrix has contraction properties with fermion external states $u(p)$ (and $v(p)$ for anti-particles). C is defined as the relation between particle (u) and anti-particle (v) spinors. In particular one has

$$\bar{u} = Cv, \tag{4.21}$$
$$\bar{v} = Cu, \tag{4.22}$$

and equivalently

$$u = \bar{v}C, \tag{4.23}$$
$$v = \bar{u}C. \tag{4.24}$$

C also fulfills properties with transposed gamma matrices namely

$$C\Gamma_i^T C^\dagger \equiv \Gamma_i' = \eta_i \Gamma_i, \tag{4.25}$$

with

$$\eta_i = \begin{cases} +1 & \text{for } \Gamma_i = 1, \gamma^5, \gamma^\mu\gamma^5, \\ -1 & \text{for } \Gamma_i = \gamma^\mu, \sigma^{\mu\nu}. \end{cases} \tag{4.26}$$

Rules given in [112] consist in defining a fermion line and inserting conjugation matrices C to recover a standard fermion bilinear such as the one in equation 4.17, while being mathematically equivalent. By ordering the fermion lines, the only subtlety is to obtain the correct sign at the end for the diagram in order for the interference patterns to remain consistent. Therefore, one needs to be careful about the signs while defining such interactions. This is explained in the next section.

4.6.3 Fermion number violation in MARTY

Golden rules

There are two golden rules in the definition of fermion number violating interactions in MARTY. The first rule is: **Not to use the charge conjugation matrix in vertices with Majorana fermions.** This is because from equation 4.20 we know that a conjugation matrix can be simplified away contracting it with a Majorana fermion. It is possible with MARTY to use charge conjugation matrices with Majorana interactions, but if one has to do it, it probably means that the vertex expression should be reconsidered by checking that no mistake has been made. Let us present an example of charge conjugation in a vertex with a Majorana N, a Dirac fermion ψ and a vector A

$$\mathcal{L} \ni \lambda A_\mu N \gamma^\mu \psi^c = \lambda A_\mu N \gamma^\mu C \bar{\psi}^T. \tag{4.27}$$

Using equations 4.20 and 4.25, one can transform the vertex into

$$\begin{aligned} \mathcal{L} \ni & \lambda A_\mu \bar{N}^T (\gamma^\mu)^T \bar{\psi}^T \\ = & \lambda A_\mu \left(\bar{N}^T (\gamma^\mu)^T \bar{\psi}^T \right)^T \\ = & -\lambda A_\mu \bar{\psi} \gamma^\mu \bar{N}\,, \end{aligned} \tag{4.28}$$

which does not contain any conjugation matrix. Vertices with two Majorana fermions should also follow this rule.

The second rule is: **For fermion number violating interactions between non-Majorana fermions, make sure that the position and sign of C are correct.** One should most of all make sure that conjugation matrices are placed correctly. In particular, in case a fermion number violating vertex or process vanishes incorrectly, conjugation matrices introduced in the Lagrangian should be checked first, in particular considering that

$$C\gamma^\mu \neq \gamma^\mu C, \tag{4.29}$$

$$C_{\alpha\beta} = -C_{\beta\alpha}. \tag{4.30}$$

The way to obtain the symbolic tensor corresponding to the conjugation matrix C in MARTY was presented in sample code 26.

Conventions can be misleading

The convention used to define fermions in MARTY is based on 4-component objects with explicit indices. It is different from the 2-component notation in which the conjugation matrix is not explicit. Using the latter convention, up- or down-indices have a different meaning and a spinor contraction like $\psi^\alpha \chi_\alpha$ with two left-handed Weyl fermions, usually abbreviated as $\psi \cdot \chi$, contains an implicit conjugation matrix when expressed in MARTY's convention, namely

$$\begin{aligned} \psi^\alpha \chi_\alpha &= \epsilon^{\alpha\beta} \psi_\alpha \chi_\beta \\ &= i\sigma_2^{\alpha\beta} \psi_\alpha \chi_\beta \\ &= \psi_\alpha C_{\alpha\beta} \chi_\beta, \end{aligned} \tag{4.31}$$

where we used 4-component notation together with the definition in equation 4.18 for the last step. ϵ is the fully anti-symmetric symbol verifying

$$\epsilon^{12} = -\epsilon^{21} = 1, \tag{4.32}$$

and $\sigma_2 = -i\epsilon$ is the second Pauli matrix. In MARTY no implicit tensor is used for index contraction i.e. all spinor indices are equivalent (no up- or down-type indices), and 4-component tensor couplings between spinors must be given.

In summary, one should remember the convention used by MARTY when defining fermion interactions and consider translating any Lagrangian expressed with a 2-component notation before building interaction terms with MARTY. In particular, a well-defined object defined in the 2-component notation properly translated in MARTY's convention should behave well in calculations, allowing the simplification procedures to finish correctly. Otherwise MARTY will complain and stop the calculation, in which case fermion interactions should be double checked in the Lagrangian. For more definitions and identities using the 2-component notation see [113].

4.7 Group theory objects

The `ModelData` class also contains the gauge group defining the theory's fundamental interactions. In general, one can obtain from a model the `Gauge`, `Flavor`, all `GaugedGroup` and `FlavorGroup` objects. In addition, the particle representations for the whole gauge (`GaugeIrrep`), the flavor (`FlavorIrrep`) or a specific group (`Irrep`) can be obtained.

Manipulating these objects is a deep feature of MARTY and is not needed in standard use cases. Representation theory is however an important feature of high energy physics models, this is why we present how to access particle representations in the following. We present the group theory structure in MARTY, starting from a high energy physics model up to the more abstract mathematical definitions of semi-simple Lie algebras. This will also provide a transition to the next chapter dedicated to group theory. Figure 4.4 presents a summary of the different interfaces presented in this section, and in particular how to get from a high-energy physics model to abstract group theory implementations in MARTY.

Except for representation objects, all groups (flavor or gauged) must be used as **pointers**. This is because they are unique in the program and must then not be copied.

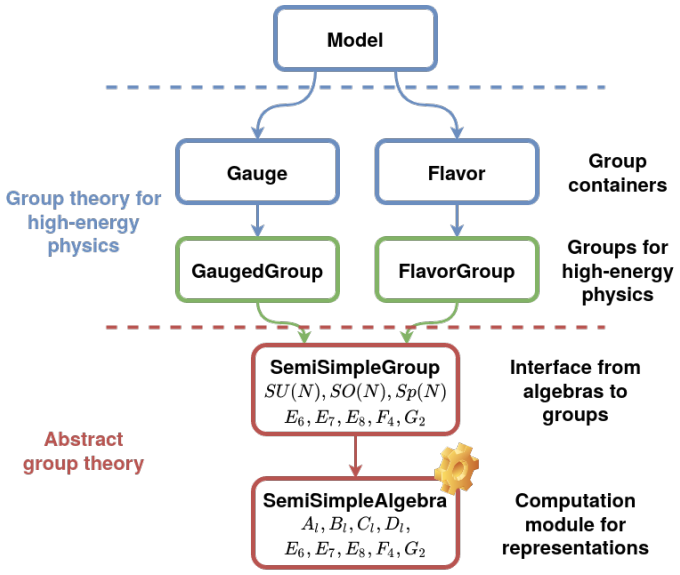


Figure 4.4 – Sketch of the successive logical top-down links from a high-energy physics model to abstract group theory implementations in `MARTY`. Representations are derived by `SemiSimpleAlgebra` and several interfaces exist to go from abstract representations to a particle physics model.

4.7.1 Gauge and flavor

The `Gauge` contains all the gauge groups and the `Flavor` all the flavor groups. They can be accessed as shown in sample code 29.

Sample code 29: Gauge and flavor

Getting the gauge and flavor of a model

```
Gauge *gauge = model.getGauge();
Flavor *flavor = model.getFlavor();
```

Warning Pointers to these objects must always be kept as they **must not** be copied.

4.7.2 Gauged and flavor groups

A gauge contains several gauged groups, and a flavor several flavor groups. From their names (user-defined when created), one can obtain the groups from a model, as depicted in sample code 30.

Sample code 30: Gauge and flavor groups

Considering a model with a $SU(3)$ `"C"` gauged group, and a $SU(3)$ `"F"` flavor.

Getting the gauge and flavor groups from the model

```
GaugedGroup *ggroup = model.getGaugedGroup("C");
FlavorGroup *fgroup = model.getFlavorGroup("F");
```

Warning The pointers to these objects must be kept as they **must not** be copied.

4.7.3 Gauge representations

Obtaining the gauge representation of a particle in a specific group is a task that can be done easily using MARTY's interface. One has to specify the particle (or its name), and the group (or its name) for the model to return the corresponding representation. More details on representations will be given in chapter 5. Examples on how to get particle representations are given in sample code 31.

Sample code 31: Getting representations from particles

We define a particle `phi` in a model containing a $SU(3)$ `"C"` gauged group, and a $SU(3)$ `"F"` flavor.

Getting the full gauge and flavor representations

```
GaugeIrrep gaugeRep = model.getGaugeIrrep("phi");
FlavorIrrep flavorRep = model.getFlavorIrrep("phi");
```

Getting a specific group representation (gauged or flavor)

```
Irrep ggroupRep = model.getGroupIrrep("phi", "C");
Irrep fgroupRep = model.getFlavorIrrep("phi", "F");
```

4.7.4 Groups and algebras

`GaugedGroup` and `FlavorGroup` objects are additional abstraction layers with respect to groups from a pure group theory point of view. They contain particular quantum field theory implementations that are not required to define abstract groups, e.g. couplings and generators (these objects are presented in sections 4.3 and 4.5.2 respectively). There are in MARTY deeper data structures for groups and algebras. Groups that MARTY can define are semi-simple Lie groups, coming with their semi-simple Lie algebras. The class `SemiSimpleAlgebra` implements the abstract representation machinery of MARTY, and `SemiSimpleGroup` is an interface to connect algebras to groups we know better ($SU(N)$, $SO(N)$, $Sp(N)$) as figure 4.4 shows. The way to obtain groups and algebras from a model is presented in sample code 32.

Sample code 32: Abstract groups and algebras

Considering a model with a $SU(3)$ `"C"` gauged group, and a $SU(3)$ `"F"` flavor.

Getting gauged and flavor groups

```
GaugedGroup *ggroup = model.getGroup("C");
FlavorGroup *fgroup = model.getFlavor("F");
```

Getting abstract groups from gauged and flavor groups

```
SemiSimpleGroup *group1 = ggroup->getGroup();
SemiSimpleGroup *group2 = fgroup->getGroup();
```

Getting algebras from the groups

```
SemiSimpleAlgebra *algebra1 = group1->getAlgebra();
SemiSimpleAlgebra *algebra2 = group2->getAlgebra();
```

Warning The pointers to these objects must always be kept as they **must not** be copied.

After discussing the link between group theory structures and high energy models in MARTY, the next chapter presents more abstract features of representation theory and the corresponding implementations in MARTY.

CHAPTER 5

Group theory

This chapter goes beyond the scope of amplitude calculations in quantum field theory as it presents a deeper feature of MARTY, its representation theory implementation, that is not required for perturbative calculations in high energy physics.

As we saw in section 3.3.3, a quantum field is an irreducible representation of the gauge group. We introduced in particular the link between semi-simple Lie algebras and groups in table 3.2, while table 3.3 presented the definition of the main representations used in physics in terms of Dynkin labels. This chapter introduces how irreducible representations are defined, computed in MARTY and the algebraic calculations that can be done with them.

Section 5.1 will present in details what semi-simple algebras are in MARTY, while sections 5.2 and 5.3 will respectively introduce irreducible representations (irreps) and the decomposition of irrep products into sums of irreps. These calculations are similar to what LieART [114], another Mathematica-based program, can do. Finally, section 5.4 will recall the link between these abstract group theory considerations and quantum field theory and in particular the calculations done by MARTY with gauge representations.

The Lorentz group A particle is also an irreducible representation of the Lorentz group, that corresponds to the spin. The Lorentz group $SO(1,3)$ has the same algebra as $SO(4)$. The algebra of $SO(4)$ is

$$\mathfrak{so}(4) = D_2 \cong A_1 \oplus A_1 = \mathfrak{su}(2) \oplus \mathfrak{su}(2). \tag{5.1}$$

In D_2 or $A_1 \oplus A_1$, one must give two Dynkin labels. The common representations are presented in table 5.1.

Particle	Dynkin labels	Dimension
Scalar	$(0,0)$	1
Left Weyl fermion	$(1,0)$	2
Right Weyl fermion	$(0,1)$	2
Dirac fermion	$(0,1) \oplus (1,0)$	$4 = 2 \oplus 2$
Vector	$(1,1)$	4

Table 5.1 – Correspondence between Lorentz representations (spin) and Dynkin labels in the algebra $D_2 \cong A_1 \oplus A_1 = \mathfrak{su}(2) \oplus \mathfrak{su}(2)$.

5.1 Semi-simple Lie algebras

5.1.1 Principle

Here we will not go into much theoretical details. More information about semi-simple Lie algebras can be found in [111, 114, 115].

Semi-simple Lie algebras have a common definition, and can be described with the same formalism. They are defined as having have no non-zero abelian ideal. One must find the maximal Cartan sub-algebra, whose dimension is called the rank ℓ of the algebra. There are seven different types of such algebras.

- A_ℓ for $\ell \geq 1$.
- B_ℓ for $\ell \geq 1$.
- C_ℓ for $\ell \geq 1$.
- D_ℓ for $\ell \geq 2$.
- E_ℓ for $6 \leq \ell \leq 8$.
- F_ℓ for $\ell = 4$.
- G_ℓ for $\ell = 2$.

Algebras E_ℓ to G_ℓ are called exceptional while A_ℓ to D_ℓ are the algebras of $SU(N)$, $SO(N)$ and $Sp(N)$ groups. They can however all be described with the same formalism. This is what is implemented in `MARTY` to define irreducible representations in all these algebras.

The Cartan sub-algebra defines the so-called simple roots of the semi-simple Lie algebra. An algebra of rank ℓ has exactly ℓ simple roots, and an irreducible representation is defined from them.

5.1.2 Semi-simple Lie algebras in `MARTY`

In `MARTY`, algebras A_ℓ to G_ℓ are `SemiSimpleAlgebra` objects. The way to create semi-simple algebras is presented in sample code 33. Irreducible representations will be presented in the next section.

Sample code 33: Semi-simple Lie algebras

Creating algebras

```
auto A2 = CreateAlgebra(algebra::Type::A, 2);
auto B7 = CreateAlgebra(algebra::Type::B, 7);
auto F4 = CreateAlgebra(algebra::Type::F4);
auto G2 = CreateAlgebra(algebra::Type::G2);
```

Note The type deduced by `auto` is `unique_ptr<SemiSimpleAlgebra>`. It is a pointer and one must use `->` to access member functions of `SemiSimpleAlgebra`.

Note Algebra names being very unspecific, the prefix `algebra::Type` is required to access algebra type names.

5.2 Irreducible representations

5.2.1 Highest-weight state

As we saw in the previous section, irreducible representations in a semi-simple Lie algebra are defined from ℓ simple roots. The roots define a ℓ-dimensional discrete lattice, in which states $|\psi_i\rangle$ are living. A state is defined with ℓ integer as

$$|\psi_i\rangle \equiv |i_1, i_2, \dots i_\ell\rangle, \quad i_j \text{ integers.} \tag{5.2}$$

An irreducible representation is defined uniquely by its highest-weight state, and from this highest-weight are deduced all other states of the representation $\mathcal{R}$. Each state $|\psi_i\rangle$ has a multiplicity m_i, and the dimension of the irrep is simply

$$d_{\mathcal{R}} = \sum_{|\psi_i\rangle \in \mathcal{R}} m_i. \tag{5.3}$$

From the highest weight state one can get all the states and their multiplicities using annihilation operators recursively.

5.2.2 The $\mathfrak{su}(2)$ example

Let us consider the simplest example, the spin in $SU(2)$. The algebra is A_1, with a one dimensional, discrete lattice. Highest weights must have a positive (unique) Dynkin label, and lower states are found by applying the only annihilation operator [1]

$$J^- = \frac{1}{\sqrt{2}}(\sigma^1 + i\sigma^2). \tag{5.4}$$

1. This is the same operator as the W^- boson in the SM that is the annihilation operator for the weak isospin: $W^- = \frac{1}{\sqrt{2}}(W^1 + iW^2)$.

From the spin 1/2 state, one can obtain the spin -1/2 state and find the 2-dimensional spin 1/2 representation. From the spin 1 highest weight, spin 0 and -1 states arise to finally obtain a 3-dimensional representation. This is represented in figure 5.1. Dynkin labels in $\mathfrak{su}(2)$ are just twice the spin value.

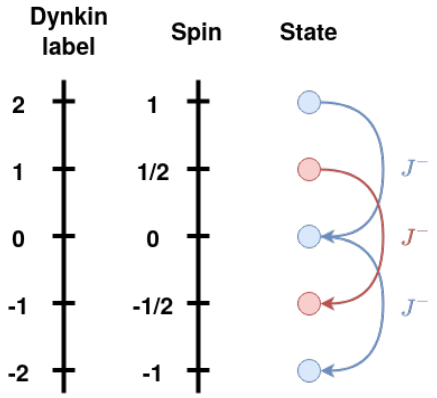


Figure 5.1 – 1-dimensional weight lattice of $A_1 = \mathfrak{su}(2)$, with spin 1/2 (red) and spin 1 (blue) representations with 2 and 3 states respectively. Correspondence with Dynkin labels is shown. The action of the annihilation operator $J^- = \frac{1}{\sqrt{2}}(\sigma^1 + i\sigma^2)$ is presented on the different states, starting each time from the highest weight of the representation.

This principle is then generalized in ℓ dimensions, and irreducible representations in all semi-simple Lie algebras can be uniquely defined.

5.2.3 The $\mathfrak{su}(3)$ example

Let us consider now the most complicated generalization of the previous section, that can still be represented on paper, the 2D case. The weight lattice of $A_2 = \mathfrak{su}(3)$ is 2-dimensional and is represented in figure 5.2.

Highest-weights may be any state in the dominant Weyl chamber, i.e. must have positive Dynkin labels. The state $|0, 0, \ldots, 0\rangle$ is always the trivial 1-dimensional representation. A highest weight state in $\mathfrak{su}(3)$ is defined by two Dynkin labels (2D plane), and this time there are two different annihilation operators that have to be applied recursively to the highest weight state to derive all states in an irreducible representation. The two annihilation operators are geometrically along $-\vec{\alpha}$ and $-\vec{\beta}$, with $\vec{\alpha}$ and $\vec{\beta}$ the two simple roots of A_2. Common representations, the quark, anti-quark and gluon of $SU(3)_C$ symmetry group are presented in figure 5.3.

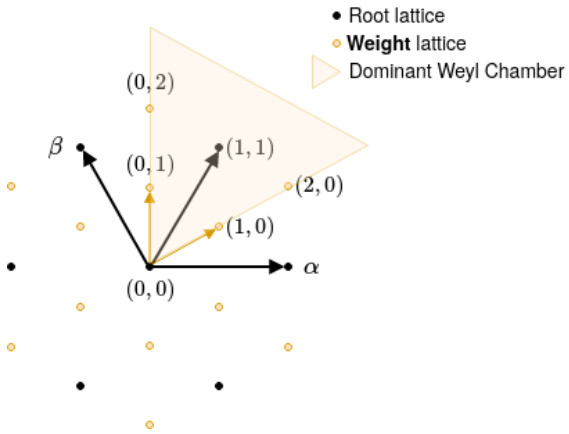


Figure 5.2 – Root and weight lattices of $\mathfrak{su}(3)$. α and β are the two simple roots. Dynkin labels of irreducible representations correspond to the position of the highest-weight state on the weight lattice. The dominant Weyl chamber is defined by the set of all positive-weight states. It contains all states that can be highest-weights, i.e. that can define an irreducible representation.

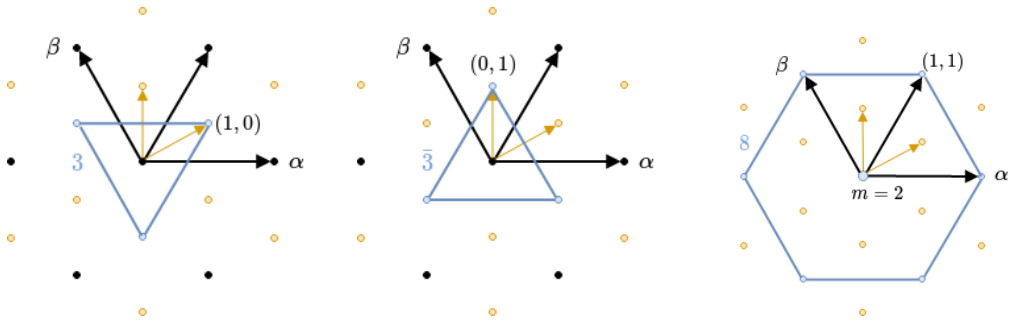


Figure 5.3 – Usual QCD representations (quark, anti-quark and gluon respectively) in the weight lattice of $\mathfrak{su}(3)$. The coordinates of their highest weight is shown. In the 8-dimensional representation the state of weight $(0, 0)$ has multiplicity 2, the total number of states is therefore indeed 8.

5.2.4 Irreducible representations in MARTY

From a semi-simple Lie algebra, one can build any irreducible representation given the Dynkin labels of its highest-weight state, as discussed in the previous section. The procedure to make MARTY derive an irreducible representation is presented in sample code 34. All the procedure, namely applying annihilation operators from the highest weight to find all the states and deriving the multiplicities, is automated in MARTY for all semi-simple Lie algebras.

Sample code 34: Irreducible representations of algebras

Taking algebras defined in sample code 33.

From the interface

```
Irrep quark = GetIrrep(A2, {1, 0});
Irrep gluon = GetIrrep(A2, {1, 1});
Irrep exotic = GetIrrep(F4, {1, 1, 0, 0});
```

Through member functions

```
Irrep quark = A2->highestWeightRep({1, 0});
Irrep gluon = A2->highestWeightRep({1, 1});
Irrep exotic = F4->highestWeightRep({1, 1, 0, 0});
```

Getting dimensions

```
cout << quark.getDimension() << endl; // 3
cout << gluon.getDimension() << endl; // 8
cout << exotic.getDimension() << endl; // 29172
```

5.3 Product decomposition

Products of irreducible representations have an important meaning in particle physics. Each interaction vertex is in fact such a product of the interacting particle representations. It can be decomposed in a sum of irreducible representations, the total number of dimensions being conserved. For an interaction to be physically consistent, the trivial representation must appear in the decomposition. With this procedure one can know for example the type of the representation that can be obtained from the annihilation of two particles, or what quark arrangements may result in a color-blind structure (mesons $q\bar{q}$, baryons qqq, or any combination of them). The procedure to decompose a product into a direct sum of irreducible representations can be found in [114]. Let us give examples with first a well-known $SU(2)$ spin

$$2 \otimes 2 = 1 \oplus 3, \tag{5.5}$$

with integers representing the dimensions of the representations. Group theory tells us that combining two $1/2$ spins one may either get a scalar or a spin 1 but not any other spin. For the collision of two gluons [2], we obtain

$$8 \otimes 8 = 1 \oplus 8 \oplus 8 \oplus 10 \oplus 10 \oplus 27, \tag{5.6}$$

which shows that we can get a neutral particle, another gluon-type particle or a more exotic 10- or 27-dimensional representation.

With a simple interface, one can make MARTY decompose such products. The result is not an `Irrep` but a `SumIrrep`. The interface allows one to sum, multiply and display any of

2. Purely in terms of $SU(3)$ color structure.

these objects. Sample code 35 shows two examples. The link between the $SU(3)$ example and physics interpretations is the answer to the question *'Which quarks arrangements can be color neutral and explain the structure of neutrons and protons ?'*. A neutral state is a trivial $SU(3)$ representation, i.e. of dimension 1. When taking products of quarks, one knows if it can be in a neutral state if the trivial representation appears in the decomposition. The example confirms that $q\bar{q}$ and qqq are the only states with 2 or 3 quarks that can be color neutral. [3]

Sample code 35: Representation product decomposition

Taking the sample algebras in sample code 33.

In the `A2` algebra

```
Irrep quark = GetIrrep(A2, {1, 0});
Irrep antiquark = GetIrrep(A2, {0, 1});

cout << "3 x 3 = " << quark * quark << endl;
// >> 3 x 3 = 3 + 6 (Total dim = 9)
cout << "3 x 3c = " << quark * antiquark << endl;
// >> 3 x 3c = 1 + 8 (Total dim = 9)
cout << "3 x 3 x 3 = " << quark * quark * quark << endl;
// >> 3 x 3 x 3 = 1 + 8 + 8 + 10 (Total dim = 27)
cout << "3 x 3 x 3c = " << quark * quark * antiquark << endl;
// >> 3 x 3 x 3c = 3 + 3 + 6 + 15 (Total dim = 27)
```

In the `F4` algebra

```
Irrep exotic1 = GetIrrep(F4, {1, 0, 0, 0});
Irrep exotic2 = GetIrrep(F4, {0, 0, 0, 1});
cout << exotic1 << endl;
// >> Representation |1,0,0,0> of dimension 52
cout << exotic2 << endl;
// >> Representation |0,0,0,1> of dimension 26
SumIrrep decomposition = exotic1 * exotic2;
cout << "52 x 26 = " << decomposition << endl;
// >> 52 x 26 = 26 + 273 + 1053 (Total dim = 1352)
```

5.4 Gauge representations

A gauge representation is simply a collection of group representations. The principle is the same as for `Irrep` and `SumIrrep`, but this time one manipulates representations in different groups at the same time in objects `GaugeIrrep` and `SumGaugeIrrep`. Taking a sample $SU(3)_C \otimes SU(2)_L$ SM gauge for example, one can take the product of a left-handed

3. Arbitrary combinations of those two building blocks will of course generate color-neutral states, as tetra-quarks $qq\bar{q}\bar{q}$ and penta-quarks $qqqq\bar{q}$ or $\bar{q}\bar{q}\bar{q}\bar{q}q$ for example.

quark with a left-handed anti-quark

$$(3,2) \otimes (\bar{3},\bar{2}) = (1 \oplus 8, 1 \oplus 3) = (1,1) \oplus (8,1) \oplus (1,3) \oplus (8,3). \tag{5.7}$$

For gauge representations, one may directly use a high-energy physics model with its interface as we showed in section 4.7. Sample code 36 presents how to perform the example above, build a Gauge from scratch independently of any quantum field theory consideration, and calculate the decomposition of $q\bar{q}$ in this $SU(3)_C \otimes SU(2)_L$ gauge.

Sample code 36: Irreducible representations of gauge groups

Building a $SU(3) \times SU(2)$ gauge

```
Gauge gauge;
gauge.addGroup(group::Type::SU, "C", 3);
gauge.addGroup(group::Type::SU, "L", 2);
```

Obtaining the quark $(3,2)$ and anti-quark $(\bar{3},\bar{2})$ representations

```
// Two representations between {},
// {1, 0} is SU(3) triplet
// {1}    is SU(2) doublet
GaugeIrrep quark = gauge.getRepresentation({{1, 0}, {1}});
GaugeIrrep antiquark = quark.getConjugatedRep();
cout << quark << endl;
// >> ( 3 , 2 )
cout << antiquark << endl;
// >> ( 3 , 2 )
```

Obtaining the decomposition of $q\bar{q}$

```
SumGaugeIrrep decomposition = quark * antiquark;
cout << decomposition << endl;
// >> ( 1 , 1 ) + ( 8 , 1 ) + ( 1 , 3 ) + ( 8 , 3 )
if (decomposition.containsTrivialRep()) {
    cout << "Contains␣trivial␣rep␣!" << endl;
    // >> Contains trivial rep !
}
```

Note The `containsTrivialRep()` method for gauge representations is used in `MARTY` to check that interaction terms do not obviously violate gauge invariance. They still can violate the gauge invariance even if respecting this condition but it is more difficult to test automatically.

5.5 Dynkin labels for common representations

In this section we present the correspondence between Dynkin labels and the most common irreducible representations (irreps) for all semi-simple Lie groups. In `MARTY`, Dynkin labels allow users to define uniquely irreps when creating particles. Each time, these positive integers must be given between `{}`, as for a $SU(3)$ `"C"` group

```
particle->setGroupRep("C", {1, 0});
```

for the fundamental representation of $SU(3)$ with Dynkin labels $(1, 0)$, typically quarks.

Gauge bosons are in the adjoint representation of their gauged groups, and generally non-trivial representations are the fundamental ones (doublet of $SU(2)$, triplet of $SU(3)$ etc), but this section also presents more exotic irreps. Trivial irreps (dimension 1) have always 0 Dynkin labels $(0, \cdots, 0)$ but need not to be defined in `MARTY`. The correspondence between groups and algebras A_ℓ, B_ℓ, C_ℓ and D_ℓ was given in table 3.2.

5.5.1 $\mathfrak{su}(N)$

$\mathfrak{su}(N)$ is an algebra of rank $N-1$ corresponding to A_{N-1}, therefore an irrep in that group is uniquely defined by $N-1$ positive integers. Starting from $\mathfrak{su}(2)$ with one label, common representations are presented in table 5.2.

Dynkin labels	Dimension	Common name
$(1, 0, \cdots, 0)$	N	Fundamental
$(0, \cdots, 0, 1)$	N	Anti-fundamental
$(1, 0, \cdots, 0, 1)$	$N^2 - 1$	Adjoint

Table 5.2 – Dynkin labels for common $\mathfrak{su}(N)$ irreducible representations.

The $\mathfrak{su}(2)$ case

For the $\mathfrak{su}(2)$ algebra, there is only one Dynkin label λ corresponding to the spin j through the relation

$$j = \frac{\lambda}{2}. \tag{5.8}$$

One can then straight-forwardly deduce the Dynkin label for a representation of spin j (dimension $2j + 1$) by multiplying it by 2.

The $\mathfrak{su}(3)$ case

$\mathfrak{su}(3)$ representations are defined with two Dynkin labels. Table 5.3 presents the correspondence with common $\mathfrak{su}(3)$ irreps.

Dynkin labels	Dimension	Common name
$(1, 0)$	3	Triplet
$(0, 1)$	3	Anti-triplet
$(1, 1)$	8	Adjoint
$(2, 0)$	6	Sextet
$(0, 2)$	6	Anti-sextet
$(2, 1)$	10	Decuplet
$(1, 2)$	10	Anti-decuplet

Table 5.3 – Dynkin labels for common $\mathfrak{su}(3)$ irreducible representations.

5.5.2 $\mathfrak{so}(N)$

$\mathfrak{so}(2\ell)$ and $\mathfrak{so}(2\ell+1)$ are algebras of rank ℓ namely D_ℓ and B_ℓ respectively. Particular cases must be mentioned for low ℓ to define vector, adjoint, and spinor representations of $\mathfrak{so}(N)$. Table 5.4 gives the Dynkin labels of common $\mathfrak{so}(N)$ irreducible representations.

Group	Algebra	Dynkin labels	Dimension	Common name
$SO(5)$	B_2	$(1,0)$	5	Vector
$SO(5)$	B_2	$(0,2)$	10	Adjoint
$SO(2\ell+1)$	B_l, $\ell \geq 3$	$(1,0,0,\cdots,0)$	$2\ell+1$	Vector
$SO(2\ell+1)$	B_l, $\ell \geq 3$	$(0,1,0,\cdots,0)$	$\ell(2\ell+1)$	Adjoint
$SO(4)$	D_2	$(1,1)$	4	Vector
$SO(4)$	D_2	$(2,1)$	6	Adjoint
$SO(6)$	D_3	$(1,0,0)$	6	Vector
$SO(6)$	D_3	$(0,1,1)$	15	Adjoint
$SO(2\ell)$	D_ℓ, $\ell \geq 4$	$(1,0,0,\cdots,0)$	2ℓ	Vector
$SO(2\ell)$	D_ℓ, $\ell \geq 4$	$(0,1,0,\cdots,0)$	$\ell(2\ell-1)$	Adjoint
$SO(2\ell)$	D_ℓ	$(0,\cdots,0,1,0)$	$2^{\ell-1}$	Left spinor
$SO(2\ell)$	D_ℓ	$(0,\cdots,0,0,1)$	$2^{\ell-1}$	Right spinor

Table 5.4 – Dynkin labels for the simplest $\mathfrak{so}(N)$ irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.3 $\mathfrak{sp}(N)$

$\mathfrak{sp}(2\ell)$ is an algebra of rank ℓ, namely C_ℓ. Dynkin labels for the fundamental and adjoint representations of $Sp(N)$ groups are given in table 5.5.

Group	Algebra	Dynkin labels	Dimension	Common name
$Sp(2\ell)$	C_ℓ	$(1,0,\cdots,0)$	2ℓ	Fundamental
$Sp(2\ell)$	C_ℓ	$(2,0,\cdots,0)$	$\ell(2\ell+1)$	Adjoint

Table 5.5 – Dynkin labels for the simplest $\mathfrak{sp}(2\ell)$ irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.4 E_6

E_6 is an exceptional algebra of rank 6. Dynkin labels for the simplest E_6 representations are given in table 5.6.

Dynkin labels	Dimension
$(1,0,0,0,0,0)$	27
$(0,1,0,0,0,0)$	351
$(0,0,1,0,0,0)$	2925
$(0,0,0,1,0,0)$	351
$(0,0,0,0,1,0)$	27
$(0,0,0,0,0,1)$	78
$(1,1,0,0,0,0)$	5824
$(1,0,1,0,0,0)$	51975
$(1,0,0,1,0,0)$	7371
$(1,0,0,0,1,0)$	650
$(1,0,0,0,0,1)$	1728

Table 5.6 – Dynkin labels for the simplest E_6 irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.5 E_7

E_7 is an exceptional algebra of rank 7. Dynkin labels for the simplest E_7 representations are given in table 5.7.

Dynkin labels	Dimension
$(1,0,0,0,0,0,0)$	133
$(0,1,0,0,0,0,0)$	8645
$(0,0,1,0,0,0,0)$	365750
$(0,0,0,1,0,0,0)$	27664
$(0,0,0,0,1,0,0)$	1539
$(0,0,0,0,0,1,0)$	56
$(0,0,0,0,0,0,1)$	912

Table 5.7 – Dynkin labels for the simplest E_7 irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.6 E_8

E_8 is an exceptional algebra of rank 8. Dynkin labels for the simplest E_8 representations are given in table 5.8.

Dynkin labels	Dimension
$(1,0,0,0,0,0,0,0)$	3875
$(0,0,0,0,0,1,0,0)$	30380
$(0,0,0,0,0,0,1,0)$	248

Table 5.8 – Dynkin labels for the simplest E_8 irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.7 F_4

F_4 is an exceptional algebra of rank 4. Dynkin labels for the simplest F_4 representations are presented in table 5.9.

Dynkin labels	Dimension
$(1,0,0,0)$	52
$(0,1,0,0)$	1274
$(0,0,1,0)$	273
$(0,0,0,1)$	26
$(1,1,0,0)$	29172
$(1,0,1,0)$	8424
$(1,0,0,1)$	1053
$(0,1,1,0)$	107406
$(0,1,0,1)$	19278
$(0,0,1,1)$	4096

Table 5.9 – Dynkin labels for the simplest F_4 irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

5.5.8 G_2

G_2 is an exceptional algebra of rank 2. Dynkin labels for the simplest G_2 representations are presented in table 5.10.

Dynkin labels	Dimension
$(1,0)$	7
$(0,1)$	14
$(1,1)$	64
$(2,0)$	27
$(0,2)$	77
$(2,1)$	189
$(1,2)$	286
$(2,2)$	729

Table 5.10 – Dynkin labels for the simplest G_2 irreducible representations. Dimensions have been calculated with `MARTY` as shown in section 5.2.

CHAPTER 6

Automated calculations with MARTY

6.1 Introduction

Several of the main features of MARTY have been discussed through the presentation of symbolic calculations, quantum fields, models and group theory in chapters 2, 3, 4 and 5 respectively. This chapter will show how the combination of all these features allow MARTY to perform calculations for beyond the Standard Model theories. The theoretical quantities that MARTY can derive are amplitudes, squared amplitudes, and Wilson coefficients. All these calculations can be done at the tree-level or at the one-loop level.

Deriving the amplitude is always the first step as squared amplitudes and Wilson coefficient calculations are based on it. The transition amplitude between an initial state i and a final state f is generally noted $i\mathcal{M}(i \to f)$ and is a part of the so-called $S-$matrix element

$$\langle f|\,\hat{S}\,|i\rangle \equiv \langle f|i\rangle + \langle f|\,i\hat{\mathcal{T}}\,|i\rangle\,, \tag{6.1}$$

with

$$\langle f|\,i\hat{\mathcal{T}}\,|i\rangle \equiv i\mathcal{M}(i \to f)\cdot(2\pi)^4\delta^{(4)}\Big(\sum_k p_k\Big), \tag{6.2}$$

and $\sum_k p_k$ the sum of momenta entering the process that must vanish because of 4-momentum conservation. We are in general interested in $i \neq f$ and equation 6.1 then reads

$$\langle f|\,\hat{S}\,|i\rangle \equiv i\mathcal{M}(i \to f)\cdot(2\pi)^4\delta^{(4)}\Big(\sum_k p_k\Big). \tag{6.3}$$

The quantum mechanical transition amplitude $\langle f|\,\hat{S}\,|i\rangle$ squared is proportional to the transition probability from i to f, $P(i \to f)$, namely

$$|i\mathcal{M}(i \to f)|^2 \propto P(i \to f), \tag{6.4}$$

that we need to calculate in order to obtain experimental predictions from beyond the Standard Model scenarios. Figure 6.1 represents a $2 \to 3$ process in quantum field theory, i.e. two incoming particles as initial state i giving after the interaction, represented by the red bulb, a final state f composed of three particles. Fields Φ_{i_k/o_k} here are generic and

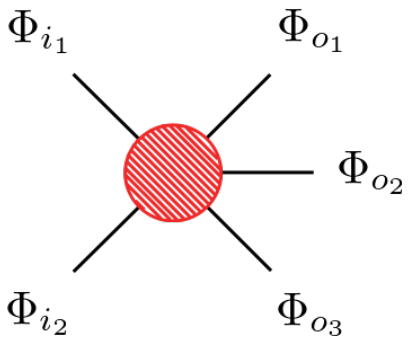


Figure 6.1 – Example of a process that can be calculated in quantum field theory. Two incoming particles (on the left) interact to give three, maybe different, particles (on the right). The interaction is represented by the red bulb. This diagram has been generated using GRAFED, see section 2.5 for more details.

can represent any kind of quantum field. The calculation of a transition amplitude means defining the initial state i and final state f of the process, and deriving the interaction strength from the Lagrangian.

After introducing library generation that is the main output format of MARTY, section 6.2 will present the building blocks of the amplitude calculation from the quantum field theory point of view, and sections 6.3, 6.4 and 6.5 will respectively introduce calculation procedures for amplitudes, squared amplitudes and Wilson coefficients.

Library generation

Once a theoretical calculation has been performed by MARTY the main output is under the form of C++ code automatically generated. The analytical results are in general too large to be interpreted explicitly, in particular when they are derived at one-loop by an automated software program. Mathematical expressions have therefore mostly a validation purpose and MARTY generates C++ numerical functions that allow us to evaluate these results given precise values for the model parameters. To illustrate this, let us use the ghost squared amplitude example presented later in equation 7.73:

$$|\mathcal{M}_g|^2 = \frac{3g_s^4}{32}\frac{s_{13}s_{14}}{s_{12}^2}, \tag{6.5}$$

with g_s the strong coupling constant and

$$s_{ij} \equiv p_i \cdot p_j, \tag{6.6}$$

which are kinematic factors depending on external momenta p_i. This is a scalar function (without free indices) that can be evaluated numerically to probe its behavior with respect to the model parameters. Such an expression will be translated into a C++ function by MARTY e.g. [1]

1. The real output is slightly different than the one presented in the following and has been modified to remove details that are not necessary for the argument presented here. The function corresponds the the squared amplitude of $c_g\bar{c}_g \to t\bar{t}$ in the SM (with c_g the gluon ghost) that will be presented in details in chapter 7.

```
complex_t M2_g(param_t const &param)
{
    const real_t g_s = param.g_s;
    const real_t s_12 = param.s_12;
    const real_t s_13 = param.s_13;
    const real_t s_14 = param.s_14;
    const complex_t IT_0 = s_13*s_14;
    const complex_t IT_1 = (-12)*IT_0;
    const complex_t IT_2 = pow(s_12, -1);
    const complex_t IT_3 = (complex_t{0, 1})*g_s;
    const complex_t IT_4 = IT_2*pow(IT_3, 2);
    const complex_t IT_5 = 0.5*IT_4;
    return (-0.03125)*IT_1*IT_5*conj(IT_5);
}
```

Several intermediate steps `IT_i` are defined by MARTY and the function returns a complex number which is the final result of the calculation. This number can then be evaluated to perform a phenomenological study of the model as presented extensively in chapter 7. The `param_t` object is the parameter structure that contains all parameters that must be user-defined. `real_t` and `complex_t` are type definitions for respectively real and complex numbers. We chose on purpose a very simple example to be displayed on only a few lines but in general expressions are much longer and very difficult to read analytically.

MARTY generates on demand comprehensive and ready-to-use C++ libraries containing calculation results such as the one presented above together with general utilities and an integrated tree-level spectrum generator for any given BSM model. More details are given in the dedicated chapter of the user manual [86].

6.2 Building blocks

In this section, the building blocks required from quantum field theory to build transition amplitudes are presented. Once the amplitude has been constructed it must be simplified, this will be discussed in section 6.3.

The LSZ formula

Before entering into more concrete considerations let us define the basis formula, the starting point of amplitude calculations in quantum field theory. It is called the **LSZ reduction formula** from its authors: Lehmann, Symanzik and Zimmermann [116]. For general fields $\{\Phi^i_{in}(P_i)\}_i$ in the initial state and $\{\Phi^j_{out}(Q_j)\}_j$ in the final state, the LSZ formula gives the expression of the corresponding S-matrix element, that we defined in

equation 6.3, for a theory with an interaction Lagrangian $\mathcal{L}_{\text{int}}$:

$$\begin{aligned} S\left(\{\Phi^i_{in}(P_i)\}_i \to \{\Phi^j_{out}(Q_j)\}_j\right) = \\ \prod_i \left(i\epsilon[\Phi^i_{in}; P_i] \cdot \int d^4X_i e^{-iP_i^\mu \cdot X_{i\mu}} \cdot \text{EOM}[\Phi^i_{in}; X_i]\right) \\ \cdot \prod_j \left(i\epsilon[\Phi^j_{out}; Q_j] \cdot \int d^4Y_j e^{iQ_j^\mu \cdot Y_{j\mu}} \cdot \text{EOM}[\Phi^j_{out}; Y_j]\right) \\ \cdot \left\langle \prod_i \Phi^i_{in}(X_i) \prod_j \Phi^j_{out}(X_j) e^{i\int d^4X \mathcal{L}_{\text{int}}} \right\rangle, \end{aligned} \tag{6.7}$$

where

- $\text{EOM}[\Phi; X]$ is the differential operator for the equation of motion for Φ at X.
- The integrated operators before the brackets project each particle on its on-shell state with the right 4-momentum P_i^μ or Q_j^μ.
- The brackets $\langle\rangle$ represent the vacuum expectation value of the fields' time-ordered product, taken in the interaction picture. It is evaluated using **Wick's theorem** [117].
- $\mathcal{L}_{\text{int}}$ is the interaction Lagrangian of the theory.
- $\epsilon[\Phi_i; P_i]$ corresponds to the spin tensor of the field Φ_i. It is equal to 1 for a scalar (spin 0) field, and is non trivial otherwise (see below for more details).
- The 4-momentum conservation implies that: $\sum_i P_i^\mu = \sum_j Q_j^\mu$.
- The hermitic conjugation of fields Φ is not specified in this equation.

Bosons' and fermions' free dynamics are governed respectively by the so-called **Klein-Gordon** and **Dirac** equations, [2] so that for a boson ϕ of mass m_ϕ and a fermion ψ of mass m_ψ:

$$\left(\Box_X + m_\phi^2\right)\phi(X) = 0, \tag{6.8}$$

$$\left(-i\not{\partial}_X + m_\psi\right)\psi(X) = 0, \tag{6.9}$$

with

$$\begin{aligned} \Box_X &\equiv \frac{\partial}{\partial X^\mu}\frac{\partial}{\partial X_\mu}, \\ \not{\partial}_X &\equiv \gamma^\mu \frac{\partial}{\partial X^\mu}, \end{aligned} \tag{6.10}$$

using the γ-matrices in Dirac space γ^μ. Then one can deduce the expression of $\text{EOM}[\Phi; X]$:

$$\text{EOM}[\phi; X] = \Box_X + m^2, \tag{6.11}$$

for a scalar of mass m and

$$\text{EOM}[\psi; X] = -i\not{\partial}_X + m, \tag{6.12}$$

2. Spin 1/2 particles are solutions of the Dirac equation and also the Klein-Gordon equation.

for a an incoming anti-fermion or an outgoing fermion of mass m and finally

$$\text{EOM}[\bar{\psi}; X] = \left(-i\not{\partial}_X - m\right)^T, \tag{6.13}$$

for a an outgoing anti-fermion or an incoming fermion of mass m.

Field insertions in the time-ordered product of equation 6.7 can be conjugated or not if the field is incoming or outgoing, particle or anti-particle. This is summarized in figure 6.2.

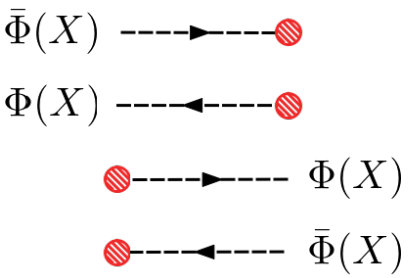


Figure 6.2 – Field insertions for a generic field Φ in the LSZ formula. Incoming fields are placed on the left and outgoing ones on the right. The rest of the process is represented by the red bulb. For bosons $\bar{\Phi}$ is the hermitic conjugate $\bar{\phi} \equiv \phi^\dagger$ and for spin $1/2$ fermions $\bar{\psi} = \psi^\dagger\gamma^0$. These diagrams have been generated using GRAFED, see section 2.5 for more details.

The Lagrangian's exponential can be expressed as a Taylor series and reads

$$e^{\int_X \mathcal{L}_{\text{int}}} = \sum_{n=0}^{+\infty} \frac{1}{n!} \left(\int_X \mathcal{L}_{\text{int}}\right)^n, \tag{6.14}$$

which can be truncated at a fixed order N in the limit of small coupling constants ($g \ll 1$) in the interaction Lagrangian, giving

$$\begin{aligned} e^{\int_X \mathcal{L}_{\text{int}}(X)} &\approx \sum_{n=0}^{N} \frac{1}{n!} \left(\int_X \mathcal{L}_{\text{int}}(X)\right)^n \\ &= 1 + \int_X \mathcal{L}_{\text{int}}(X) + \frac{1}{2}\int_{X_1,X_2} \mathcal{L}_{\text{int}}(X_1)\mathcal{L}_{\text{int}}(X_2) \\ &\quad + \ldots + \frac{1}{n!}\int_{X_1,\ldots,X_N} \mathcal{L}_{\text{int}}(X_1)\ldots\mathcal{L}_{\text{int}}(X_N). \end{aligned} \tag{6.15}$$

Once developed, the calculation can start following several steps:

- Replace $\epsilon[\Phi; P]$ in equation 6.7 by the appropriate **external leg** (see the next section).
- For all terms in equation 6.15, calculate the time-ordered product to find all non-zero field arrangements contracting pairs of fields, using Wick's theorem. These non-trivial contributions correspond to the so-called **Feynman diagrams**.
- Replace the contracted pairs in the time-ordered product by the appropriate **propagators** (see the next section).

The exact algebraic steps to simplify all the integrals introduced by the LSZ formula will not be detailed here as they are quite involved. One can have more details about this procedure, that is automated in MARTY, in [29].

6.2.1 Propagators and external fields

Without considering interactions for the moment, two main ingredients are required to apply the LSZ formula in 6.7: The Spin tensors $\epsilon[\Phi; P]$ and the field contractions in the time-ordered product, the so-called **propagators** describing the dynamics of free particles.

Scalar fields (spin 0)

The scalar field has no spin by definition, its polarization tensor is then trivial, namely for a scalar $\phi(X)$

$$\epsilon[\phi; P] = 1. \tag{6.16}$$

The propagator for the scalar field reads

$$\langle\, \overbrace{\phi(X)\phi^*(Y)}\, \rangle = \int \frac{d^4q}{(2\pi)^4} \left(\frac{i}{q^2 - m_\phi^2 + i\varepsilon} \cdot e^{iq\cdot(Y-X)} \right), \tag{6.17}$$

with m_ϕ the mass of $\phi(X)$ and ε an infinitesimal parameter to avoid the pole in $q^2 = m_\phi^2$.

Vector fields (spin 1)

A vector field $A^\mu(X)$ has a non trivial spin tensor

$$\begin{aligned} \epsilon[A^\mu; P] &= -\epsilon^{*\mu}_\lambda(P), \\ \epsilon[A^{\dagger\mu}; P] &= -\epsilon^{\mu}_\lambda(P), \end{aligned} \tag{6.18}$$

with a λ index that runs over its spin degrees of freedom (*dof*) [3]. This is represented in figure 6.3.

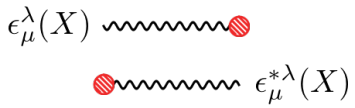


Figure 6.3 – External legs for a vector boson as a particle in the LSZ formula. Minus signs appearing in equation 6.18 are omitted to avoid any confusion with Feynman rules considering that they have no impact on physical observables anyway. For anti-particles, ϵ and ϵ^* are swapped. These diagrams have been generated using GRAFED, see section 2.5 for more details.

The propagator of a massless vector field reads

$$\langle\, \overbrace{A^\mu(X)A^{\nu*}(Y)}\, \rangle = \int \frac{d^4q}{(2\pi)^4} \frac{-i\left(g^{\mu\nu} - (1-\xi)\dfrac{q^\mu q^\nu}{q^2}\right)}{q^2 + i\varepsilon} \cdot e^{iq\cdot(Y-X)}, \tag{6.19}$$

with $g^{\mu\nu}$ the Minkowski metric, and ξ a gauge fixing parameter.

3. A vector field has either 2 dof if it is massless (spin projection ± 1) or 3 dof if it has a non zero mass (spin projection $0, \pm 1$).

The propagator of a massive vector field is the following:

$$\langle \overbrace{A^{\mu}(X)A^{\nu *}(Y)} \rangle = \int \frac{d^4q}{(2\pi)^4} \frac{-i\left(g^{\mu\nu} - (1-\xi)\frac{q^{\mu}q^{\nu}}{q^2 - \xi m_A^2}\right)}{q^2 - m_A^2 + i\varepsilon} \cdot e^{iq\cdot(Y-X)}, \tag{6.20}$$

with m_A the mass of the field $A^{\mu}(X)$.

Fermion field (spin $1/2$)

There are several realizations of spin $1/2$ particles and the Dirac fermions are used in the following. Weyl and Majorana fermion identities can be obtained from the ones for Dirac particles using respectively projection relations

$$\psi_{L/R} = P_{L/R}\psi, \tag{6.21}$$

and the conjugation relation for Majorana fermions ψ_M

$$C\bar{\psi}_M^T = \psi_M. \tag{6.22}$$

The Dirac field $\psi(X)_{\alpha}$ is a 4-component spinor with 4 degrees of freedom: particle/anti-particle and spin $\pm 1/2$. They are split in two two-component spin tensors, one for the particle ($u_{\sigma}^{\alpha}(P)$) and one for the anti-particle ($v_{\sigma}^{\alpha}(P)$), with σ the index running over the two possible spin projections $\pm 1/2$. The LSZ elements are then defined as

$$\begin{aligned} \epsilon[\psi_{\alpha}; P] &= \bar{u}_{\alpha}^{\sigma}(P), \\ \epsilon[\bar{\psi}_{\alpha}; P] &= u_{\alpha}^{\sigma}(P), \end{aligned} \tag{6.23}$$

for a particle, and

$$\begin{aligned} \epsilon[\psi_{\alpha}; P] &= \bar{v}_{\alpha}^{\sigma}(P), \\ \epsilon[\bar{\psi}_{\alpha}; P] &= v_{\alpha}^{\sigma}(P), \end{aligned} \tag{6.24}$$

for an anti-particle. This is summarized in figure 6.4.

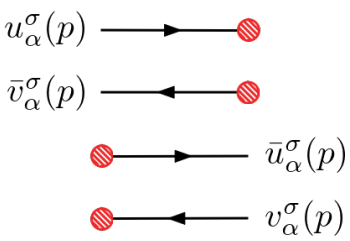


Figure 6.4 – External legs for a Dirac fermion in the LSZ formula. These diagrams have been generated using GRAFED, see section 2.5 for more details.

The propagator of a massive Dirac field is the following:

$$\langle \overbrace{\psi_{\alpha}(X)\bar{\psi}_{\beta}(Y)} \rangle = \int \frac{d^4q}{(2\pi)^4} \frac{i\left(\not{q} + m_{\psi}\right)_{\alpha\beta}}{q^2 - m_{\psi}^2 + i\varepsilon} \cdot e^{iq\cdot(Y-X)}, \tag{6.25}$$

with m_ψ the mass of the field $\psi_\alpha(X)$ that can be set to zero straight-forwardly. As fermions anti-commute with each other the inverted propagator simply reads

$$\begin{aligned}\langle \overbrace{\overline{\psi_\alpha}(X)\psi_\beta(Y)}\rangle = -\langle \overbrace{\psi_\beta(Y)\overline{\psi_\alpha}(X)}\rangle &= -\int \frac{d^4q}{(2\pi)^4}\frac{i\left(\not{q}+m_\psi\right)_{\beta\alpha}}{q^2-m_\psi^2+i\varepsilon}\cdot e^{-iq\cdot(Y-X)} \\ &= \int \frac{d^4q}{(2\pi)^4}\frac{i\left(\not{q}-m_\psi\right)_{\beta\alpha}}{q^2-m_\psi^2+i\varepsilon}\cdot e^{iq\cdot(Y-X)}.\end{aligned} \tag{6.26}$$

6.2.2 Feynman rules

Once the LSZ formula introduced in equation 6.7 and the associated algebraic manipulations have been implemented, amplitude calculations can be expressed in terms of Feynman rules directly in momentum space i.e. without any integration (except for calculations at the loop level). Once all the diagrams for a given process are known, or equivalently all non-zero contractions in the time-ordered product of the LSZ formula, each diagram can be calculated using well-known shortcuts, the **Feynman rules**. They include external legs like those presented in figures 6.3 and 6.4, symmetry factors, momentum conservation rules, momentum integration in loop diagrams, and more importantly [4] simple expressions for interaction vertices preventing us to apply explicitly the LSZ formula.

MARTY uses both methods, the LSZ reduction formula and Feynman rules. First, Feynman rules are calculated from the Lagrangian by applying the LSZ formula and performing all the required algebra. The LSZ reduction formula is still used at this point to derive symmetry factors and momentum conservation at each vertex, but the calculation is faster and the result is more compact. This allows MARTY to provide a fully general calculation procedure while using relevant shortcuts to make it more efficient. As Feynman rules represent an important interest for users in terms of physics and code validation, we present them through typical use cases in a MARTY program.

Get Feynman rules in MARTY

Feynman rules are stored in the class `FeynmanRule`. This class is mostly encapsulated by the `Model` class. This means that a user will not have to manipulate explicitly the vertices in a standard use case. Feynman rules will be automatically computed before a calculation if they have not already been derived. The procedure to calculate the Feynman rules of a model and retrieve them is presented in sample code 37. There is a simple interface for the user to get the information about the different vertices, being the Feynman diagrams (using *GRAFED*) or the symbolic expressions. One can compute Feynman rules at any time during model building.

4. As stated here Feynman rules are more than interaction vertices but in the following we will not consider the other rules as they do not represent a challenge in automated BSM calculations.

Sample code 37: Feynman rules

Launch the calculation of Feynman rules

```
model.computeFeynmanRules();
```

Get Feynman rules

```
// No need to ask the explicit computation when obtaining the rules.
auto rules = model.getFeynmanRules();
```

Display the rules

```
Display(rules); // Displays expressions in standard output
Show(rules); // Shows Feynman diagrams for rules with GRAFED
```

Note The `auto` keyword deduces the type `vector<FeynmanRule>` here, i.e. a list of rules.

Note The computation of Feynman rules can be done several times if needed. When getting Feynman rules, they will be calculated only if it is not already done, and then simply returned.

Read Feynman rules from MARTY

It is important to be able for a user to read Feynman rules, and check that they are correct. When doing model building, this is the most important part. An incorrect vertex coming from a misunderstanding of different conventions in the model will end up in wrong results.

A Feynman rule is a set of fields that enter the Wick theorem, with which other fields will contract. Here is an example of a Feynman rule for a fermion-photon interaction:

```
(0) : Rule for A_mu(p_1) psi_b(p_2)^(*) psi_a(p_3) :
   -i*e*gamma_{mu,a,b}
```

One can see the field content of the first line, with corresponding indices and momenta. The expression of the Feynman rule is given in the line below. We see the coupling, the γ-matrix, but more importantly one may want to know which fermion is incoming, `psi_b(p_2)^(*)` or `psi_a(p_3)`. A vertex is not defined with the fields it contains but the ones that can be contracted with it. This principle is explained in figure 6.5. One can now be certain that ψ^* in the vertex corresponds to the incoming fermion, and must then be associated with the second index `b` of the γ-matrix.

Let us now consider other Feynman rules as examples such as a scalar QED theory, with a $U(1)$ gauge. There are two vertices, namely a 3-vertex with a derivative of the scalar ϕ and a 4-vertex. The Feynman rules are the following

```
(0) : Rule for A_mu(p_1) phi(p_2) phi(p_3)^(*) :
   i*e*(p_1_mu + 2*p_2_mu)

(1) : Rule for A_mu(p_1) A_nu(p_2) phi(p_3) phi(p_4)^(*) :
   2*i*e^2*g_{mu,nu}
```

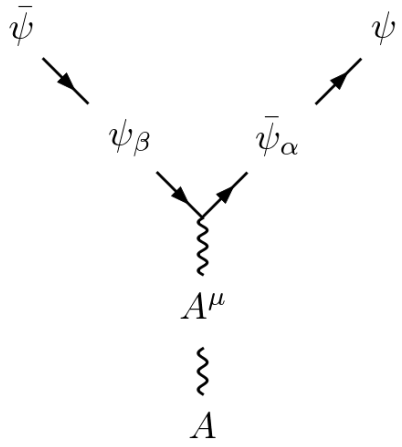


Figure 6.5 – Vertex for a Feynman rule, with a fermion-photon interaction example. The fields in the vertex are shown in the middle, and fields using the rule (contracting with it) are on the outside. The outside fields are the ones defining a Feynman rule, that MARTY displays as the content of a vertex. This diagram has been generated using GRAFED, see section 2.5 for more details.

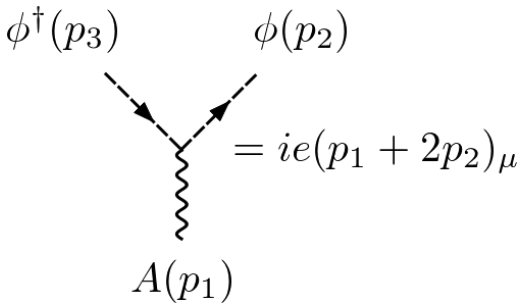


Figure 6.6 – 3-vertex rule in Scalar QED. The diagram presented here does not correspond to the inner vertex but to MARTY's Feynman rule, as explained in figure 6.5. All momenta in Feynman rules are directed towards the vertex. This diagram has been generated using GRAFED, see section 2.5 for more details.

One can see the momentum dependence that arises in the 3-vertex. Figure 6.6 shows in more details this dependence. Momenta in Feynman rules are always considered incoming by default.

Let us complete the presentation of Feynman rules through a QCD example. There is a fermion-gluon interaction similar to the QED vertex with a $SU(3)$ generator, together with a 4-gluon vertex introducing $SU(3)$ structure constants f^{ABC}. The two rules are the following in MARTY:

```
(4) : Rule for G_{h,tau}(p_1) X_{b,eps}(p_2)^(*) X_{a,del}(p_3) :
    i*g*T_{h,a,b}*gamma_{tau,del,eps}

(5) : Rule for G_{a,+tau}(p_1) G_{b,+mu}(p_2) G_{c,nu}(p_3) G_{d,rho}(p_4) :
    -i*g^2*(f_{a,d,%h}*f_{b,c,%h}*g_{+mu,+tau}*g_{rho,nu}
           + f_{a,c,%h}*f_{b,d,%h}*g_{+mu,+tau}*g_{rho,nu}
           - f_{a,c,%h}*f_{b,d,%h}*delta_{+mu,nu}*delta_{rho,+tau}
           - f_{a,b,%h}*f_{c,d,%h}*delta_{+mu,nu}*delta_{rho,+tau}
           - f_{a,d,%h}*f_{b,c,%h}*delta_{+mu,rho}*delta_{+tau,nu}
```

```
+ f_{a,b,%h}*f_{c,d,%h}*delta_{+mu,rho}*delta_{+tau,nu})
```

The 4-vertex reads (renaming indices to a vertex $G^{A\mu}(p_1)G^{B\nu}(p_2)G^{C\rho}(p_3)G^{D\sigma}(p_4)$)

$$\begin{aligned} -ig^2\Big(& f^{adh}f^{bch}g^{\mu\nu}g^{\rho\sigma} + f^{ach}f^{bdh}g^{\mu\nu}g^{\rho\sigma} \\ & - f^{ach}f^{bdh}g^{\mu\sigma}g^{\nu\rho} - f^{abh}f^{cdh}g^{\mu\sigma}g^{\nu\rho} \\ & - f^{adh}f^{bch}g^{\mu\rho}g^{\nu\sigma} + f^{abh}f^{cdh}g^{\mu\rho}g^{\nu\sigma}\Big), \end{aligned} \tag{6.27}$$

which can be factored to recover the well-known rule [29]

$$\begin{aligned} -ig^2\Big(& f^{abh}f^{cdh}\left(g^{\mu\rho}g^{\nu\sigma} - g^{\mu\sigma}g^{\nu\rho}\right) \\ & + f^{ach}f^{bdh}\left(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\sigma}g^{\nu\rho}\right) \\ & + f^{adh}f^{bch}\left(g^{\mu\nu}g^{\rho\sigma} - g^{\mu\rho}g^{\nu\sigma}\right)\Big). \end{aligned} \tag{6.28}$$

6.3 Amplitudes

The calculation of transition amplitudes is the basis of theoretical predictions. They are not directly observables but are required to calculate squared amplitudes (see section 6.4) and Wilson coefficients (see section 6.5). In this section the main procedures required to calculate amplitudes, and the way to obtain such theoretical quantities with MARTY are presented.

6.3.1 Finding diagrams

Finding all the diagrams for a given process can be done in several ways. FeynArts [73] for example uses a hard-coded set of possible topologies for tree-level and one-loop processes with a given number of external legs. Once done, it tries to fit particles to each leg of each topology. MARTY uses a more straight-forward way (although not particularly simpler) by explicitly using the Wick theorem and trying all possible Lagrangian interactions for each vertex in the diagram. [5] This can be longer but has the advantage of being fully general, in particular not limited to the one-loop order or to a maximum number of external legs.

The algorithm finding all the diagrams is highly optimized to avoid deriving multiple times the same diagram and search only for distinct contractions, keeping track of the degeneracy factor. Let us consider the MSSM example that contains about 10^4 interaction terms. A 1-loop amplitude with 3 external particles requires to develop to $\mathcal{L}^3$ in perturbation theory i.e. to test a priori $(10^4)^3 = 10^{12}$ terms. This cannot be done in a reasonable amount of time on a standard computer. The algorithm must therefore benefit from the redundancy in the different terms to reduce the number of tests and find only a small

5. The maximum number of vertices is determined by the order of expansion.

number of independent diagrams. In the above example, there are typically only 10^2 to 10^3 independent diagrams starting from the initial 10^{12} possibilities.

The Wick theorem method results in a set of possible diagrams, each of which is associated to an initial mathematical expression that relies on external legs, propagators and Feynman rules for vertices. Each diagram comes with a symmetry factor that is automatically derived by applying Wick's theorem, and a possible sign coming from Grassmann ordering (fermions and ghosts) that is also taken care of.

6.3.2 Gauge fixing

Gauge fixing can take place for any vector boson, independently of a gauge group. Taking the most general case of a massive vector boson A^μ of mass M with a Goldstone boson ϕ and a ghost c, one can write down the propagators of these different particles depending on the gauge fixing parameter ξ [29], as presented in figure 6.7. Correspondence between the different gauges available in MARTY and the ξ parameter is detailed in table 6.1.

$$A^\mu \sim\sim A^\nu = -i\frac{g^{\mu\nu} - (1-\xi)\frac{p^\mu p^\nu}{p^2-\xi M^2}}{p^2 - M^2},$$
$$\phi \text{ ----▶---- } \phi = \frac{i}{p^2 - \xi M^2},$$
$$c \text{ •••▶••• } c = \frac{i}{p^2 - \xi M^2}.$$

Figure 6.7 – Propagators for a vector boson and its Goldstone and ghost bosons depending on the gauge fixing parameter ξ. These diagrams have been generated using GRAFED, see section 2.5 for more details.

Name	Parameter value	Name in MARTY
Feynman 't Hooft	$\xi = 1$	`gauge::Feynman`
Lorenz	$\xi = 0$	`gauge::Lorenz`
Unitary	$\xi = \infty$	`gauge::Unitary`
$\mathcal{R}_\xi$	ξ	`gauge::NotDefined`

Table 6.1 – List of the different gauges that one can choose for a particular vector boson in MARTY. The $\mathcal{R}_\xi$ gauge lets an explicit ξ dependence in the calculation that should cancel in physical observables. The unitary gauge ($\xi = \infty$) is possible only for massive vector bosons.

One can see that setting the gauge choice for a vector boson will simply modify its propagator, and masses of the associated ghost and Goldstone bosons. As physical results must be gauge invariant, doing a calculation in one gauge or another should give the same result. In particular, a calculation in the $\mathcal{R}_\xi$ gauge will be expressed as a function of ξ but should not depend on it. In the special case of the unitary gauge, the propagator or the vector becomes

$$-i\frac{g^{\mu\nu} - \frac{p^\mu p^\nu}{M^2}}{p^2 - M^2}, \tag{6.29}$$

and the ghost and Goldstone bosons acquire an infinite mass $\sqrt{\xi}M \to \infty$ and decouple from the theory. In particular, MARTY simply disables these scalars in diagrams for the unitary gauge. The latter gauge must not be used for massless vector bosons as the limit $\xi \to \infty$ is ill-defined. The procedure to fix a gauge choice for a vector boson is presented in sample code 38.

Sample code 38: Gauge fixing

Setting the unitary gauge for the W boson and Feynman gauge for the photon A in the Standard Model

```
model.setGaugeChoice("W", gauge::Unitary);
model.setGaugeChoice("A", gauge::Feynman);
```

Note The default gauge choice in MARTY is `gauge::Feynman` with $\xi = 1$.

For now Goldstone - ghost interactions as presented in [118] are not defined in MARTY's built-in models. They arise after the SM gauge symmetry breaking and are not written in general. These gauge fixing terms, depending on ξ, can be user-defined in the Lagrangian. When using the corresponding vertices, ξ will be replaced by its value before the simplification procedures. For processes using such interactions, the corresponding interaction terms must be added as demonstrated in section 4.5.

6.3.3 Initial amplitude expression

The amplitude $i\mathcal{M}(i \to f)$ of a transition from an initial state i to a final state f is defined from the S-matrix element

$$\begin{aligned}\left\langle f|\hat{S}|i\right\rangle &\equiv \left\langle f|(1+i\hat{\mathcal{T}})|i\right\rangle \\ &= \langle f|i\rangle + (2\pi)^4\delta^{(4)}\left(\sum_i p_i - \sum_f p_f\right)\cdot i\mathcal{M}(i\to f),\end{aligned} \tag{6.30}$$

with p_i incoming and p_f outgoing momenta. The quantity that MARTY calculates contains the $(2\pi)^4\delta^{(4)}\left(\sum_i p_i - \sum_f p_f\right)$ factor coming from the general momentum conservation but it is removed from the result to obtain $i\mathcal{M}(i \to f)$. The term proportional to the identity is not relevant in quantum field theory calculations as we are interested in processes where $|i\rangle \neq |f\rangle$. Feynman rules are properly inserted at each interaction vertex, and momentum conversation in the diagram is calculated from the LSZ formula that introduces $V_e + V_i + E$ integrals, one for each external vertex (V_e), internal vertex (V_i) and edge (E) in the diagram. These integrals simplify following the rules

$$\int d^4X e^{iX(p-q)} = (2\pi)^4\delta^{(4)}(p-q), \tag{6.31}$$

$$\int d^4q\, \delta^{(4)}(p-q)f(q) = f(p). \tag{6.32}$$

There is always one momentum conservation property that is not integrated at the end of the calculation i.e. a delta function containing

$$\sum_i p_i - \sum_f p_f = 0. \tag{6.33}$$

The number of edges in a diagram reads

$$E = V - 1 + N_L = V_I + V_E - 1 + N_L, \tag{6.34}$$

with $V = V_I + V_E$ the total number of vertices, and N_L the number of loops. A tree with V vertices has $V - 1$ edges, and each additional connection will add a loop to the diagram. The LSZ formula introduces

$$V = V_E + V_I = V_E + V_I - 1 + 1 \tag{6.35}$$

position-space integrals. $V_E + V_I - 1$ of them simplify one by one the momentum integrals, and the last one (the $+1$ left) gives the momentum conservation on the whole diagram. With initially E momentum space integrals, we are left with

$$E - (V_E + V_I - 1) = V_I + V_E - 1 + N_L - (V_E + V_I - 1) = N_L \tag{6.36}$$

momentum integrals in the final expression of the amplitude. The latter must then be simplified, as explained in section 6.3.4.

6.3.4 Simplification of expressions

Once the amplitude has been constructed using Feynman rules, it must be simplified in several ways to obtain the final result. Without simplification, no further numerical evaluation would be possible because of the initial expression size. The different simplification procedures presented in the following are the reasons why we need automated symbolic calculations.

Group theory

Group generators define the algebra through their commutation relation

$$\left[T^A, T^B\right] \equiv i f^{ABC} T^C, \tag{6.37}$$

with f^{ABC} the algebra's structure constants. Group theory simplifications include simple tensor contractions and trace of generators such as

$$Tr(T^{A_1} T^{A_2} \cdots T^{A_N}). \tag{6.38}$$

Traces can be calculated in all representations of semi-simple Lie groups. They are decomposed in a combination of invariant fully-symmetric tensors $d^{A_1 \cdots A_N}$ and f^{ABC} contributions. There are additional simplification identities in the defining representation of these groups [115, 119] such as

$$T^A_{ij} T^A_{kl} = \frac{1}{2}\left(\delta_{il}\delta_{jk} - \frac{1}{N}\delta_{ij}\delta_{kl}\right) \tag{6.39}$$

in $SU(N)$.

Group theory simplifications are mostly used at the squared amplitude level, more details are given in section 6.4.

Diracology

Fermions are ubiquitous in particle physics calculations and in particular spin $1/2$ particles. The identities presented in the following can be generalized to higher spins such as spin $3/2$ particles but as MARTY does not support them yet we only discuss relations for spin $1/2$ particles.

Transition amplitudes are always expressed with fermion bilinears:

$$\bar{\psi}\Gamma\xi, \tag{6.40}$$

with $\bar{\psi}$ and ξ two fermions and Γ some combination of γ-matrices. Most of the processes involving fermions that are calculated for a phenomenological purpose will have one or two bilinears (two or four external fermions). The γ-matrix denomination includes here γ^μ and all objects built from it:

$$\gamma^5 \equiv i\gamma^0\gamma^1\gamma^2\gamma^3, \tag{6.41}$$

$$P_L \equiv \frac{1-\gamma^5}{2}, \tag{6.42}$$

$$P_R \equiv \frac{1+\gamma^5}{2}, \tag{6.43}$$

$$\sigma^{\mu\nu} \equiv \frac{i}{2}\left[\gamma^\mu, \gamma^\nu\right], \tag{6.44}$$

$$C \equiv -i\gamma^0\gamma^2. \tag{6.45}$$

γ^μ are generators of the Clifford algebra respecting

$$\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}. \tag{6.46}$$

They are built from the basis of 2×2 hermitian matrices composed of the identity and

the three Pauli matrices and read

$$\begin{aligned}
\gamma^0 &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \\
\gamma^1 &= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \\
\gamma^2 &= \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}, \\
\gamma^3 &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},
\end{aligned} \tag{6.47}$$

where we expressed γ^μ in the Weyl realization acting on a vector space with Dirac fermions ψ_D defined from chiral particles $\psi_{L/R}$ (left/right):

$$\psi_D \equiv \psi_L \oplus \psi_R = \begin{pmatrix} \psi_{L1} \\ \psi_{L2} \\ \psi_{R1} \\ \psi_{R2} \end{pmatrix}. \tag{6.48}$$

In this basis the chirality operator γ^5 is diagonal:

$$\gamma^5 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.49}$$

and projectors P_L and P_R indeed respect together projection identities, namely

$$\begin{aligned}
P_L + P_R &= 1, \\
P_L^2 &= P_L, \\
P_R^2 &= P_R, \\
P_L P_R = P_R P_L &= 0.
\end{aligned} \tag{6.50}$$

One can derive new properties from the definitions above, e.g. in a D-dimensional space-time

$$\begin{aligned}
\gamma^\mu \gamma_\mu &= D, \\
\left\{\gamma^\mu, \gamma^5\right\} &= 0, \\
\gamma^\mu P_L &= P_R \gamma^\mu.
\end{aligned} \tag{6.51}$$

An automated computer program must therefore apply all the relations above to fully simplify the results. At the tree level calculations are often trivial but at one-loop one has to implement general simplifications for any combinations of matrices Γ arising in amplitudes.

Traces of γ-matrices arise in fermion loops and squared amplitudes and can be calculated recursively from the two most elementary relations

$$\mathrm{Tr}\left(\gamma^\mu\gamma^\nu\right) = 4g^{\mu\nu}, \tag{6.52}$$

$$\mathrm{Tr}\left(\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma\gamma^5\right) = -4i\epsilon^{\mu\nu\rho\sigma}, \tag{6.53}$$

with $\epsilon^{\mu\nu\rho\sigma}$ the fully anti-symmetric symbol defined by $\epsilon^{0123} = 1$. For example, the trace of $2n$ matrices without γ^5 is expressed as a function of traces with $2n-2$ matrices:

$$\mathrm{Tr}\left(\gamma^{\mu_1}\gamma^{\mu_2}\ldots\gamma^{\mu_{2n}}\right) = \sum_{i=2}^{2n}(-1)^i g^{\mu_1\mu_i}\,\mathrm{Tr}\left(\gamma^{\mu_2}\ldots\hat{\gamma}^{\mu_i}\ldots\gamma^{\mu_{2n}}\right), \tag{6.54}$$

where $\hat{\gamma}$ means that the corresponding matrix is removed from the trace. Similarly, one can derive the chiral trace

$$\begin{aligned}\mathrm{Tr}\left(\gamma^{\mu_1}\gamma^{\mu_2}\ldots\gamma^{\mu_{2n}}\gamma^5\right) = \sum_{i=1}^{2n-1}(-1)^{\lfloor\frac{i-1}{2}\rfloor}\sum_{j=i+1}^{2n}(-1)^{i+j+1}g^{\mu_i\mu_j} \\ \cdot\,\mathrm{Tr}\left(\gamma^{\mu_1}\ldots\hat{\gamma}^{\mu_i}\ldots\hat{\gamma}^{\mu_j}\ldots\gamma^{\mu_{2n}}\gamma^5\right).\end{aligned} \tag{6.55}$$

The trace of any odd number of γ-matrices vanishes and using $\left(\gamma^5\right)^2 = 1$ and equation 6.51 one can prove that there is no independent trace with more that one γ^5. Traces with projectors $P_{L/R}$ and $\sigma^{\mu\nu}$ can be trivially deduced.

The conjugation matrix

$$C \equiv -i\gamma^0\gamma^2 \tag{6.56}$$

also appears in amplitudes due to Majorana fermions or fermion-number violating interactions, and must be simplified using its properties e.g.

$$C^2 = -1, \tag{6.57}$$

$$C\gamma^\mu = -\gamma^{\mu T}C, \tag{6.58}$$

$$v = C\bar{u}^T. \tag{6.59}$$

In particular, the matrix C must always cancel out in simplifications and if it appears in a trace it means that the calculation is ill-defined. More details can be found in section 4.6.

To fully simplify transition amplitudes one finally has to apply the Dirac equation, namely on on-shell spinors with momentum p:

$$\not{p}u(p) = mu(p), \tag{6.60}$$

$$\not{p}v(p) = -mv(p), \tag{6.61}$$

with $u(p)$ and $v(p)$ the spinors of fermions and anti-fermions respectively (see section 6.2.1) and

$$\not{p} \equiv \gamma^\mu p_\mu. \tag{6.62}$$

One-loop calculation

As we said in section 6.3.3, a one-loop calculation requires the evaluation of one momentum integral such as

$$I = \int \frac{d^4q}{i\pi^2} \frac{\prod_{i=1}^{n} q^{\mu_i}}{\prod_{j=0}^{m-1}((q-p_j)^2 - m_j^2)} \tag{6.63}$$

for the $m-$point function of rank n, with $p_0 = 0$, $\{p_j\}_{j\geq 1}$ combinations of external momenta and m_j the masses of the particles in the loop. m is therefore the number of propagators and n, the rank, the number of momenta in the numerator. Letters are associated to n-point functions, starting with A for the 1-point function, up to E for the 5-point function. Such integrals can be decomposed in different Lorentz structure. Considering for example the 3-point function of rank 2

$$C^{\mu\nu} \equiv \int \frac{d^4q}{i\pi^2} \frac{q^\mu q^\nu}{(q^2 - m_0^2)((q-p_1)^2 - m_1^2)((q-p_2)^2 - m_2^2)}, \tag{6.64}$$

one can write without loss of generality

$$\begin{aligned} C^{\mu\nu} = {} & C_{00}(p_i, m_j) g^{\mu\nu} + C_{11}(p_i, m_j) p_1^\mu p_1^\nu + C_{22}(p_i, m_j) p_2^\mu p_2^\nu \\ & + C_{12}(p_i, m_j)(p_1^\mu p_2^\nu + p_1^\nu p_2^\mu). \end{aligned} \tag{6.65}$$

The decomposition is done by MARTY and factors C_{ij} are numerical functions implemented in the Fortran / C library LoopTools [73] that are used when results are evaluated numerically.

The scalar factors coming from one-loop integrals can have a divergent part that is regularized by taking the space-time dimension $D = 4 - 2\epsilon$. Integrals then take the general form

$$I \equiv \frac{a}{\epsilon} + b + \mathcal{O}(\epsilon). \tag{6.66}$$

Factors of D coming from Minkowski index contractions must therefore be kept to determine the local terms they generate when multiplied by a divergent integral [120, 121]. For the scalar 1-point function for example, one can derive the finite part of $DA_0(m^2)$

$$\text{Finite}(DA_0(m^2)) = \text{Finite}((4-2\epsilon)A_0(m^2)) = -2m^2 + 4 \cdot \text{Finite}(A_0(m^2)). \tag{6.67}$$

MARTY automatically adds local terms when necessary before replacing D by 4 in amplitudes.

One loop calculations are for now limited to

- At most 1-point function of rank 2, $A^{\mu\nu}$.
- At most 2-point function of rank 3, $B^{\mu\nu\rho}$.
- At most 3-point function of rank 4, $C^{\mu\nu\rho\sigma}$.
- At most 4-point function of rank 5, $D^{\mu\nu\rho\sigma\lambda}$.
- At most 5-point function of rank 4, $E^{\mu\nu\rho\sigma}$.

- Propagators with denominators of the type $\frac{1}{p^2 - X(m)}$. Note that one can absorb in general a factor in front of p^2 in the numerator.

The rank limitations [6] are due to the integrals LoopTools provides. They could however be relaxed in the future as we could implement reduction formulas for higher-rank integrals, in particular Passarino Veltman reductions [122, 123]. It can be difficult to adapt the procedure to generalized propagators as all standard calculations are based on master integrals that are provided by libraries like LoopTools. One can still do tree-level calculations in models relying on such propagators and as they are mostly exotic theories (Lorentz violating models for example), tree-level quantities are usually sufficient for phenomenological purpose.

Expansion, index contraction

In order to apply simplifications such group theory identities, diracology or equations of motions, one has to expand the expression at many places. For examples, the following trace

$$\mathrm{Tr}\Big((\not{p} + m)(\not{p} + m)\Big), \tag{6.68}$$

must be expanded into

$$\mathrm{Tr}\Big(\not{p}^2\Big) + 2m\,\mathrm{Tr}\Big(\not{p}\Big) + m^2\,\mathrm{Tr}(1) = 4p^2 + 0 + 4m^2, \tag{6.69}$$

using trace identities. For one expansion with a few terms there is no difficulty. However, most of the required simplifications in amplitudes imply expansion and generate again more terms to expand. This is in particular the case for γ-matrix traces in equations 6.54 and 6.55 that require to expand the amplitude to recognize γ-matrix products and generate sums of $g^{\mu\nu}$ that must be further expanded to simplify the result. At the one-loop level one has to expand propagators to recognize integrals of the form of equation 6.64, and the decomposition procedure such as the one presented in equation 6.65 generates again new terms.

We only presented a few examples above but there are in general many reasons to expand the amplitude when it must be fully simplified. As the total number of terms in the result grows exponentially with the number of different objects to expand, a naive procedure cannot be used and more care must be given to this problem. The basic idea is to always keep manageable expressions in terms of memory and time, otherwise the calculation cannot be performed or the result becomes unusable. Some elements of implementation are given in section 6.4 as this issue is more important in squared amplitude calculations.

6.3.5 The procedure using MARTY

External legs, that we discussed from a theoretical point of view in section 6.2.1, are the only pieces of information a user has to give beside the order of expansion. From external legs, MARTY finds all possible diagrams as presented in figure 6.8.

6. Rank limitations can also affect some gauges because a vector boson in unitary or Lorenz gauge introduces additional momenta in the numerators and denominators of propagators.

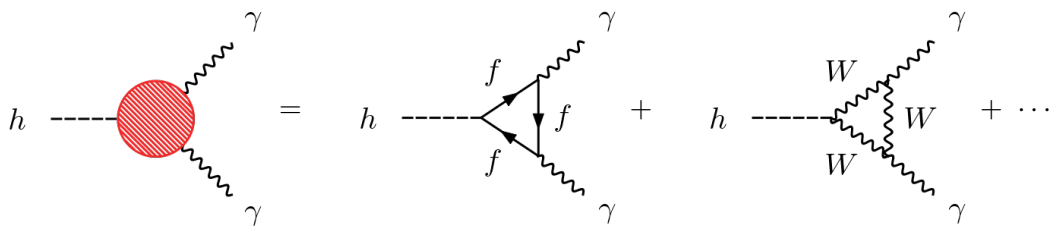


Figure 6.8 – Different 1-loop Feynman diagrams possible for the $h \to \gamma\gamma$ process in the SM. There are for example fermion and W-boson triangles. These diagrams have been generated using GRAFED, see section 2.5 for more details.

An external leg carries four pieces of information:

- The underlying quantum field.
- The direction, incoming or outgoing.
- The conjugation, particle or anti-particle. This is optional, by default external fields are particles.
- The equation of motion, on-shell or off-shell. This is optional, by default external fields are on-shell.

MARTY provides a simple interface to create field insertions that is shown in sample code 39. There are four interface functions (`Incoming()`, `Outgoing()`, `AntiPart()`, `OffShell()`) that can be combined to build the relevant field insertion, starting from a `Particle` object or the particle name.

Sample code 39: Field insertions

An incoming off-shell fermion `"psi"`

```
Incoming(OffShell("psi"));
// Or
// OffShell(Incoming("psi"));
```

An outgoing anti `"phi"`

```
Outgoing(AntiPart("psi"));
// Or
// AntiPart(Outgoing("psi"));
```

To give a list of insertions as function parameters, one can put them in curly braces `{}` (here for an electron self-energy calculation):

```
{Incoming(OffShell("e")), Outgoing(OffShell("e"))}
```

Knowing how to create external legs for MARTY, one can launch without effort an amplitude calculation as demonstrated in sample code 40.

Sample code 40: Amplitude calculation for $h \to ee$

Calculate the amplitude from the model

```
auto res = model.computeAmplitude(
    TreeLevel, // or OneLoop
    {Incoming("h"), Outgoing("e"), Outgoing(AntiPart("e"))}
);
```

Display the results

```
Display(res); // Prints symbolic result in standard output
Show(res); // Shows Feynman diagrams with GRAFED
```

Get the different terms of the amplitude

```
for (auto &diagram : res.getDiagrams()) {
    Expr &term = diagram.getExpression();
    // Do something to term
}
```

Note `auto` deduces the type `Amplitude` that contains all expressions and diagrams of the process.

For now library generation is only available for scalar quantities. An amplitude (with non-scalar external particles) has indices and cannot be given directly to the library generator presented in the user manual [86]. To generate C++ code corresponding to an amplitude, one must decompose it in Wilson coefficients, that are scalar quantities in front of different operators. For example the vacuum energy $i\Pi_{\mu\nu}(p)$ of a vector boson A can be decomposed in general as

$$i\Pi_{\mu\nu}(p) = \alpha(p^2, m^2) g_{\mu\nu} + \beta(p^2, m^2) p_\mu p_\nu, \tag{6.70}$$

which corresponds to the amplitude

$$i\mathcal{M} = \alpha(p^2, m^2) \left(\epsilon^*(p) \cdot \epsilon(p)\right) + \beta(p^2, m^2) \left(p.\epsilon^*(p)\right) \left(p \cdot \epsilon(p)\right), \tag{6.71}$$

with $\epsilon(p)$ polarization tensors for the external bosons. Calculating Wilson coefficients for this amplitude as explained in section 6.5 allows us to obtain $\alpha(p^2, m^2)$ and $\beta(p^2, m^2)$ that can be given to MARTY for code generation.

6.4 Squared Amplitudes

6.4.1 Generalities

Cross-sections are among the most straight-forward observables one can obtain from a theory as they are simply proportional to a number of events we measure in particle colliders. The instantaneous luminosity of a detector is expressed, in SI units, in $\mathrm{m}^{-2}{\cdot}\mathrm{s}^{-1}$ and is proportional to the number of particle collisions per unit of time in a given area. [7]

7. Although it is expressed in m^{-2}, the instantaneous luminosity is integrated over the whole collision surface.

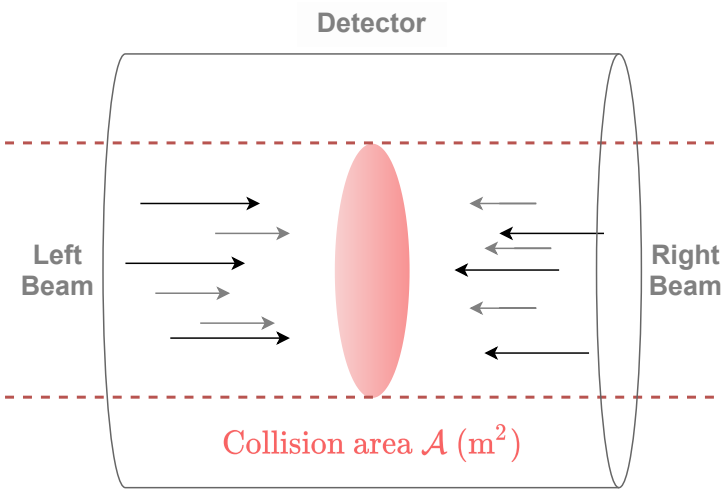


Figure 6.9 – Simple representation of the working principle of most particle colliders. Two particle beams are accelerated and meet at a precise position where a detector is placed to measure the products of collisions.

Figure 6.9 represents the working principle of particle colliders. There are in general two beams of particles colliding at a given position, the instantaneous luminosity is then the number of collisions in this area per unit of time. At the LHC for example the two proton [8] beams are accelerated in two separate tubes and cross each other only at precise positions, where the different detectors are placed.

The integrated luminosity represents the total number of collisions between an initial time t_i and a final time t_f, namely

$$\mathcal{L}_{\text{int}} = \int_{t_i}^{t_f} \mathcal{L}(t)dt, \tag{6.72}$$

and is thus expressed in m^{-2} if $\mathcal{L}(t)$ is the instantaneous luminosity. The number of times N_A an event A will happen in this period of time then reads

$$N_A \equiv \sigma(A) \cdot \mathcal{L}_{\text{int}}, \tag{6.73}$$

with $\sigma(A)$ the **cross-section** of the event, always positive and expressed in m^2 so that the number of events is dimension-less. Note that N_A does not predict the exact number of times the event A will happen as the underlying process, following quantum mechanics, is fundamentally random. N_A is thus a statistical average.

While the luminosity is a pure hardware parameter, the cross-section is decoupled from the experimental context and contains the theory prediction for the event A. The larger is the cross-section the more likely it becomes to observe the event at colliders, in the limit of the zero cross-section corresponding to a forbidden process.

MARTY does not compute directly cross-sections but automates the complicated theoretical part i.e. the squared amplitude $|\mathcal{M}(A)|^2$. For incoming particles I with d_I degrees of freedom (dof) and outgoing particles O with d_O dof, the averaged squared amplitude

8. Sometimes lead ions are used instead of protons at the LHC.

is (as a function of the amplitude $i\mathcal{M}(A)$ that depends on the particle dof)

$$|\bar{\mathcal{M}}(A)|^2 \equiv \frac{\eta_O}{\prod_I d_I} \sum_{d_I, d_O} |\mathcal{M}(\mathcal{A})|^2. \tag{6.74}$$

This number takes into account the spin, gauge and flavor dimensions. In the SM for example the gluon has $d_g = 8 \times 2 = 16$ dof because it is in the octet representation of $SU(3)$ and has 2 spin dof as a massless vector boson. η_O is a combinatorial factor taking into account indistinguishable outgoing particles. If all outgoing fields are distinguishable, [9] we have $\eta_O = 1$. Otherwise, this number takes a factor $1/n!$ for each group of n indistinguishable outgoing particles.

This quantity contains the theoretical prediction under the form of a positive number whose value depends on incoming and outgoing particles momenta (energy, speed, direction, ...). Multiplied by a kinematic factor $K(A)$ that also depends on the event A and the Lorentz invariant phase space $d\Pi_{\text{LIPS}}$ (depending only on external momenta), one can obtain the differential cross-section

$$d\sigma(A) = K(A) \cdot |\bar{\mathcal{M}}(A)|^2 d\Pi_{\text{LIPS}}, \tag{6.75}$$

that can be integrated over to derive the total cross-section

$$\sigma(A) = \int d\sigma(A) = \int K(A) \cdot |\bar{\mathcal{M}}(A)|^2 d\Pi_{\text{LIPS}}, \tag{6.76}$$

related to the measured number of events in colliders by equation 6.73.

The hard part of a cross-section calculation is the squared amplitude $|\mathcal{M}|^2$ because the other steps to arrive at the prediction for colliders is model independent and more importantly does not require symbolic calculation. In particular, a simple numerical program can calculate σ from a squared amplitude value. This is why MARTY does not provide an automated procedure for this calculation.

Squaring an amplitude $i\mathcal{M}$ means calculating

$$|\mathcal{M}|^2 = i\mathcal{M} \cdot (i\mathcal{M})^\dagger, \tag{6.77}$$

implying several manipulations to obtain the final result. The amplitude has indices, summed or not, and they must all be contracted to obtain a scalar result that can be evaluated numerically. This requires several simplification procedures that are presented in the following sections.

6.4.2 Spin sums

As seen in section 6.3, amplitudes are expressed as functions of external particles, e.g. for N field insertions $\Phi_i(p_i)$ indexed by i from 1 to N

$$i\mathcal{M} = iT^{A_1 \dots A_N}_{\alpha_1 \dots \alpha_N}(p_1, \dots, p_N) \Phi_1(p_1)^{A_1}_{\lambda_1 \alpha_1} \dots \Phi_N(p_N)^{A_N}_{\lambda_N \alpha_N}, \tag{6.78}$$

9. Two field insertions are distinguishable if they belong to different species (electron and muon for example) or if there is one particle and one anti-particle.

with A_i generalized indices for gauge and flavor vector spaces, α_i Lorentz indices when relevant, λ_i spin indices and $T^{A_1\cdots A_N}_{\alpha_1\ldots\alpha_N}(p_1,\cdots,p_N)$ a general tensor function depending on external momenta, main result of the amplitude calculation. $\Phi_i(p_i)^{A_i}_{\lambda_i\alpha_i}$ are the external fields in momentum space, that have different realizations for spin 0, 1/2 and 1 particles. External fields carry the free indices of the amplitude [10] that are summed up when it is squared, namely

$$\begin{aligned}|\mathcal{M}|^2 &= i\mathcal{M}_{\lambda_1\ldots\lambda_N}\left(i\mathcal{M}_{\lambda_1\ldots\lambda_N}\right)^\dagger \\ &= T^{A_1\ldots A_N}_{\alpha_1\ldots\alpha_N}(p_1,\ldots,p_N)\left(T^{B_1\ldots B_N}_{\beta_1\ldots\beta_N}(p_1,\ldots,p_N)\right)^\dagger \prod_i\left(\sum_{\lambda_i}\Phi_i(p_i)^{A_i}_{\lambda_i\alpha_i}\left(\Phi_i(p_i)^{B_i}_{\lambda_i\beta_i}\right)^\dagger\right),\end{aligned} \tag{6.79}$$

where sums over external fields' spins λ_i is explicitly represented because they can be replaced by well-known identities that are presented in the following. For a scalar ϕ, there is no spin to sum, no Lorentz index and the identity is trivial

$$\sum_\lambda \phi(p)^A_\lambda\left(\phi(p)^B_\lambda\right)^\dagger = \delta^{AB}. \tag{6.80}$$

For fermions, the spin tensor is noted $u(p)$ for particles and $v(p)$ for anti-particles. Equations of motion for spin $1/2$ particles of mass m imply the following spin sum rules:

$$\begin{aligned}\sum_\lambda u(p)^A_{\lambda\alpha}\bar{u}(p)^B_{\lambda\beta} &= \left(\not{p}+m_i\right)_{\alpha\beta}\cdot\delta^{AB}, \\ \sum_\lambda v(p)^A_{\lambda\alpha}\bar{v}(p)^B_{\lambda\beta} &= \left(\not{p}-m_i\right)_{\alpha\beta}\cdot\delta^{AB},\end{aligned} \tag{6.81}$$

with α and β Dirac indices, and the spin index λ summed over the two possible values of spin, $\pm 1/2$. We have used the identities

$$\bar{u}_{/v}(p) \equiv u_{/v}(p)^\dagger\gamma^0, \tag{6.82}$$

$$\not{p} \equiv p_\mu\gamma^\mu. \tag{6.83}$$

For spin 1 particles, the spin tensor is usually noted ϵ and the spin sum rule reads

$$\sum_\lambda \epsilon(p)^A_{\lambda\mu}\left(\epsilon(p)^B_{\lambda\nu}\right)^\dagger \to -g_{\mu\nu}\cdot\delta^{AB} \tag{6.84}$$

for massless particles and

$$\sum_\lambda \epsilon(p)^A_{\lambda\mu}\left(\epsilon(p)^B_{\lambda\nu}\right)^\dagger = \left(-g_{\mu\nu}+\frac{p_\mu p_\nu}{M^2}\right)\cdot\delta^{AB} \tag{6.85}$$

for a vector boson of mass M. Here λ takes the spin values -1, 0 and 1 while we know from quantum field theory that a massless vector boson has only two allowed spin values

10. These indices are generally not represented in the literature, one thus has to keep in mind that the amplitude is not a simple number but an indexed tensor in the spin spaces of external fields.

because as it lies on the light cone, spin 0 is forbidden. For abelian gauge theories, for example the $U(1)$ group of the electromagnetic interaction in the SM, the replacement in equation 6.84 gives correct results as the non-physical spin 0 contribution vanishes through Ward identities. For non-abelian gauge theories however, this contribution does not vanish and must be compensated. In MARTY we still use the simple spin sum rules in equation 6.84 for non-abelian groups and proper quantization is ensured by ghosts as explained in 6.3.2.

Consequently, MARTY has to apply equations 6.80, 6.81, 6.84 and 6.85 to replace external fields in the results. The specific simplifications that must then be performed in the different vector spaces (Minkowski, Dirac and group representations) will be presented in the following section.

There are two differences between spin tensor definitions in this section and in the literature that are important to highlight:

- Spin tensors are not defined for scalar particles in the literature, as a proper definition would imply the trivial relation $\phi(p) = 1 \, \forall p$.
- Spin tensors do not carry gauge and flavor indices, noted A and B here. Their mathematical definition includes only spin and Lorentz indices.

These are not accidental differences as they are meant to keep track of external legs all along the calculation without losing any information. For example, one could encounter in the literature an amplitude such as

$$i\mathcal{M} \propto T_{\mu}^{AB} \epsilon(p)^{\mu}, \tag{6.86}$$

with some tensor T_{μ}^{AB} and gauge indices A and B. One cannot tell just from this expression what are the external particles, and which carry gauge indices. We can only infer that there is one vector boson, and two particles carrying a gauge index (not knowing if the vector is one of them). In MARTY however, an equivalent amplitude could read

$$i\mathcal{M} \propto T_{\mu}^{AB} \epsilon(p)_{\lambda}^{\mu A} \phi(k)_{\lambda'}^{B} \phi(q)_{\lambda''}, \tag{6.87}$$

with the explicit index dependence for external fields that makes it clear what process has been calculated, with two scalar particles. When doing a calculation by hand it is always known what are the external legs and where the indices come from, but for an automated software program information loss can be an issue if it is needed later on. In this case, keeping gauge indices and spin tensors for scalars is used to calculate Wilson coefficients (see section 6.5) and to connect pieces of amplitudes that have been calculated separately. One can check that using modified spin tensors and the sum rules defined in equations 6.80, 6.81, 6.84 and 6.85, any observable calculated from the amplitude (like its square) will yield the same result as when standard prescriptions are used.

6.4.3 Traces

Starting from 6.79 and applying sum rules defined in the previous section for external fields, the squared amplitude takes the form

$$|\mathcal{M}|^2 = T_{\alpha_1 \ldots \alpha_N}^{A_1 \ldots A_N}(p_1, \ldots, p_N) \left(T_{\beta_1 \ldots \beta_N}^{A_1 \ldots A_N}(p_1, \ldots, p_N) \right)^{\dagger} \cdot S_{\alpha_1 \ldots \alpha_N \; \beta_1 \ldots \beta_N}(p_1, \ldots, p_N), \tag{6.88}$$

with $S_{\alpha_1 \ldots \alpha_N \ \beta_1 \ldots \beta_N}(p_1, \ldots, p_N)$ the result of sum rules depending on external momenta. All indices are now contracted, and the only tensors that can appear in this expression are: [11]

- External momenta p_i^μ.
- Minkowski metric $g_{\mu\nu}$.
- Fully anti-symmetric symbol $\epsilon^{\mu\nu\rho\sigma}$.
- Gamma matrices γ^μ and related tensors.
- Group generators T_{ij}^A, f^{ABC} and related tensors.

As all indices are contracted, the number of possible structures remaining in the expression are limited, namely

- Products of momenta $g_{\mu\nu} p_i^\mu p_j^\nu$.
- Fully anti-symmetric product of momenta $\epsilon_{\mu\nu\rho\sigma} p_i^\mu p_j^\nu p_k^\rho p_l^\sigma$.
- γ-matrix traces, $\mathrm{Tr}\left(\gamma^{\mu_1} \ldots \gamma^{\mu_N}\right)$ or $\mathrm{Tr}\left(\gamma^{\mu_1} \ldots \gamma^{\mu_N} \gamma^5\right)$.
- Group generator traces $\mathrm{Tr}\left(T^{A_1} \ldots T^{A_N}\right)$.

Traces can be calculated as explained in section 6.3 to yield simpler tensors. The result is then composed only of scalar quantities once we define

$$s_{ij} \equiv g_{\mu\nu} p_i^\mu p_j^\nu, \tag{6.89}$$

and

$$e_{ijkl} \equiv \epsilon_{\mu\nu\rho\sigma} p_i^\mu p_j^\nu p_k^\rho p_l^\sigma, \tag{6.90}$$

that is non zero only for independent momenta i, j, k, l i.e. with four different momenta in a process with at least five external legs (otherwise one can apply momentum conservation to replace one of the four momenta by the other ones). A squared amplitude in MARTY is typically expressed as a real function of model parameters and scalar quantities such as s_{ij} and e_{ijkl} that have to be user-defined at the numerical level.

For group theory traces more structures can arise, in particular when the amplitude already contains a trace from a loop. Figure 6.10 shows an example of a process containing a loop at the amplitude level. Such a trace can be expressed in general as

$$\mathrm{Tr}\left(T^A T^B T^C\right) = d^{ABC} + iC f^{ABC}, \tag{6.91}$$

with C a numerical constant, d^{ABC} a fully symmetric tensor and f^{ABC} the fully anti-symmetric structure constants of the underlying algebra. When taking the square of such an amplitude, one obtains

$$\left\| \left(d^{ABC} + iC f^{ABC}\right) \right\|^2 = d^{ABC} d^{ABC} + C^2 f^{ABC} f^{ABC}, \tag{6.92}$$

11. There can be a few exceptions but the way BSM theories are built in general imply that all indices are Lorentz or gauge / flavor symmetry indices, in which case invariant Lagrangian terms that can be written contain only tensors of this list without loss of generality.

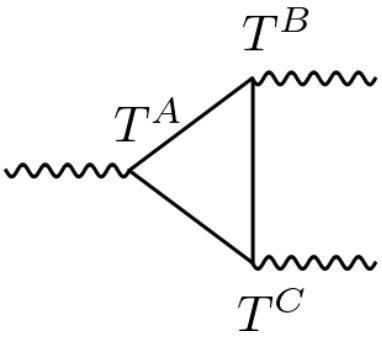


Figure 6.10 – Example of a generic process containing a trace of group theory structures at the amplitude level. Each of the three vertices carries a generator $T^{A/B/C}$ and the amplitude is therefore proportional to $\mathrm{Tr}\left(T^A T^B T^C\right)$. This diagram has been generated using GRAFED, see section 2.5 for more details.

in which the products of f and d vanish trivially through their respective (anti-)symmetry. Finding the values of all the possible contractions for all algebras and products of diagrams is a very tedious task as one can see in [124]. Instead of trying to *catch 'em all* it is simpler to build a *pokedex*, catalog of all possible structures, do the work only for the simpler ones and let the user fill the missing values. In the example above one can define a scalar quantity in a given group G

$$d_{33}^G \equiv d_G^{ABC} d_G^{ABC}. \tag{6.93}$$

The value of d_{33} will be provided by MARTY in $SU(N)$ but exotic quantities such as d_{55} or d_{5333} may not be. The latter can be retrieved from the literature if needed. The main work of MARTY is to perform all the complicated algebra and let only to users the task to define scalar quantities from the process kinematics (s_{ij}, e_{ijkl} defined above) and possibly group theory in one-loop processes with multiple external vector bosons.

A word on group indices

Figure 6.10 presented a process containing a trace of algebra generators T^A, that depends on the representation inside the loop $\mathcal{R}$ and should be noted $T^A(\mathcal{R})$. Equation 6.91 thus becomes

$$\mathrm{Tr}\left(T^A(\mathcal{R})T^B(\mathcal{R})T^C(\mathcal{R})\right) = d^{ABC}(\mathcal{R}) + i\beta(\mathcal{R})f^{ABC}, \tag{6.94}$$

with $d^{ABC}(\mathcal{R})$ depending on the representation. This means that identities such as equation 6.93 should in principle be defined for all representations in a given group. Following [124], we implemented in MARTY the reduction of fully-symmetric tensors to reference tensors together with the initial trace calculation. The previous equation then reads

$$\mathrm{Tr}\left(T^A(\mathcal{R})T^B(\mathcal{R})T^C(\mathcal{R})\right) = I(\mathcal{R})d^{ABC} + iC(\mathcal{R})f^{ABC}, \tag{6.95}$$

with d^{ABC} the reference 3D fully-symmetric tensor that is representation-independent. MARTY calculates automatically the expression of the trace for an arbitrary number of generators [12], group indices $I(\mathcal{R})$ and coefficients $C(\mathcal{R})$ in all semi-simple Lie groups and

12. Traces with more generators generate more terms and there is therefore a computational limitation.

their representations, letting to users only the definition of reference tensor products such as the one defined in equation 6.93, that have the nice property of being representation-independent. These products, hard to calculate in a general and systematic way, can be found in the literature e.g. in [124].

6.4.4 A computational challenge

As specified in section 6.3, an amplitude calculation is already a time consuming task that can generate very large expressions, especially at the loop level with numerous diagrams. For an amplitude with N terms, the squared quantity will have at least N^2, but the number is actually much larger because of spin sums, traces and index contractions that we presented previously in this section. There are two main challenges for the squared amplitude:

- The calculation must be well-contained in space (memory) and time for large expressions to allow a program to be executed on a standard laptop.
- The resulting expression must be well-enough factored to be as compact as possible. An expression can be too large to be usable at the numerical level because of compilation issues or even time optimization, as they are often meant to be called millions, billions or more times to scan the parameter space of a BSM theory.

This issue has been for some time a strong limitation for the calculation of squared amplitudes at the loop level. With optimizations that we implemented at the CSL level MARTY is now able to handle one-loop squared amplitudes in most cases. These optimizations are:

- **A Smart expansion.** The best way to obtain a factored result is to avoid expanding anything. This is unfortunately not possible, as index contraction requires to expand indexed structures. A smart expansion must therefore be able to expand only what is needed for a precise calculation and keep the rest of the expression factored. This is possible using the CSL algorithm `DeepPartialExpand()`, more efficient than the usual `DeepExpand()`.
- **A Smart factorization.** This procedure comes at the end of a calculation and must be able to find common factors to minimize the number of mathematical operations defining the result, even with only partial matches. This is possible using the CSL algorithm `DeepHardFactor()`, much more efficient than the usual `DeepFactor()`.
- **A Compression algorithm.** Finally, a compression algorithm can recognize patterns in the result to find new abbreviations and reduce the size of the expression (in the symbolic program and the generated code). This is possible using the CSL algorithm `matcher::compress()`.

These three specific algorithms bring two main gains. The first one takes place during the symbolic calculations. By expanding less terms and using an efficient factorization algorithm, expressions are smaller and all the calculations become faster while taking much less memory. The better factorization combined with the compression brings another important gain, this time during the numerical computations i.e. when using the

code generated by MARTY to evaluate the results (see section 6.1). With a smaller expression and by finding common patterns that can be evaluated only once, the number of numerical operations required to evaluate a large expression is lowered significantly. This second gain is the most important. While the symbolic calculations are performed only once to make MARTY generate the final library, the following phenomenological analysis relies in general on multiple evaluations of the numerical results (squared amplitudes or Wilson coefficients). This is for example the case when scanning the parameter space of a BSM model and evaluating the results for each scenario. Consequently, the generated numerical program must be as efficient as possible and the three algorithms discussed above are necessary for that purpose.

6.4.5 Squared amplitudes in MARTY

We saw in section 6.3 and in particular in sample code 40 how to calculate amplitudes with MARTY. Squaring an amplitude is even simpler as demonstrated in sample code 41.

Sample code 41: Squared amplitudes

Calculating a transition amplitude

```
auto res = model.computeAmplitude(
    TreeLevel, // or OneLoop
    {Incoming("h"), Outgoing("e"), Outgoing(AntiPart("e"))}
);
```

Squaring the amplitude

```
Expr square = model.computeSquaredAmplitude(res);
```

Displaying the result in standard output

```
cout << square << endl;
```

Note The squared amplitude is a simple CSL expression that can be used without additional abstraction.

6.5 Wilson coefficients

6.5.1 Generalities

Wilson coefficients are complex-valued functions in front of operator structures in MARTY. In Effective Field Theories (EFT), it is convenient to use an effective Hamiltonian instead of a Lagrangian:

$$\hat{\mathcal{H}}_{eff} \equiv \sum_i C_i \hat{\mathcal{O}}_i, \tag{6.96}$$

with $\hat{\mathcal{O}}_i$ effective operators and C_i their Wilson coefficients. The transition amplitude between an initial state i and a final state f is the matrix element of this Hamiltonian:

$$i\mathcal{M}(i \to f) = \langle f|(-i\hat{\mathcal{H}}_{eff})|i\rangle = -i\sum_i C_i \langle f|\hat{\mathcal{O}}|i\rangle. \tag{6.97}$$

The operator matrix elements $\langle f|\hat{\mathcal{O}}|i\rangle$ are not in general calculated pertubatively and contain long distance (low energy) effects. This is in particular the case when considering hadron physics in which the long distance QCD effects break the purely perturbative calculations that MARTY automates. Nevertheless, the BSM dependence lies in the Wilson coefficients and a perturbative calculation can be used to determine their values as explained in [125]. Consequently, the contributions of matrix elements – that are model-independent – to observables can be measured in colliders or calculated with lattice QCD once and for all. Then, theoretical physicists only need to compute Wilson coefficients C_i to study a new BSM scenario.

In MARTY a matrix element is simply a particular index contraction of external fields. A general amplitude with N external fields $\{\Phi_I^{A_I}\}_I$ with generalized indices A_I can be expressed as

$$i\mathcal{M} \equiv -i\alpha \sum_i C_i \cdot T_i^{A_1\cdots A_N} \cdot \Phi_1^{A_1} \cdots \Phi_N^{A_N}, \tag{6.98}$$

with $T_i^{A_1\cdots A_N}$ all different external fields contractions in the resulting amplitude and α a convention dependent constant. Multiplying the result by $\frac{i}{\alpha}$, Wilson coefficients can be identified as the scalar factors C_i in front of the different matrix elements.

When asked, MARTY can decompose an amplitude in the different external field contractions it encounters and return the coefficients in front, taking into account a global user-defined factor α. This can be used to extract precise contributions in the amplitude.

6.5.2 Additional simplifications

From equation 6.98, the derivation of Wilson coefficients seems to be only a matter of coefficient identification in the amplitude result. In practice, this requires more care and in particular to define correctly and uniquely the different structures $T_i^{A_1\cdots A_N}$.

LO vs. NLO

For now MARTY provides matching at the Leading Order (LO) at tree-level or one-loop and Next-to-Leading Order (NLO) complications are therefore not mentioned in the following. Although it is possible in MARTY to perform calculations with integrated-out particles such as the W-boson in the SMEFT that has an effective propagator

$$\frac{-ig_{\mu\nu}}{p^2 - M_W^2} \to \frac{ig_{\mu\nu}}{M_W^2} + \mathcal{O}\left(\frac{p^2}{M_W^2}\right), \tag{6.99}$$

the matching procedure has not been automated. In case one wants to calculate NLO corrections (one-loop quantities correcting the tree level), the process can be calculated in both the effective and full theories. Then, the matching has to be done by hand using the coefficient in both theories. See [125] for more details on NLO matching.

Fierz identities

For 4-fermion operators, a term in the amplitude is typically expressed as

$$i\mathcal{M} \ni \alpha \left(\bar{\psi}_1 \Gamma^A \psi_2\right) \cdot \left(\bar{\psi}_3 \Gamma^B \psi_4\right), \tag{6.100}$$

with α a coefficient, ψ_i the four external fermions and $\Gamma^{A,B}$ Dirac structures. In the following we define

$$\left(\Gamma^A\right)_{ij} \equiv \left(\bar{\psi}_i \Gamma^A \psi_j\right). \tag{6.101}$$

One common issue that has to be addressed with these operators is that the amplitude will typically contain different pairings between fermions such as

$$i\mathcal{M} \ni \beta \left(\Gamma^A\right)_{14} \left(\Gamma^B\right)_{32}, \tag{6.102}$$

whereas Wilson coefficients for a given process are always defined with fixed pairings, for example $(12)(34)$. One then has to find the contribution of a term $(14)(32)$ to a coefficient defined with $(12)(34)$ in the literature. This is possible using general Fierz-type identities [126]. Defining Γ^A in a minimal basis one has two well-know choices:

$$\begin{aligned} \Gamma^A &\in \left\{1, \gamma^5, \gamma^\mu, \gamma^\mu\gamma^5, \sigma^{\mu\nu}\right\}, \\ \Gamma^A &\in \left\{P_L, P_R, \gamma^\mu P_L, \gamma^\mu P_R, \sigma^{\mu\nu}\right\}, \end{aligned} \tag{6.103}$$

that we name standard and chiral basis respectively. The dual bases are defined as

$$\begin{aligned} \Gamma_A &\in \left\{1, \gamma^5, \gamma_\mu, \gamma^5\gamma_\mu, \sigma_{\mu\nu}\right\}, \\ \Gamma_A &\in \left\{P_L, P_R, \gamma_\mu P_R, \gamma_\mu P_L, \frac{1}{2}\sigma_{\mu\nu}\right\}, \end{aligned} \tag{6.104}$$

by imposing

$$\mathrm{Tr}\left(\Gamma^A \Gamma_B\right) = \lambda \cdot \delta^A_B, \tag{6.105}$$

with λ equal to 4 and 2 for the standard and chiral bases respectively.

As derived from the completeness relation in [126], the generalized Fierz-type identity reads

$$\left(\Gamma^A\right)_{14}\left(\Gamma^B\right)_{32} = \frac{1}{\lambda^2}\sum_{C,D}\mathrm{Tr}\left(\Gamma^A\Gamma_C\Gamma^B\Gamma_D\right)\left(\Gamma^D\right)_{12}\left(\Gamma^C\right)_{34}. \tag{6.106}$$

This general relation allows us to express all contributions with only one fermion pattern. As equation 6.106 is linear in Γ^A and Γ^B and that the bases in equation 6.103 are complete, Fierz identities can be used for any initial $\Gamma^{A,B}$.

By applying twice Fierz identities we can simplify general products of bilinears depending on non-elementary $\hat{\Gamma}^A$ and $\hat{\Gamma}^B$ and obtain

$$\left(\hat{\Gamma}^A\right)_{12}\left(\hat{\Gamma}^B\right)_{34} = \frac{1}{\lambda^4}\sum_{C,D,E,F}\mathrm{Tr}\left(\hat{\Gamma}^A\Gamma_C\hat{\Gamma}^B\Gamma_D\right)\mathrm{Tr}\left(\Gamma^D\Gamma_E\Gamma^C\Gamma_F\right)\left(\Gamma^F\right)_{12}\left(\Gamma^E\right)_{34}, \tag{6.107}$$

which is now expressed only with basis elements Γ^E and Γ^F. These simplifications imply a lot of algebra but are necessary to fully simplify 4-fermion amplitudes and extract general Wilson coefficients. Note that this is completely unnecessary when calculating a squared amplitude as complicated Dirac structures $\hat{\Gamma}^A$ and $\hat{\Gamma}^B$ are simplified in traces anyway.

Fermion ordering

For processes that rely on fermion-violating interactions or Majorana particles the order of one fermion bilinear can be undefined. Let us consider a simple example with two fermions ψ and ξ. One can obtain an amplitude such as

$$i\mathcal{M} = \alpha\left(\bar{\psi}\Gamma_1\xi\right) + \beta\left(\bar{\xi}\Gamma_2\psi\right), \tag{6.108}$$

with two bilinears that have a different flow. Following identities given in section 4.6.2 a fermion bilinear can be reversed using the conjugation matrix C and

$$\begin{aligned}\bar{\xi}\Gamma_2\psi &= \left(\bar{\xi}\Gamma_2\psi\right)^T\\ &= \psi^T\Gamma_2^T\bar{\xi}^T\\ &= \bar{\psi}C\Gamma_2^TC\xi\\ &= -\bar{\psi}\Gamma_2'\xi,\end{aligned} \tag{6.109}$$

where Γ_2' is the conjugate of Γ_2 defined in equation 4.25 that always has a regular form. [13] The amplitude then reads

$$i\mathcal{M} = \bar{\psi}\left(\alpha\Gamma_1 - \beta\Gamma_2'\right)\xi. \tag{6.110}$$

In order to illustrate the relevance of such a transformation, let us consider the case $\Gamma_2' = \Gamma_1$. The amplitude becomes

$$i\mathcal{M} = (\alpha-\beta)\bar{\psi}\Gamma_1\xi, \tag{6.111}$$

13. The distinction between particles and anti-particles (u and v spinors) is not considered here as the identities are identical and that terms with the same fermion flow but different particle conjugations (such as $\bar{u}_\psi v_\xi$ and $\bar{v}_\psi u_\xi$) cannot arise in the same calculation.

where the coefficient in front of $\bar{\psi}\Gamma_1\xi$ is now $\alpha - \beta$ whereas in the bare result of equation 6.108 one would have read only the α contribution. Together with Fierz identities presented in the section above, fermion ordering allows us to define one unique structure and apply it to all terms in the amplitude. This is required to match the results with operators in the literature that are always defined with fixed fermion flows.

Once again, fermion ordering is irrelevant in squared amplitude calculations as using identities of section 4.6.2 one can extend spin sums defined in equation 6.81 to mixed terms such as

$$\begin{aligned}\sum_\lambda u(p)^A_{\lambda\alpha}v(p)^B_{\lambda\beta} &= \left[\sum_\lambda u(p)^A_{\lambda\alpha}\bar{u}(p)^B_{\lambda\delta}\right]C_{\delta\beta} = \left(\not{p}+m_i\right)_{\alpha\delta}C_{\delta\beta}\cdot\delta^{AB},\\ \sum_\lambda \bar{u}(p)^A_{\lambda\alpha}\bar{v}(p)^B_{\lambda\beta} &= C_{\alpha\delta}\left[\sum_\lambda v(p)^A_{\lambda\delta}\bar{v}(p)^B_{\lambda\beta}\right] = C_{\alpha\delta}\left(\not{p}-m_i\right)_{\delta\beta}\cdot\delta^{AB},\end{aligned} \tag{6.112}$$

with an explicit dependence on C that will cancel out before any trace calculation to finally recover well-defined fermion loops.

Matching of different operator bases

Once the amplitude has been maximally reduced one can identify Wilson coefficients at LO directly in the result. In practice, MARTY could stop there and delegate this part to users as it does not require much algebra. It is however more practical and more importantly less error prone to have an automated way to identify coefficients as we demonstrate in the following.

The final challenge consists in matching conventions regularly used in the literature to the automated calculation output. As equation 6.103 introduced in the case of γ-matrices, multiple bases can be used to define the set of operators we consider. In flavor physics, different bases can even be used in the same operator such as

$$\mathcal{O}_{10} = (\bar{s}\gamma^\mu P_L b)\left(\bar{\mu}\gamma_\mu\gamma^5\mu\right) \tag{6.113}$$

that mixes P_L and γ^5 from two different bases. Then, depending on the basis used to express the results one has to know the contribution to $\mathcal{O}_{10}$ of structures such as

$$T_1 = (\bar{s}\gamma^\mu b)\left(\bar{\mu}\gamma_\mu\gamma^5\mu\right) \tag{6.114}$$

in the standard basis or

$$T_2 = (\bar{s}\gamma^\mu P_L b)\,(\bar{\mu}\gamma_\mu P_R\mu) \tag{6.115}$$

in the chiral basis.

In MARTY we chose the standard basis as the reference because it contains the identity. As the chiral basis does not contain the identity that it is always implicit in expressions, it can cause trouble to identify operator contributions. We therefore project both the result (T_1 and T_2 in the example) and the operator for which we search the contribution ($\mathcal{O}_{10}$)

on the same basis. In this example we obtain

$$\begin{aligned}
\mathcal{O}_{10} &= \frac{1}{2}\left(\bar{s}\gamma^{\mu}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right) - \frac{1}{2}\left(\bar{s}\gamma^{\mu}\gamma^{5}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right), \\
T_1 &= \left(\bar{s}\gamma^{\mu}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right) \\
T_2 &= \frac{1}{4}\left(\bar{s}\gamma^{\mu}b\right)\left(\bar{\mu}\gamma_{\mu}\mu\right) - \frac{1}{4}\left(\bar{s}\gamma^{\mu}\gamma^{5}b\right)\left(\bar{\mu}\gamma_{\mu}\mu\right) \\
&\quad + \frac{1}{4}\left(\bar{s}\gamma^{\mu}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right) - \frac{1}{4}\left(\bar{s}\gamma^{\mu}\gamma^{5}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right).
\end{aligned} \tag{6.116}$$

For the target operator the relation must be reversed to find the correct contributions from the basis that we defined, namely

$$\begin{aligned}
\alpha\left(\bar{s}\gamma^{\mu}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right) &\to +\alpha\mathcal{O}_{10} \\
\alpha\left(\bar{s}\gamma^{\mu}\gamma^{5}b\right)\left(\bar{\mu}\gamma_{\mu}\gamma^{5}\mu\right) &\to -\alpha\mathcal{O}_{10},
\end{aligned} \tag{6.117}$$

that can be derived in this case from $1 = P_R + P_L$ and $\gamma^5 = P_R - P_L$. Let us now consider an explicit amplitude that could be an output of MARTY:

$$i\mathcal{M} \equiv i\alpha T_1 + i\beta T_2. \tag{6.118}$$

One can derive from the amplitude expression the contribution of $T_{1,2}$ to $\mathcal{O}_{10}$, namely (multiplying the amplitude by i to obtain the Wilson coefficient):

$$-\mathcal{M} = -\left(\alpha + \frac{\beta}{2}\right)\mathcal{O}_{10} + \ldots, \tag{6.119}$$

where the dots mean that there are other contributions and we finally obtain the coefficient

$$C_{10} = -\left(\alpha + \frac{\beta}{2}\right). \tag{6.120}$$

This procedure to match operators expressed in different bases is a challenge when one writes an automated program to calculate Wilson coefficients and is not a simple identification process in the amplitude. The amplitude together with the effective operators must be projected on a given basis to obtain the final coefficients. Furthermore, a simple user interface must be provided to define easily operator structures from which users want to extract the Wilson coefficients.

Wilson coefficients in MARTY

There are two ways to derive Wilson coefficients in MARTY. If one wants only to decompose the result on different operator structures without searching for precise contributions (or when the decomposition is very simple), the amplitude can be calculated separately and then given to MARTY for the coefficient extraction as shown in sample code 42.

Sample code 42: Wilson coefficients 1/2

Let us consider here a variable `res` containing the result of an amplitude calculation as shown in sample code 40.

Decomposing the amplitude in the operator basis

```
auto wilsons = model.getWilsonCoefficients(res);
Display(wilsons);
```

The type deduced by `auto` is `WilsonSet`.

Getting the expressions of Wilson coefficients

```
int nCoefs = wilsons.size();
Expr C1 = wilsons[0].coef.getCoefficient();
Expr C2 = wilsons[1].coef.getCoefficient();
```

Note The resulting expressions can be given to the library generators as presented in the user manual [86].

When calculating Wilson coefficients, in particular for 4-fermion operators, MARTY must apply specific procedures at the amplitude level and therefore cannot take an amplitude calculated separately. [14] This is presented in sample code 43.

Sample code 43: Wilson coefficients 2/2

Setting options for the Wilson coefficients

```
FeynOptions options;
// Setting the Dirac basis for the decomposition:
options.setWilsonOperatorBasis(OperatorBasis::Standard); // or ::Chiral
// Defining a global coefficient factored out from Wilsons:
options.setWilsonOperatorCoefficient(e*e/(4*CSL_PI));
```

Calculating directly the Wilson coefficients

```
auto wilsons = model.computeWilsonCoefficients(
    OneLoop,
    {Incoming("b"), Outgoing("s"), Outgoing("A")}, // b -> s gamma
    options // send the options the the model
    );
Display(wilsons);
```

14. The amplitude can be calculated separately to extract Wilson coefficients but with 4-fermion operators complications arise in operator definitions and without a specific treatment of the amplitude it is hard to recover a well-defined basis.

6.6 Automating calculations

In some phenomenological analyses it is necessary to perform a large number of similar calculations. In this case the method of using explicit particle names is tedious and not recommended. The calculation of all 2-to-2 processes for a model with 50 particles for example is a task that one has to automate. While the `getParticles()` method of a model returns the list of all particles, the method that must be used in that case is `getPhysicalParticles()`. Contrary to the first option, this method removes redundancies. For a Dirac fermion ψ for example the left- and right-handed parts are also considered as particles and will cause redundancies in calculations. The `getPhysicalParticles()` method will take care of it and return a list of independent physical particles. This is also possible to filter even more this list, as summarized in sample code 44.

Sample code 44: Get particles lists from a model

Exhaustive list, but not suited to automate calculations

```
auto particles = model.getParticles();
```

Removing redundancies

```
auto particles = model.getPhysicalParticles();
```

Filtering physical particles

```
auto fermions = model.getPhysicalParticles(
    [&](Particle p) { return p->isFermionic(); }
);
auto bosons = model.getPhysicalParticles(
    [&](Particle p) { return p->isBosonic(); }
);
auto vectors = model.getPhysicalParticles(
    [&](Particle p) { return (p->getSpinDimension() == 3); }
);
auto Ni = model.getPhysicalParticles(
    [&](Particle p) { return (p->getName()[0] == 'N'); }
);
```

Note The deduced return type is each time `std::vector<mty::Particle>` allowing one to iterate over it.

Note The lambda expression given to filter out the result can be any user-defined boolean predicate taking a `Particle` as parameter.

Once the lists of particles have been filtered out from the model, one can iterate over them to automate the calculation of a large number of processes. This is presented in sample code 45.

Sample code 45: Automate a large number of calculations

```
for (auto f : fermions) {
    for (auto v : vectors) {
        auto ampl = model.computeAmplitude( // f fbar -> v vbar
            TreeLevel,
            {Incoming(f), Incoming(AntiPart(f)),
            Outgoing(v), Outgoing(AntiPart(v))}
        );
        auto squared = model.computeSquaredAmplitude(ampl);
        // Store, do something with the squared amplitude
    }
}
```

This chapter presented some of the main challenges and requirements for automated BSM one-loop calculations. They are multiple, starting from the theory Lagrangian up to the final analytical and simplified quantities. The derivation of Feynman rules and diagrams, the γ-matrix identities and traces, the group theory simplifications, the tensor reduction at one-loop, the general Fierz identities, the spin sum rules and the equations of motions must be applied automatically by MARTY in order to generate theoretical quantities for general BSM scenarios. All these steps must be based on a reliable high-performance symbolic computation machinery able to manipulate large expressions and generate compact results quickly. As discussed in this chapter, CSL is the symbolic computation library embedded in MARTY and its algorithms have been designed specifically for high energy physics purposes. Several examples of results demonstrating the ability of MARTY to perform all the calculations previously introduced are presented in the next chapter.

CHAPTER 7

Selection of results

7.1 Introduction

In this chapter some of the results of the calculations that we have performed using MARTY are presented. They rely on the simplification procedures developed in chapter 6 to calculate amplitudes, squared amplitudes and Wilson coefficients at tree-level and at one-loop. For pedagogical and validation purposes we consider known examples in the SM and beyond. However, MARTY has not been developed for these specific examples and is fully general. Therefore, the calculations that we develop can be straight-forwardly generalized to all models that can be built in MARTY. [1]

The results presented in this chapter have been translated into automated tests for MARTY to facilitate future releases. All related programs can be found on the public gitlab repository [127].

7.1.1 Validation

There are five main features of MARTY that we can test:

- **Symbolic calculations**, i.e. the mathematical accuracy of CSL.
- **Built-in models** in MARTY, for now the Standard Model (SM), 2 Higgs Doublets Models (2HDM) and the Minimal Super-symmetric Standard Model (MSSM). If calculations are accurate but the model is inconsistent, the results will be wrong. Checking that models are valid is therefore an important part of the test procedures.
- **Transition amplitude** calculations.
- **Squared amplitude** calculations.
- **Wilson coefficient** calculations.

The two first features are not tested directly. We can consider the validity of symbolic manipulations and built-in models for granted as they are indirectly tested when considering high energy physics calculations. Consequently, when building a new model

1. See chapter 4 for the models that can be implemented in MARTY.

with MARTY one should check that interaction vertices are consistent with its theoretical definition before doing any calculation with it.

7.1.2 Different kinds of tests

In the following we will refer to three types of tests:

- **Numerical tests.** They simply compare numbers such as squared amplitudes. They are composed of two programs: The MARTY script at the symbolic level and a numerical C++ code using a library generated by MARTY during the execution of the first program.
- **Symbolic tests.** Their success depends on the precise mathematical expressions of the results. These tests do not use any numerical library generated by MARTY. [2]
- **Termination tests.** They are not for comparison purpose, just to make sure that a program gets to the end without error. They are less informative but still relevant because MARTY performs a large number of consistency checks during a calculation. If the procedures finish normally it means that they are mathematically consistent, even if it does not guarantee the result's accuracy.

7.2 Amplitude calculations

A pure amplitude calculation [3] is harder to test as one must look at the exact symbolic expression, that depends on conventions and more importantly on the exact simplification procedures that can change between two versions of the code. One example of difficulty is momentum conservation. Consider a $1 + 2 \to 3 + 4$ process whose amplitude is proportional to $(p_1 + p_2)^2$. Using a slightly different procedure, the momentum conservation could yield an amplitude proportional to $(p_3 + p_4)^2$ instead. The two are mathematically equivalent but the comparison is not trivial. In general, it is almost impossible to ensure that two different realizations of the same procedure will yield the exact same symbolic expression. This is in particular true when the simplification methods are improved to lighten the results.

There are therefore very few direct tests of amplitudes, but as one can see in the next sections they can be indirectly tested in squared amplitudes and Wilson coefficients. If examples are chosen appropriately there are no feature of amplitudes that cannot be tested indirectly in squared amplitude and Wilson coefficient calculations.

There are for now two bare amplitude tests that address the conjugation matrix problem introduced in section 4.6 when considering fermion-number violating interactions.

2. When automated, one still has to find a way to generate a number because it is the only reliable object that one can use for comparison when writing tests.

3. As opposed to squared amplitude and Wilson coefficient calculations.

7.2.1 Conjugation matrix consistency

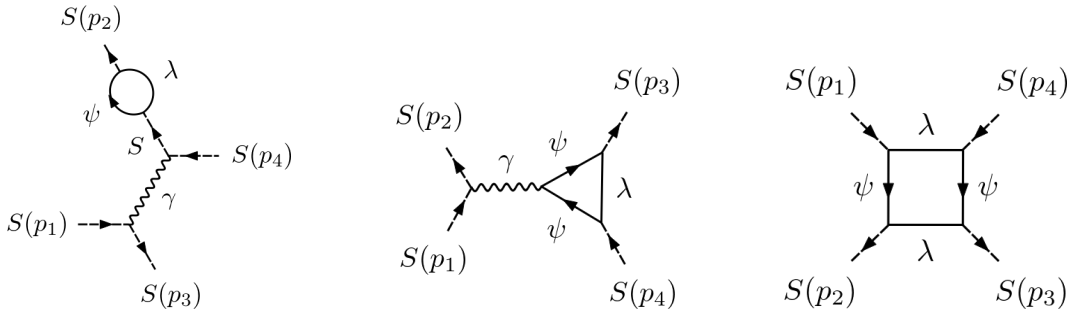
We test here that fermion loops with Majorana fermions are simplified consistently. As explained in section 4.6, conjugation matrices can appear in vertices or Majorana propagators and must be simplified in traces. If an inconsistency is detected during the calculation, `MARTY` will not be able to calculate traces and the program will be stopped. This test is a **termination test.**

The Lagrangian composed of a vector boson A, a scalar boson S, a pseudo-scalar boson P, a Dirac fermion ψ and a Majorana fermion λ reads [128]

$$\begin{aligned}\mathcal{L} = &-\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \frac{1}{2}\bar{\lambda}i\partial\!\!\!/\lambda + \bar{\psi}\left(iD\!\!\!/ - m\right)\psi \\ &+ \left(D_\mu S\right)^\dagger\left(D^\mu S\right) + m^2 S^\dagger S + \left(D_\mu P\right)^\dagger\left(D^\mu P\right) + m^2 P^\dagger S \\ &+ iQ\left[\bar{\psi}\left(S + i\gamma^5 P\right)\lambda - \bar{\lambda}\left(S^\dagger + i\gamma^5 P^\dagger\right)\psi\right] \\ &+ \frac{Q^2}{2}\left(S^\dagger P - S P^\dagger\right)^2,\end{aligned} \tag{7.1}$$

where $D_\mu \equiv \partial_\mu + iQA_\mu$ and $F_{\mu\nu} \equiv \partial_\mu A_\nu - \partial_\nu A_\mu$. S, P and ψ have a charge Q with respect to the $U(1)$ group gauged by the photon A. The last interaction term is purely scalar and is not the purpose of this test.

From this Lagrangian the amplitude for the diffusion process $SS \to SS$ can be calculated at the one-loop level. Examples of contributions are presented in figure 7.1. From



(a) 2-point function **(b)** 3-point function **(c)** 4-point function

Figure 7.1 – Examples of diagrams with Majorana particles in loops for the theory with the Lagrangian given in 7.1 for the process $SS \to SS$.These diagrams have been generated automatically by `GRAFED`, see section 2.5 for more details.

the fermion propagators given in sections 4.6 and 6.2 one can derive expressions for fermion traces that appear in the diagrams in figure 7.1 and simplify the conjugation matrix to recover standard traces. The trace for the contribution in 7.1c for example can be expressed generically as

$$\mathrm{Tr}\left((q\!\!\!/ + m)(q\!\!\!/ + m)C(q\!\!\!/ + m)^T(q\!\!\!/ + m)^T C\right), \tag{7.2}$$

that can be transformed into a trace that we know how to calculate, namely

$$\mathrm{Tr}\left((q\!\!\!/ + m)(q\!\!\!/ + m)(q\!\!\!/ - m)(q\!\!\!/ - m)\right), \tag{7.3}$$

where we used the relations

$$\begin{aligned} C\gamma^\mu C^\dagger &= -\left(\gamma^\mu\right)^T, \\ C \cdot C^\dagger &= 1, \\ C^\dagger &= -C. \end{aligned} \tag{7.4}$$

The simplification from 7.2 to 7.3 is performed automatically by `MARTY`, and the amplitude is indeed free of any Dirac index as all traces are calculated successfully. The test is also done for the $PP \to PP$ process that contains γ^5 matrices and also works as expected in `MARTY`.

Note that we used a very generic form of the trace in equations 7.2 and 7.3, not specifying signs and momenta. This is because the test is only about consistency, i.e. simplification of conjugation matrices and transposed γ-matrices. In the following section a precise amplitude test of Majorana interactions is presented.

7.2.2 Relative Sign of Interfering Feynman graphs (RSIF)

Once the calculation procedures can simplify the conjugation matrix in a consistent way, yielding well-defined fermion chains with no conjugation or transposed γ-matrix, one has to check the relative signs between the different contributions. The purpose is to test the RSIF at the amplitude level, this is thus a **symbolic test**. We follow [112] and in particular the 5-legs process in section 3.3, $\psi\psi \to \Phi\psi\lambda$. Again ψ is a Dirac fermion, λ a Majorana and Φ a boson. We consider the following generic Lagrangian which is obtained after removing generation indices a, b, c of equation 2.1 in [112]:

$$\begin{aligned} \mathcal{L} = &\frac{1}{2}\bar{\lambda}(i\partial\!\!\!/ - M)\lambda + \bar{\psi}(i\partial\!\!\!/ - m)\psi \\ &+ \frac{1}{2}g_i\bar{\lambda}\Gamma_i\lambda\Phi + h_i\bar{\psi}\Gamma_i\psi\Phi + k_i\left(\bar{\lambda}\Gamma_i\psi\Phi^\dagger + \text{ h.c.}\right), \end{aligned} \tag{7.5}$$

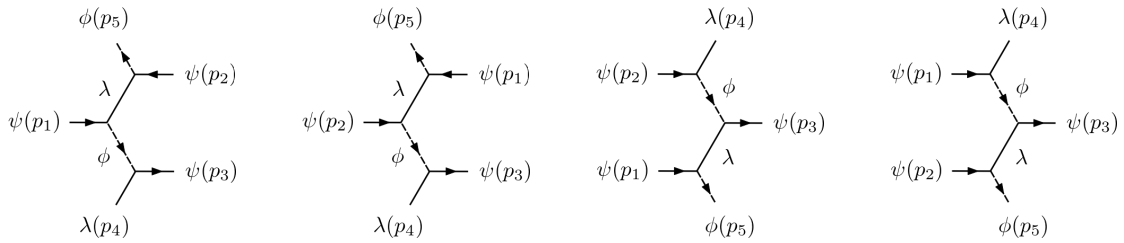
with g_i, h_i, k_i coupling constants, $\Gamma_i = 1, i\gamma^5, \gamma_\mu\gamma^5, \gamma_\mu, \sigma_{\mu\nu}$, and Φ a boson of any kind contracting in the appropriate way with Γ_i to yield a Lorentz invariant Lagrangian. We decided to specialize the test on common examples for a scalar boson ϕ and a vector boson A. Based on equation 7.5 our final Lagrangian reads

$$\begin{aligned} \mathcal{L} = &\frac{1}{2}\bar{\lambda}(i\partial\!\!\!/ - M)\lambda + \bar{\psi}(i\partial\!\!\!/ - m)\psi \\ &+ \frac{g}{2}\bar{\lambda}\lambda\phi + h\bar{\psi}\psi\phi + k\left(\bar{\psi}\lambda\phi + \text{h.c.}\right) \\ &+ \frac{g}{2}\bar{\lambda}\gamma_\mu\lambda A^\mu + h\bar{\psi}\gamma_\mu\psi A^\mu + k\left(\bar{\psi}\gamma_\mu\lambda A^\mu + \text{h.c.}\right), \end{aligned} \tag{7.6}$$

where we used the same couplings for ϕ and A as they will not interfere with each other.

Considering the fermion-violating interactions $\psi\psi \to \phi\psi\lambda$ and $\psi\psi \to A\psi\lambda$, the only relevant terms in the interaction Lagrangian are

$$\mathcal{L}_{\text{int}} \ni k\left(\bar{\psi}\lambda\phi + \bar{\psi}\gamma_\mu\lambda A^\mu + \text{h.c.}\right). \tag{7.7}$$



(a) $i\mathcal{M}_0$ **(b)** $i\mathcal{M}_1$ **(c)** $i\mathcal{M}_2$ **(d)** $i\mathcal{M}_3$

Figure 7.2 – Diagrams for the fermion-number violating process $\psi\psi \to \phi\psi\lambda$ from the Lagrangian defined in equation 7.6. Diagrams for $\psi\psi \to A\psi\lambda$ are identical replacing ϕ by A. Diagrams (b), (c) and (d) correspond respectively to diagrams 1), 2) and 3) in figure 3.3 of [112].These diagrams have been generated automatically by `GRAFED`, see section 2.5 for more details.

There are four possible diagrams for the process $\psi\psi \to \phi\psi\lambda$ as shown in figure 7.2.

Exact expressions for diagrams 7.2b, 7.2c and 7.2d are given respectively in equations 3.4a, 3.4b and 3.4c of [112] (diagram 7.2a is not given). Defining our process $\psi_1(p_1) + \psi_2(p_2) \to \psi_3(p_3) + \lambda_4(p_5) + \Phi_5(p_5)$ those expressions can be translated in our conventions, namely

$$i\mathcal{M}_1 = (+1)\frac{i}{(p_3+p_4)^2(p_5-p_1)^2}\Big[\bar{v}_1(p_1)(\not{p}_5 - \not{p}_1 + M)u_2(p_2)\Big]\cdot\Big[\bar{u}_3(p_3)v_4(p_4)\Big], \tag{7.8}$$

$$i\mathcal{M}_2 = (-1)\frac{i}{(p_2-p_4)^2(p_5-p_1)^2}\Big[\bar{u}_3(p_3)(\not{p}_1 - \not{p}_5 + M)u_1(p_1)\Big]\cdot\Big[\bar{u}_4(p_4)u_2(p_2)\Big], \tag{7.9}$$

$$i\mathcal{M}_3 = (+1)\frac{i}{(p_1-p_4)^2(p_5-p_2)^2}\Big[\bar{u}_3(p_3)(\not{p}_2 - \not{p}_5 + M)u_2(p_2)\Big]\cdot\Big[\bar{u}_4(p_4)u_1(p_1)\Big], \tag{7.10}$$

which represent the test for ϕ, where we used the trivial relation

$$\mathbb{1}' \equiv C\mathbb{1}C^\dagger = \mathbb{1}. \tag{7.11}$$

Setting the masses m and M to zero for simplicity, applying the Dirac equation and defining $s_{ij} \equiv p_i \cdot p_j$, equations 7.8 to 7.10 become

$$i\mathcal{M}_1 = (-1)\frac{i}{4s_{15}s_{34}}\Big[\bar{v}_1(p_1)\not{p}_5 u_2(p_2)\Big]\cdot\Big[\bar{u}_3(p_3)v_4(p_4)\Big], \tag{7.12}$$

$$i\mathcal{M}_2 = (+1)\frac{i}{4s_{15}s_{24}}\Big[\bar{u}_3(p_3)\not{p}_5 u_1(p_1)\Big]\cdot\Big[\bar{u}_4(p_4)u_2(p_2)\Big], \tag{7.13}$$

$$i\mathcal{M}_3 = (-1)\frac{i}{4s_{14}s_{25}}\Big[\bar{u}_3(p_3)\not{p}_5 u_2(p_2)\Big]\cdot\Big[\bar{u}_4(p_4)u_1(p_1)\Big]. \tag{7.14}$$

The output of `MARTY` for the same amplitudes is

```
1  :  -1/4*i*g^3*(p_3_%nu + p_4_%nu)*gamma_{+%nu,%gam,%eta}*lam_{l,%beta}(p_4)
  *phi_m(p_5)^(*)*psi_{i,%gam}(p_1)^(*)*psi_{k,%beta}(p_3)^(*)*psi_{j,%eta}(p_2)
  /(s_34*(s_23 + s_24 + -s_34))
```

```
2 : -1/4*i*g^3*(p_2_%nu + -p_4_%nu)*gamma_{+%nu,%gam,%eps}*lam_{l,%eta}(p_4)^(*)
  *phi_m(p_5)^(*)*psi_{i,%eps}(p_1)*psi_{k,%gam}(p_3)^(*)*psi_{j,%eta}(p_2)
  /(s_24*(s_23 + s_24 + -s_34))
```

```
3 : 1/4*i*g^3*(p_1_%mu + -p_4_%mu)*gamma_{+%mu,%gam,%eta}*lam_{l,%eps}(p_4)^(*)
  *phi_m(p_5)^(*)*psi_{i,%eps}(p_1)*psi_{k,%gam}(p_3)^(*)*psi_{j,%eta}(p_2)
  /(s_14*(s_13 + s_14 + -s_34))
```

One can see that signs are not in agreement with equations 7.12, but that the form is different. External fields carry spin indices `i,j,k,l,m`, including for the scalar field `phi(p_5)` as explained in section 6.2. Then, momentum conservation $p_1 + p_2 = p_3 + p_4 + p_5$ has to be applied in order to match both expressions, namely

$$\begin{aligned} s_{23} + s_{24} - s_{34} &= -\frac{1}{2}(-p_2 + p_3 + p_4)^2 = -\frac{1}{2}(p_1 - p_5)^2 = s_{15}, \\ s_{13} + s_{24} - s_{34} &= -\frac{1}{2}(-p_1 + p_3 + p_4)^2 = -\frac{1}{2}(p_2 - p_5)^2 = s_{25}. \end{aligned} \tag{7.15}$$

Finally using momentum conservation, massless fermions and the Dirac equation we can match

$$\begin{aligned} \bar{v}_1(\not{p}_3 + \not{p}_4)u_2 &= -\bar{v}_1\not{p}_5 u_2, \\ \bar{u}_3(\not{p}_2 - \not{p}_4)u_1 &= +\bar{u}_3\not{p}_5 u_1, \\ \bar{u}_3(\not{p}_1 - \not{p}_4)u_2 &= +\bar{u}_3\not{p}_5 u_2, \end{aligned} \tag{7.16}$$

which imply the following equivalent expressions for MARTY's output

$$i\mathcal{M}_1^{\texttt{MARTY}} = (+1)\frac{i}{4s_{15}s_{34}}\Big[\bar{v}_1(p_1)\not{p}_5 u_2(p_2)\Big]\cdot\Big[\bar{u}_3(p_3)v_4(p_4)\Big], \tag{7.17}$$

$$i\mathcal{M}_2^{\texttt{MARTY}} = (-1)\frac{i}{4s_{15}s_{24}}\Big[\bar{u}_3(p_3)\not{p}_5 u_1(p_1)\Big]\cdot\Big[\bar{u}_4(p_4)u_2(p_2)\Big], \tag{7.18}$$

$$i\mathcal{M}_3^{\texttt{MARTY}} = (+1)\frac{i}{4s_{14}s_{25}}\Big[\bar{u}_3(p_3)\not{p}_5 u_2(p_2)\Big]\cdot\Big[\bar{u}_4(p_4)u_1(p_1)\Big]. \tag{7.19}$$

We thus recover the result of [112] with a global sign that is convention dependent, i.e. the RSIF that are the only physical signs are correctly predicted by MARTY.

Similarly for A couplings, one must consider the relation

$$\gamma'_\mu \equiv C\gamma_\mu^T C^\dagger = -\gamma_\mu, \tag{7.20}$$

and the amplitude in equation 7.12 takes a relative sign. We will not derive here the analytical result and compare it to the output of MARTY as it is more involved than in the scalar case, but this can be done and has been implemented with MARTY to be tested the same way. In both cases, MARTY predicts the same relative signs as those given in [112].

This validation example demonstrates that amplitude tests are more difficult to implement because there are multiple ways to express an amplitude. Furthermore, MARTY is an automated program and its results are not always expressed in the same way as the

ones derived by hand because its procedures are slightly different, suited for a computer. One will then probably have to do algebra by hand to match the output of MARTY with an expression in the literature when looking for a match at the amplitude level. This can be done as we showed in this example but is in general a tedious task. As squared amplitudes at the numerical level are decoupled from conventions, they are much more relevant for automated testing purposes.

7.3 Squared amplitudes

Squared amplitude calculations have been presented in section 6.4. They represent very good tests for MARTY as they rely on all aspects of general calculations and are much easier to find in the literature than amplitude expressions. We present first $1 \to 2$ processes i.e. partial decay widths at tree-level and one-loop. Then, a few well-known $2 \to 2$ processes will be presented with the calculation of more diverse observables. In this section we always consider the squared amplitude averaged over incoming degrees of freedom as defined in equation 6.74 but refer to it simply by writing $|\mathcal{M}|^2$.

While the average over incoming degrees of freedom is done by MARTY automatically, the numerical calculation of cross-sections from squared amplitudes (adding kinematic factors and possibly integrating over the phase space) must be done by the user. However, as explained in section 6.4, the derivation of a cross-section from a squared amplitude is a simple numerical computation that can be quickly implemented as needed.

All calculations presented in the following have been performed using the built-in Standard Model in MARTY. See the SM_Model class or the user manual [86] for more details. To load the SM, reproduce the test and obtain expressions and diagrams for (squared-)amplitudes one simply has to write e.g. for $h \to WW$ at tree-level

```
SM_Model sm;
auto ampl = sm.computeAmplitude(TreeLevel,
    {Incoming("h"), Outgoing("W"), Outgoing(AntiPart("W"))});
Display(ampl); // Displays the expressions in the terminal
Show(ampl);   // Show the Feynman diagram(s)
auto squared = sm.computeSquaredAmplitude(ampl);
Display(squared); // Displays the expression in the terminal
```

This code sample can be straight-forwardly modified to calculate other processes.

7.3.1 Tree-level partial decay widths

For a particle of mean life-time τ, the decay rate is define as the inverse

$$\Gamma \equiv \frac{1}{\tau}, \tag{7.21}$$

and has a dimension of energy in natural units $\hbar = c = 1$. Stable particles have a zero decay rate and infinite life-time, this is in particular the case for photons and electrons in the Standard Model. For the up quark, since it cannot be observed as a free particle its life-time is not well defined. Instead we measure it for the proton (uud) and we have [35]

$$\tau_p > 10^{31} \text{ to } 10^{32} \text{ years}, \tag{7.22}$$

which means that there is for now no experimental evidence that the proton is not absolutely stable and its mean life-time is at least much larger than the age of the Universe that is of the order of 10^{10} years.

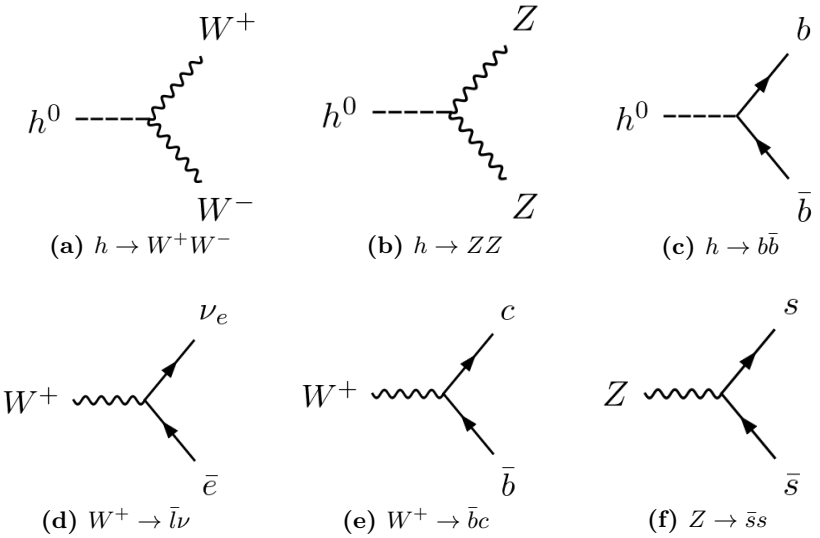
Life-time is an important observable that one can predict from a high energy physics model. In particular, the decay rate Γ_i for one given $1 \to 2$ decay mode i is simply proportional to the squared amplitude of this process and reads in the center of mass frame

$$\Gamma_i = \frac{|\vec{p}|}{8\pi M^2} |\mathcal{M}_i|^2, \tag{7.23}$$

with M the decaying particle mass, $|\vec{p}|$ the outgoing momentum and $|\mathcal{M}_i|^2$ the squared amplitude of the process. For a particle with multiple decay modes i (into 2 or more particles) the mean life-time therefore reads

$$\tau = \frac{1}{\sum_i \Gamma_i}. \tag{7.24}$$

Let us consider the calculation of Γ_i with `MARTY` for a particular mode i to obtain the partial contribution to Γ by injecting the squared amplitude in equation 7.23. We consider



(a) $h \to W^+W^-$ **(b)** $h \to ZZ$ **(c)** $h \to b\bar{b}$

(d) $W^+ \to \bar{l}\nu$ **(e)** $W^+ \to \bar{b}c$ **(f)** $Z \to \bar{s}s$

Figure 7.3 – Examples of $1 \to 2$ decay modes calculated with `MARTY` in the SM. (a) and (b) are not physical because $m_h < 2M_W, 2M_Z$ but are considered for testing purposes. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

the decays shown in figure 7.3 and assign momenta p_1, p_2, p_3 for the process $1 \to 2 + 3$

and define momenta squared $s_{ij} \equiv p_i \cdot p_j$. The results of MARTY are the following:

$$|\mathcal{M}(h \to W^+W^-)|^2 = \frac{2M_W^2 e^2}{\sin^2\theta_W}\left(1 + \frac{s_{23}^2}{2M_W^4}\right), \tag{7.25}$$

$$|\mathcal{M}(h \to ZZ)|^2 = 4M_W^2 e^2\left(1 + \frac{s_{23}^2}{2M_Z^4}\right)\left(\sin\theta_W + \frac{\sin^3\theta_W}{2\cos^2\theta_W} + \frac{\cos^2\theta_W}{2\sin^2\theta_W}\right)^2, \tag{7.26}$$

$$|\mathcal{M}(h \to b\bar{b})|^2 = \frac{e^2 m_b^2}{4M_W^2\sin^2\theta_W}\left(-12m_b^2 + 12s_{23}\right), \tag{7.27}$$

$$|\mathcal{M}(W \to \nu_e\bar{e})|^2 = \frac{2e^2}{3\sin^2\theta_W}\left(s_{23} + \frac{s_{12}s_{13} - \frac{1}{2}M_W^2 s_{23}}{M_W^2}\right), \tag{7.28}$$

$$|\mathcal{M}(W \to c\bar{b})|^2 = \frac{2e^2 V_{cb}^2}{3\sin^2\theta_W}\left(s_{23} + \frac{s_{12}s_{13} - \frac{1}{2}M_W^2 s_{23}}{M_W^2}\right), \tag{7.29}$$

$$\begin{aligned}|\mathcal{M}(Z \to s\bar{s})|^2 = &-\frac{2em_s^2\sin\theta_W}{3\cos\theta_W}f(e,\theta_W) \\ &+ \frac{4}{9}\left(s_{23} + \frac{s_{12}s_{13} - \frac{1}{2}M_Z^2 s_{23}}{M_Z^2}\right)\cdot\left(e^2\frac{\sin^2\theta_W}{\cos^2\theta_W} + \frac{1}{4}f(e,\theta_W)^2\right),\end{aligned} \tag{7.30}$$

where we neglected fermion masses in $W \to f\bar{f}'$ decays, used e as electromagnetic coupling constant, θ_W the Weinberg angle, and M_W, M_Z and m_b are the masses of W^+, Z and b respectively. We also defined a custom abbreviation $f(e,\theta_W)$ to lighten the expression:

$$f(e,\theta_W) = e\frac{\sin\theta_W}{\cos\theta_W} + 3e\frac{\cos\theta_W}{\sin\theta_W}. \tag{7.31}$$

Results have been compared to the corresponding analytical expressions in [35] and match perfectly, even though MARTY's output is not expressed in the same way.

Final remarks

One can see that some final simplifications could be done by hand e.g. for functions of θ_W, but the results are overall well factored and compact. This is what we expect from an automated computer program, the resulting expressions must only be well suited for further numerical evaluation. Analytical interpretation is possible only on very simple processes like the ones presented above and are thus rarely useful.

Expressions in equations 7.25 to 7.30 have been evaluated using the following command:

```
Evaluate(squared, eval::abbreviation);
```

If not evaluated, expressions are abbreviated by MARTY and equation 7.26 for example reads

$$|\mathcal{M}(h \to ZZ)|^2 = \left(1 + \frac{1}{2}Ab_{0022}\right)C_{0050}C_{0050}^*, \tag{7.32}$$

where MARTY has defined automatically

$$\begin{aligned}
Ab_{0022} &= \frac{s_{23}^2}{M_Z^2}, \\
C_{0050} &= 2iAb_{0018}, \\
Ab_{0018} &= \frac{1}{2}Ab_{0016}Ab_{0017}, \\
Ab_{0017} &= eM_W, \\
Ab_{0016} &= Ab_{0014} + Ab_{0015} + 2\sin\theta_W, \\
Ab_{0015} &= \frac{\cos^2\theta_W}{\sin\theta_W}, \\
Ab_{0014} &= \frac{\sin^3\theta_W}{\cos^2\theta_W}.
\end{aligned} \tag{7.33}$$

7.3.2 One-loop partial decay widths

In this section we demonstrate how to calculate partial decay widths at the one-loop level. We consider processes that are important for phenomenology, the double photon / gluon production from a Higgs particle, namely $h \to \gamma\gamma$ and $h \to gg$ loop amplitudes in the Standard Model. These two processes are in particular involved in the measurement [4] that lead to the Higgs discovery in 2012 by ATLAS [19] and CMS [18] at the LHC.

For simplicity we consider only the top quark contributions to the two processes. The diagrams are presented in figure 7.4. There is only a constant factor of difference between the top loop contributions in $h \to \gamma\gamma$ and $h \to gg$, which depend on the photon (electric charge) and gluon (color charge) couplings of the top quark. The first squared amplitude is

$$|\mathcal{M}(h \to t\bar{t} \to \gamma\gamma)|^2 \propto \left[\left(\frac{2}{3}\right)^2 \mathrm{Tr}(1)\right]^2 = \frac{16}{9}, \tag{7.34}$$

where the factor $2/3$ is the top quark electric charge and the trace is over the three quark colors and is trivially equal to 3. The diagram involving gluons has a different factor given by

$$|\mathcal{M}(h \to t\bar{t} \to gg)|^2 \propto \left[\mathrm{Tr}\left(T^A T^B\right)\right]^2 = \left[\frac{1}{2}\delta^{AB}\right]^2 = 2, \tag{7.35}$$

with T^A generators of the triplet representation of $SU(3)$ normalized by

$$\mathrm{Tr}\left(T^A T^B\right) = \frac{1}{2}\delta^{AB}. \tag{7.36}$$

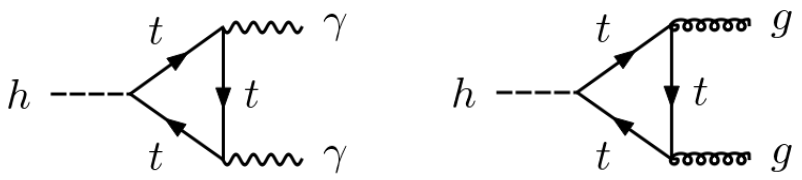
Indices A and B are in the octet representation of $SU(3)$ (gluons), which imply the identity

$$\left(\delta^{AB}\right)^2 = \delta^{AB}\delta^{AB} = \delta^{AA} = 8. \tag{7.37}$$

4. Strictly speaking we should consider $gg \to h$ instead of $h \to gg$ to reproduce the same process as in the LHC. However, the calculation is the same up to an integer factor.

Setting strong and electromagnetic couplings α_s and α_{em} to the same value, the SM thus predicts

$$\frac{|\mathcal{M}(h \to t\bar{t} \to gg)|^2}{|\mathcal{M}(h \to t\bar{t} \to \gamma\gamma)|^2} = \frac{9}{8}. \tag{7.38}$$



(a) $h \to \gamma\gamma$ **(b)** $h \to gg$

Figure 7.4 – Leading contributions, top quark loops, to the processes $h \to \gamma\gamma$ and $h \to gg$. There are 2 different diagrams for each: once the two bosons are distinguished in the process $1 \to 2+3$, the top quark loop can flow in two different directions ((123) or (132) cycles). These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

Symbolic results at one-loop in `MARTY` will not be given explicitly as they are more involved. They typically depend on integral functions $C_{ij}(m_h^2, 0, 0, m_t^2, m_t^2, m_t^2)$ introduced in section 6.3.4. We can however give the numerical result. Considering only one spatial direction without loss of generality, the momentum conservation in the center of mass frame for $h \to \gamma\gamma$ gives:

$$\begin{pmatrix} m_h \\ 0 \end{pmatrix} = \begin{pmatrix} E_\gamma \\ E_\gamma \end{pmatrix} + \begin{pmatrix} E_\gamma \\ -E_\gamma \end{pmatrix}, \tag{7.39}$$

implying

$$E_\gamma = \frac{m_h}{2}, \tag{7.40}$$

and similarly for the gluon. Injecting $M = m_h$ and $|p| = \frac{m_h}{2}$ in equation 7.23 one can derive the partial one-loop decay widths for diagrams in figure 7.4 from the squared amplitude given by `MARTY` (in natural units):

$$\Gamma(h \to t\bar{t} \to \gamma\gamma) = 7.553 \cdot 10^{-7}\ \text{GeV}, \tag{7.41}$$

$$\Gamma(h \to t\bar{t} \to gg) = 2.080 \cdot 10^{-4}\ \text{GeV}, \tag{7.42}$$

that are in perfect agreement with the analytical expressions given in [129]. To obtain the above numbers we used the following set of SM parameters:

$$\begin{aligned} \alpha_{em} &= 1/137, \\ \alpha_s &= 0.1142, \\ m_h &= 125.1, \\ m_t &= 173.34. \end{aligned} \tag{7.43}$$

Furthermore, we can check equation 7.38 with `MARTY` by imposing $\alpha_{em} = \alpha_s$, simply to check explicitly the groups factors. We found a ratio exactly equal to $1.125 = 9/8$ as expected.

7.3.3 Cross sections for 2 to 2 processes

Generalities

Before considering specific examples of $2 \to 2$ cross-sections let us present general features. We use the kinematics of $A + A \to B + B$ processes in the center of mass frame, with particles A and B of mass m_A and m_B respectively. 4-momentum conservation thus gives

$$\begin{pmatrix} E_A \\ \vec{p_i} \end{pmatrix} + \begin{pmatrix} E_A \\ -\vec{p_i} \end{pmatrix} = \begin{pmatrix} E_B \\ \vec{p_f} \end{pmatrix} + \begin{pmatrix} E_B \\ -\vec{p_f} \end{pmatrix}, \tag{7.44}$$

with 3D momenta $\vec{p_i}$ and $\vec{p_f}$. Using the on-shell condition $p^2 = m^2$ one can derive energy and momentum values in the previous equation with respect to two parameters:

- The center of mass energy E_{CM}.
- The angle θ between $\vec{p_i}$ and $\vec{p_f}$.

We deduce

$$E_A = E_B = \frac{E_{CM}}{2}, \tag{7.45}$$

$$p_i = E_{CM}\sqrt{1 - \frac{4m_A^2}{E_{CM}^2}}, \tag{7.46}$$

$$p_f = E_{CM}\sqrt{1 - \frac{4m_B^2}{E_{CM}^2}}, \tag{7.47}$$

which require

$$\begin{aligned} E_{CM} &> 2m_A, \\ E_{CM} &> 2m_B. \end{aligned} \tag{7.48}$$

From these identities one can derive kinematic variables defined by `MARTY` e.g.

$$\begin{aligned} s_{12} = p_1 \cdot p_2 &= \frac{E_{CM}^2 - 2m_A^2}{2}, \\ s_{13} = p_1 \cdot p_3 &= \frac{E_{CM}}{4}\left(1 - K \cdot \cos\theta\right), \\ s_{14} = p_1 \cdot p_4 &= \frac{E_{CM}}{4}\left(1 + K \cdot \cos\theta\right), \end{aligned} \tag{7.49}$$

where we defined

$$K \equiv \frac{p_i p_f}{E_{CM}^2} = \sqrt{1 - \frac{4m_A^2}{E_{CM}^2}}\sqrt{1 - \frac{4m_B^2}{E_{CM}^2}}. \tag{7.50}$$

The differential cross-section depending on the solid angle Ω for a $2 \to 2$ process in the center of mass frame is expressed as a function of the squared amplitude and reads

$$\frac{d\sigma}{d\Omega} = \frac{1}{64\pi^2 E_{CM}^2}\frac{|\vec{p_f}|}{|\vec{p_i}|}|\mathcal{M}|^2. \tag{7.51}$$

The total cross-section can be derived by integrating over the solid angle Ω. The integration over the polar angle ϕ is trivial and we are left with an integral over θ:

$$\sigma = \frac{1}{32\pi E_{CM}^2} \frac{|\vec{p_f}|}{|\vec{p_i}|} \int_0^\pi |\mathcal{M}|^2 \sin\theta d\theta. \tag{7.52}$$

7.3.4 $e^+e^- \to \mu^+\mu^-$ at tree-level

The $e^+e^- \to \mu^+\mu^-$ process was one of the dominant processes at LEP as it has a high probability at the typical energy range of this collider. The leading contributions come from a photon exchange at low energies and a Z boson exchange around 90 GeV at the center of mass i.e. the Z resonance. These contributions are presented in figure 7.5.

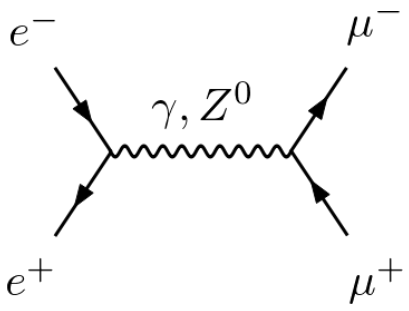


Figure 7.5 – Diagrams for the $e^+e^- \to \mu^+\mu^-$ process in the Standard Model at tree-level, neglecting the Higgs contribution. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

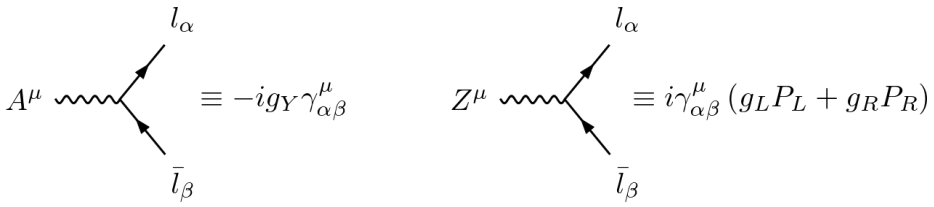
Several quantities can be calculated from this process, we present two of them in this section. The first one is the total integrated cross-section as a function of the center of mass energy E_{CM} to demonstrate `MARTY`'s ability to calculate a $2 \to 2$ cross-section and to reproduce the correct interference term between the two diagrams in particular. The second quantity that we consider is the forward-backward asymmetry generally noted $\mathcal{A}_{FB}(P)$ for a process P.

In the following calculations, we use the Breit-Wigner propagator [130] to address the Z resonance at $E_{CM} \approx M_Z$. The Z propagator denominator of momentum p therefore reads

$$\Delta_Z(p^2) = \frac{1}{p^2 - M_Z^2 + iM_Z\Gamma_Z}, \tag{7.53}$$

where the term $iM_Z\Gamma_Z$ has been added to the standard propagator, which depends on the full Z decay width Γ_Z that was defined in equation 7.21. This propagator shifts the pole to a non-real value, avoiding then the divergence at $p^2 = M_Z^2$. This comes from the fact that the Z particle is unstable with a finite lifetime $\tau_Z = 1/\Gamma_Z$. If a Z boson is mostly virtual in a diagram, i.e. that it is far from its pole $p^2 = M_Z^2$, the standard propagator can be used. Around the resonance however, the particle is real and its propagation must take into account the fact that it can decay (and not only in the final state considered in the process). The Breit-Wigner propagator introduced in equation 7.53 is a very good approximation of this effect in most cases as stated in [130] and in particular for the propagator of a single decaying particle.

We need now to define the couplings between particles used in the following calculations. Vertices with their corresponding Feynman rules are shown in figure 7.6. For simplicity fermion masses are set to zero; $m_e = m_\mu = 0$. We used the following values



(a) Photon coupling **(b)** Z boson coupling

Figure 7.6 – Feynman rules defined in our toy model to calculate $e^+e^- \to \mu^+\mu^-$ at tree-level. Leptons have (a) a vector coupling to the photon proportional to their charge -1 and (b) chirality-specific couplings to the Z boson different for left- (g_L) and right- (g_R) currents. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

for SM parameters:

$$\begin{aligned} e &= \sqrt{4\pi/137}, \\ M_Z &= 91.19\ \text{GeV}, \\ \Gamma_Z &= 2.49\ \text{GeV}, \\ \theta_W &= 0.1566\pi, \end{aligned} \tag{7.54}$$

from which the couplings are defined:

$$\begin{aligned} g_Y &= e, \\ g_L &= \frac{e}{\cos\theta_W \sin\theta_W}\left(-\frac{1}{2} + \sin^2\theta_W\right), \\ g_R &= e\tan\theta_W. \end{aligned} \tag{7.55}$$

Results for the squared amplitude

Considering the two diagrams presented in figure 7.5, we can define the two amplitudes as $i\mathcal{M}_\gamma$ and $i\mathcal{M}_Z$ for the photon and Z boson respectively. The total squared amplitude is therefore

$$\begin{aligned} |\mathcal{M}|^2 &= |i\mathcal{M}_\gamma + i\mathcal{M}_Z|^2 \\ &= |\mathcal{M}_\gamma|^2 + |\mathcal{M}_Z|^2 + \mathcal{M}_\gamma\mathcal{M}_Z^* + \mathcal{M}_\gamma^*\mathcal{M}_Z, \end{aligned} \tag{7.56}$$

in which the two first terms correspond respectively to the pure photon and pure Z contributions and the two last terms correspond to the interference between the two diagrams. This can be calculated with `MARTY`, the output (simply evaluating abbreviations)

is the following:

$$\begin{aligned}
|\mathcal{M}|^2 = & \, 4g_L^4 s_{14}s_{23}|\Delta_Z|^2 \\
& + 4g_R^4 s_{14}s_{23}|\Delta_Z|^2 \\
& + 8g_L^2 g_R^2 s_{13}s_{24}|\Delta_Z|^2 \\
& - \frac{2ig_Y^2}{s_{12}}\left(ig_L^2 s_{14}s_{23}\Delta_Z + ig_R^2 s_{14}s_{23}\Delta_Z + 2ig_L g_R s_{13}s_{24}\Delta_Z\right) \\
& + \frac{4ig_Y^2}{s_{12}}\Bigg(- ig_L g_R s_{13}s_{24}\Delta_Z^* s_{12} \\
& \qquad - ig_Y^2 \frac{s_{14}s_{23} + s_{13}s_{24}}{2s_{12}} \\
& \qquad - \frac{s_{14}s_{23}}{2}\left(ig_L^2\Delta_Z^* + ig_R^2\Delta_Z^*\right)\Bigg),
\end{aligned} \tag{7.57}$$

where we used the Breit-Wigner propagator for the Z boson defined in equation 7.53 to lighten the output. One can recognize the different contributions in the expression above, namely

$$|\mathcal{M}_\gamma|^2 = 2g_Y^4 \frac{s_{14}s_{23} + s_{13}s_{24}}{s_{12}^2}, \tag{7.58}$$

$$|\mathcal{M}_Z|^2 = 4g_L^4 s_{14}s_{23}|\Delta_Z|^2 + 4g_R^4 s_{14}s_{23}|\Delta_Z|^2 + 8g_L^2 g_R^2 s_{13}s_{24}|\Delta_Z|^2, \tag{7.59}$$

$$\mathcal{M}_\gamma\mathcal{M}_Z^* = \frac{4ig_Y^2}{s_{12}}\Bigg(- ig_L g_R s_{13}s_{24}\Delta_Z^* s_{12} - \frac{s_{14}s_{23}}{2}\left(ig_L^2\Delta_Z^* + ig_R^2\Delta_Z^*\right)\Bigg), \tag{7.60}$$

$$\mathcal{M}_\gamma^*\mathcal{M}_Z = -\frac{2ig_Y^2}{s_{12}}\left(ig_L^2 s_{14}s_{23}\Delta_Z + ig_R^2 s_{14}s_{23}\Delta_Z + 2ig_L g_R s_{13}s_{24}\Delta_Z\right), \tag{7.61}$$

from which we can check for example that indeed $\mathcal{M}_\gamma^*\mathcal{M}_Z = (\mathcal{M}_\gamma\mathcal{M}_Z^*)^*$.

Total cross-section

The total integrated cross-section has been defined in equation 7.52 as a function of the squared amplitude $|\mathcal{M}|^2$ that `MARTY` provides. It depends in particular on the center of mass energy E_{CM}. The calculation of $|\mathcal{M}|^2$ is done at the symbolic level and automatically by `MARTY` whereas equation 7.52 must be applied at the numerical level.

The numerical results obtained from `MARTY` are presented in figure 7.7. They have been compared to analytical calculations and match perfectly. One can see the Z peak around $E_{CM} = M_Z = 91.19\,\text{GeV}$ where the process is largely dominated by this resonance and peaks around $\sigma_{\gamma+Z} \approx 1.9\text{nb}$. At low energies the photon dominates with its resonance at $E_{CM} = 0$ as it is massless.

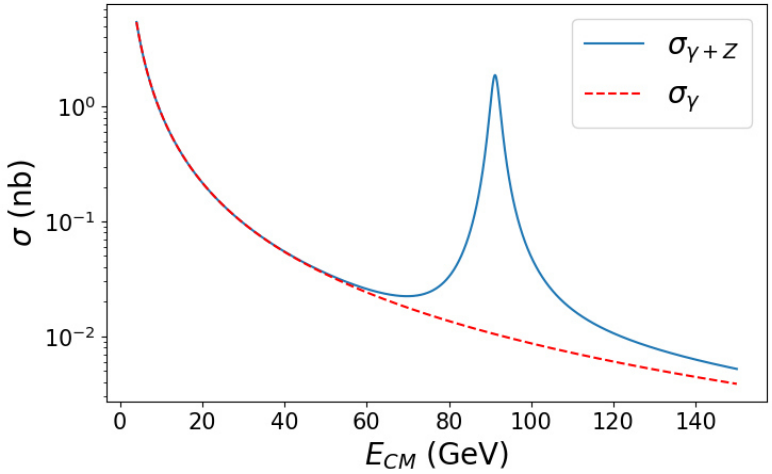
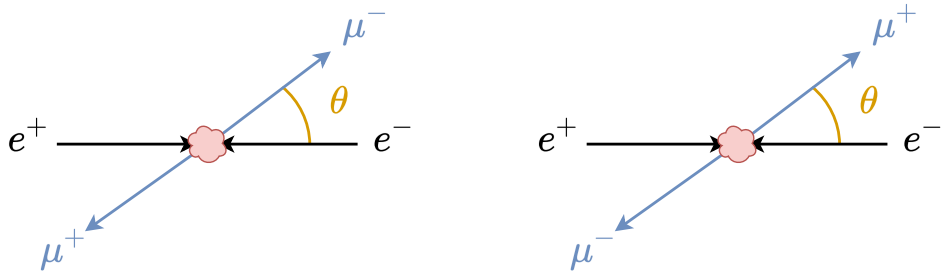


Figure 7.7 – Total integrated cross-section for $e^+e^- \to \mu^+\mu^-$ in the Standard Model as a function of the center of mass energy E_{CM} for (dashed red line) only the photon contribution and (plain blue line) the full contribution with the photon and the Z boson.

Forward-backward asymmetry

Kinematics for a $2 \to 2$ process have been defined in equation 7.44 but the orientation of the angle θ was not defined. More precisely, θ can be defined as the angle between $\vec{p_i}$ and $\vec{p_f}$ or between $\vec{p_i}$ and $-\vec{p_f}$. When the two incoming particles are indistinguishable (or equivalently the two outgoing ones) the two possible definitions of θ will produce the same results. However, when particles are distinguishable (such as particle / anti-particle) the process can be asymmetrical and depend on the definition of θ. Such an asymmetry is called a **forward-backward asymmetry** that can be measured by comparing processes such as those presented in figure 7.8.



(a) Backward process **(b)** Forward process

Figure 7.8 – Example of measurement that can be asymmetrical in the $e^+e^- \to \mu^+\mu^-$ process between the realization (a) with an angle θ between the two particles e^- and μ^- and the realization (b) with the same angle θ but this time between the particle e^- and the anti-particle μ^+.

Such an asymmetry arises when the differential cross-section has a term proportional

to $\cos\theta$, namely [5]

$$\frac{d\sigma}{d\Omega} \ni A\cos\theta, \tag{7.62}$$

with some non-vanishing value A that can be a function of the other parameters. When the differential cross-section is integrated over theta such a contribution vanishes because

$$\int_0^\pi A\cos\theta d\theta = A\int_0^\pi \cos\theta d\theta = 0. \tag{7.63}$$

However, if one is interested in the differential cross-section, there is a difference between the two scenarios presented in figure 7.8. Recalling equation 7.44, these two scenarios correspond to flipping the sign of $\cos\theta$. Therefore, the sign of the A-term defined above is also flipped. In order to measure the difference the so-called forward-backward asymmetry is defined as [131]:

$$\mathcal{A}_{FB} \equiv 2\pi\frac{\int_0^{\pi/2}\frac{d\sigma}{d\theta}d\theta - \int_{\pi/2}^{\pi}\frac{d\sigma}{d\theta}d\theta}{\sigma} \propto A, \tag{7.64}$$

where the 2π factor comes from the integration over the polar angle ϕ.

The result of the calculation of $\mathcal{A}_{FB}(e^+e^- \to \mu^+\mu^-)$ as a function of the center of mass energy E_{CM} is presented in figure 7.9, and is in agreement with [132].

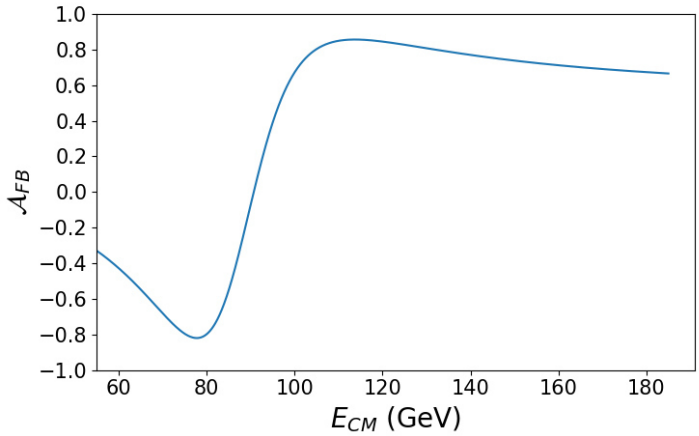


Figure 7.9 – Result of the forward-backward asymmetry in MARTY for the process $e^+e^- \to \mu^+\mu^-$ at tree-level with γ and Z boson contributions as a function of the center of mass energy E_{CM}.

5. Strictly speaking all odd powers of $\cos\theta$ generate an asymmetry but in the example presented here there is no higher order term.

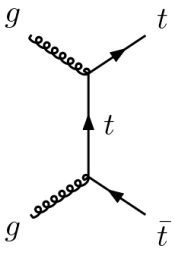
7.3.5 $gg \to t\bar{t}$ at tree-level

Motivation

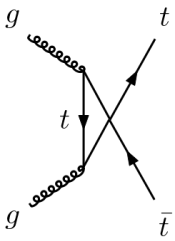
The $t\bar{t}$ production from gluon-fusion is of great relevance at the LHC for several reasons:

- The top quark is the heaviest particle of the Standard Model and should therefore be more sensitive to very high energy phenomena that we did not observed yet, i.e. new physics effects.
- The $t\bar{t}$ production cross-section is large at the LHC. To produce a $t\bar{t}$ pair one needs about $350\ \mathrm{GeV}$ and the center of mass energy at the LHC is about $13\ \mathrm{TeV}$. Taking into account the parton effect, interacting particles coming from the proton beams carry only a fraction of the total energy, typically between $10\ \mathrm{GeV}$ and $1\ \mathrm{TeV}$. This means that the production peak for $t\bar{t}$ lies in the typical energy range of incoming protons at the LHC. Details about **Parton Distribution Functions** (PDF) can be found in [133].
- The gluon-fusion channel $gg \to t\bar{t}$ at tree-level is the main contribution to the $t\bar{t}$ production at the LHC. In the Standard Model other contributions come from $q\bar{q} \to t\bar{t}$ at tree-level and finally higher-order quantities.

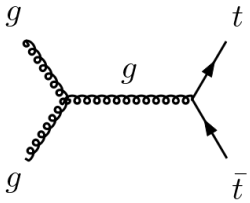
From a theoretical point of view $gg \to t\bar{t}$ is also very useful to test `MARTY`'s abilities because as the $SU(3)$ color group is non-abelian there are more diagrams contributing to the process and gauge invariance must be carefully considered. Diagrams contributing to the amplitude $i\mathcal{M}(gg \to t\bar{t})$ at tree-level are presented in figure 7.10.



(a) QED-like t-channel



(b) QED-like u-channel



(c) QCD-specific g^3-vertex contribution

Figure 7.10 – Diagrams contributing to the amplitude of $gg \to t\bar{t}$ at tree-level in the Standard Model. There are QED-like diagrams in (a) t-channel and (b) u-channel and (c) a contribution coming from the non-abelian property of the strong nuclear force different from QED. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

Complications due to Yang-Mills

The two first diagrams $i\mathcal{M}_1$ and $i\mathcal{M}_2$ in figure 7.10 are proportional to $T^A T^B$ with A and B external gluon indices, whereas the last diagram has an amplitude

$$i\mathcal{M}_3 \propto f^{ABC}T^C = -i\left[T^A, T^B\right], \tag{7.65}$$

where we see the explicit dependency on the non-abelian property of $SU(3)$

$$\left[T^A, T^B\right] \neq 0. \tag{7.66}$$

When calculating the squared amplitude, one can use the spin sum rule defined in equation 6.84 for the massless external gluons. However, the $g_{\mu\nu}$ term is just a shortcut that happens to produce correct results in QED. As the gluon is massless it has two possible polarizations ϵ_1 and ϵ_2. They are transverse so that if the gluon propagates on the z-axis we have

$$\epsilon_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \epsilon_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \tag{7.67}$$

The spin sum can then be derived explicitly and reads

$$\sum_\lambda \epsilon^\mu_\lambda(p)\epsilon^{*\nu}_\lambda(p) = \epsilon^\mu_1\epsilon^{*\nu}_1 + \epsilon^\mu_2\epsilon^{*\nu}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{7.68}$$

A general polarization $\epsilon_\lambda(p)$ can be any combination of ϵ_1 and ϵ_2 but as the spin sum must be independent on the choice of basis and Lorentz covariant one can without loss of generality calculate it using only ϵ_1 and ϵ_2.

One can see that indeed the spin sum in equation 7.68 is not equal to

$$-g_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7.69}$$

which contains two non-zero contributions on the t and z components. It can be demonstrated that the unwanted contributions in $-g_{\mu\nu}$ get contracted, during the squared amplitude calculation, to terms proportional to $[T^A, T^B]$ that vanish in QED. [6] In QCD however, these contributions do not vanish and spin sum rules given in equation 6.84 have to be adapted. It is also possible to still use $-g_{\mu\nu}$ but in this case the unwanted terms have to be compensated by new contributions. This can be done using the so-called Faddeev-Popov procedure [134] built from the BRST symmetry [135, 136]. We introduce un-physical particles in the theory to compensate our incorrect way to describe gauge

6. Using the QED Ward identity it is straight-forward to demonstrate that $-g_{\mu\nu}$ can be used for abelian gauge theories.

invariance for non-abelian symmetry groups such as $SU(3)$. These particles are called **ghosts** and are anti-commuting scalar bosons, i.e. two ghosts $c_1(X)$ and $c_2(Y)$ satisfy the relation

$$c_1(X)c_2(Y) = -c_2(Y)c_1(X), \tag{7.70}$$

which is important to determine the signs of Feynman diagrams containing such particles. Ghosts lie in the same representation as their corresponding gauge bosons and the Feynman rule for the ghost-gluon interaction in the Standard Model is presented in figure 7.11.

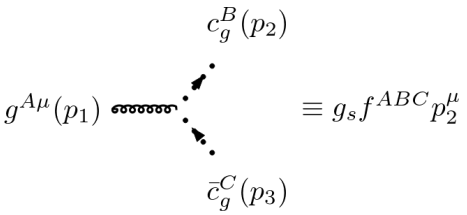
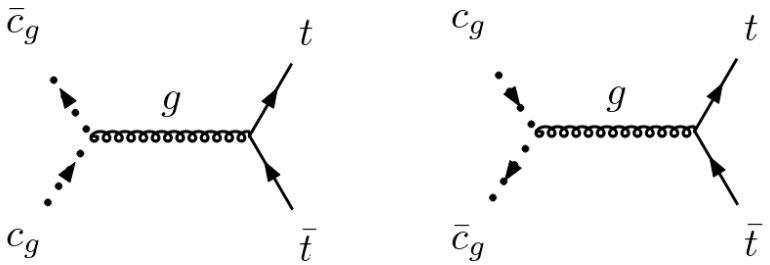


Figure 7.11 – Feynman rule for the gluon-ghost interaction in the Standard Model. g_s is the strong coupling constant. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

Ghost contributions must be considered in loops when calculating an amplitude but we will not present this kind of calculations here. We are interested in the way ghosts compensate our incorrect spin sum. Unwanted contributions arise at the squared amplitude level and must be compensated by calculating the same squared amplitude, replacing pairs of external gluons by pairs of ghosts.[7] The corrected squared amplitude that we note $|\mathcal{M}_c|^2$ therefore reads

$$|\mathcal{M}_c|^2 = |\mathcal{M}|^2 - |\mathcal{M}_g|^2 - |\bar{\mathcal{M}}_g|^2, \tag{7.71}$$

where $|\mathcal{M}|^2$ refers to the purely gluonic squared amplitude and $\mathcal{M}_g$ and $\bar{\mathcal{M}}_g$ are the two ghost contributions that are presented in figure 7.12. Ghost squared amplitudes compensate exactly the un-physical polarizations we introduced in the spin sum and we recover the physical squared amplitude $|\mathcal{M}_c|^2$.



(a) $i\mathcal{M}_g$ **(b)** $i\bar{\mathcal{M}}_g$

Figure 7.12 – Ghost contributions to the squared amplitude for $gg \to t\bar{t}$ at tree-level in the Standard Model. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

7. By construction, ghosts always appear in pairs in loops and external legs.

Results with MARTY

The full analytical calculation [8] can be found in chapter 7 of [137] and we compared our results numerically with [35]. We used as numerical inputs the following Standard Model values:

$$\begin{aligned} m_t &= 172.76 \text{ GeV}, \\ g_s &= \sqrt{4\pi \cdot 0.1179}, \end{aligned} \tag{7.72}$$

where g_s is the strong coupling constant.

The analytical results for the ghost contributions read in MARTY:

$$|\mathcal{M}_g|^2 = \frac{3g_s^4}{32}\frac{s_{13}s_{14}}{s_{12}^2}, \tag{7.73}$$

$$|\bar{\mathcal{M}}_g|^2 = \frac{3g_s^4}{32}\frac{s_{23}s_{24}}{s_{12}^2}. \tag{7.74}$$

The result for the gluon diagrams will not be written here as it is much longer. We calculated the total cross-section from the squared amplitudes $|\mathcal{M}|^2$, $|\mathcal{M}_g|^2$ and $|\bar{\mathcal{M}}_g|^2$ derived by MARTY. Results are presented in figure 7.13. We note that un-physical polarizations (present in the dashed red curve, removed by ghosts in the plain blue curve) lower the integrated cross-section by 10% around the peak and are therefore not negligible. As we previously discussed, the cross-section has indeed a peak in the LHC typical energy range for incoming gluons. This peak has its maximum around 500 GeV where the cross-section reaches 20 pb.

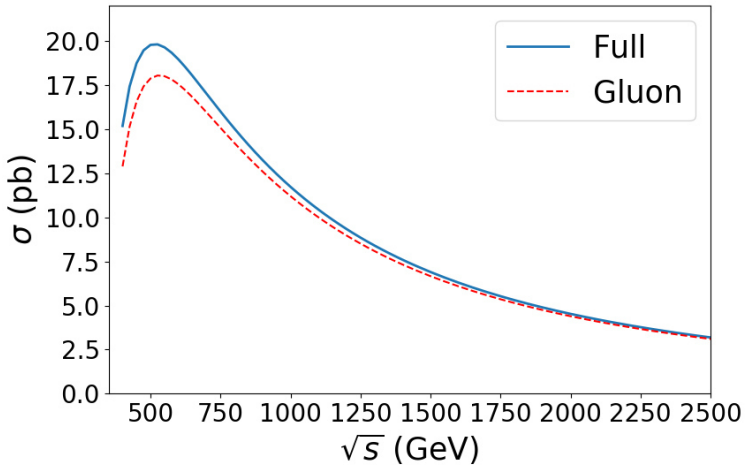


Figure 7.13 – Integrated cross-section for the process $gg \to t\bar{t}$ at tree-level in the Standard Model with (dashed red line) only the gluon contributions and (blue plain line) gluon and ghosts contributions corresponding to the final physical result.

8. A demonstration of this calculation together with all related material can be found on the website <https://marty.in2p3.fr/publications.html>.

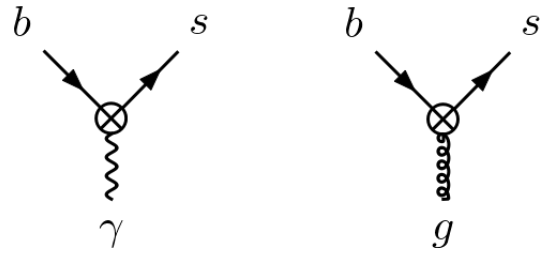
7.4 Wilson coefficients

In this section we present Wilson coefficient calculations that have been introduced in section 6.5. These calculations are particularly relevant in flavor physics in which they are used to decouple perturbative model-dependent contributions (Wilson coefficients) from non-perturbative model-independent ones (matrix elements). In the following several examples of 2- and 4-fermion operators in $b \to s$ transitions in the Standard Model and the phenomenological Minimal Supersymmetric Standard Model [65] (pMSSM) are presented.

7.4.1 Magnetic 2-fermion operators

Presentation

Effective 2-fermion operators for the $b \to s$ transition in the Standard Model Effective Field Theory (SMEFT) are presented in figure 7.14. The amplitude for such processes is



(a) $b \to s\gamma$ operator **(b)** $b \to sg$ operator

Figure 7.14 – Effective $b \to s$ 2-fermion operators in the SMEFT. The interaction represented by $\otimes$ is generic and does not assume any particular contribution. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

in general composed of several contributions. In this section we will consider only the magnetic dipole moment, namely for the photon

$$i\mathcal{M}_\gamma \ni \alpha \cdot \epsilon_\mu(q) \left(\bar{s} i\sigma^{\mu\nu} q_\nu P_X b\right), \tag{7.75}$$

with α a coefficient, q and $\epsilon_\mu(q)$ respectively the momentum and polarization tensor of the outgoing photon, $P_X \in \{P_L, P_R\}$ and

$$\sigma^{\mu\nu} = \frac{i}{2}\left[\gamma^\mu, \gamma^\nu\right]. \tag{7.76}$$

The tensor structure in equation 7.75 does not arise naturally when calculating naively the amplitude. One has to apply Dirac equation to simplify $\sigma^{\mu\nu} q_\nu$ in the fermion current and obtain new structures that depend only on p_b, p_s, γ^μ and P_X. These new structures can be identified with the result of the amplitude calculation.

For the gluon as outgoing boson the expression is similar simply adding the $SU(3)$ tensor coupling T^A:

$$i\mathcal{M}_g \ni \alpha \cdot \epsilon_\mu(q) \left(\bar{s} i\sigma^{\mu\nu} q_\nu P_X T^A b\right). \tag{7.77}$$

$b \to s\gamma$ and $b \to sg$ transitions are zero at tree-level in the SM. In general, there is no Flavor Changing Neutral Current (FCNC) in the SM i.e. a neutral particle like γ, Z^0 or h that couples to two quarks of different flavors. Hence, such processes only appear at the loop level and we will therefore present one-loop quantities in the following.

The muon anomalous magnetic moment The extraction of a coefficient such as the one presented in equation 7.75 can also be used to calculate the so-called muon anomalous magnetic moment $(g-2)_\mu$ that presents small but significant experimental deviations from the SM [60,138] (see section 1.2.3 for more details). This quantity is therefore of great importance for BSM phenomenology. The calculation is the same as for $b \to s\gamma$ and one calculates the coefficient β in the $\mu \to \mu\gamma$ amplitude defined as

$$i\mathcal{M} \ni \beta \cdot \epsilon_\mu(q) \left(\bar{\mu} i\sigma^{\mu\nu} q_\nu \mu\right). \tag{7.78}$$

Using the calculations that we present in the following for $b \to s$ transitions one can calculate for example the leading contribution to $(g-2)_\mu$, at the one-loop level and for any BSM model using `MARTY`.

The effective Hamiltonian

Considering only top quark contributions for simplicity we define the effective Hamiltonian as [139]:

$$\mathcal{H}_{eff} \equiv -\frac{4G_F}{\sqrt{2}} V_{ts}^* V_{tb} \Big[C_7(\mu) Q_7 + C_7'(\mu) Q_7' + C_8(\mu) Q_8 + C_8'(\mu) Q_8' \Big], \tag{7.79}$$

with μ an energy scale, G_F the Fermi coupling constant defined through the relation

$$\frac{G_F}{\sqrt{2}} = \frac{e^2}{8 M_W^2 \sin^2\theta_W}, \tag{7.80}$$

and operators $Q_{7/8}^{(\prime)}$ defined by [9]

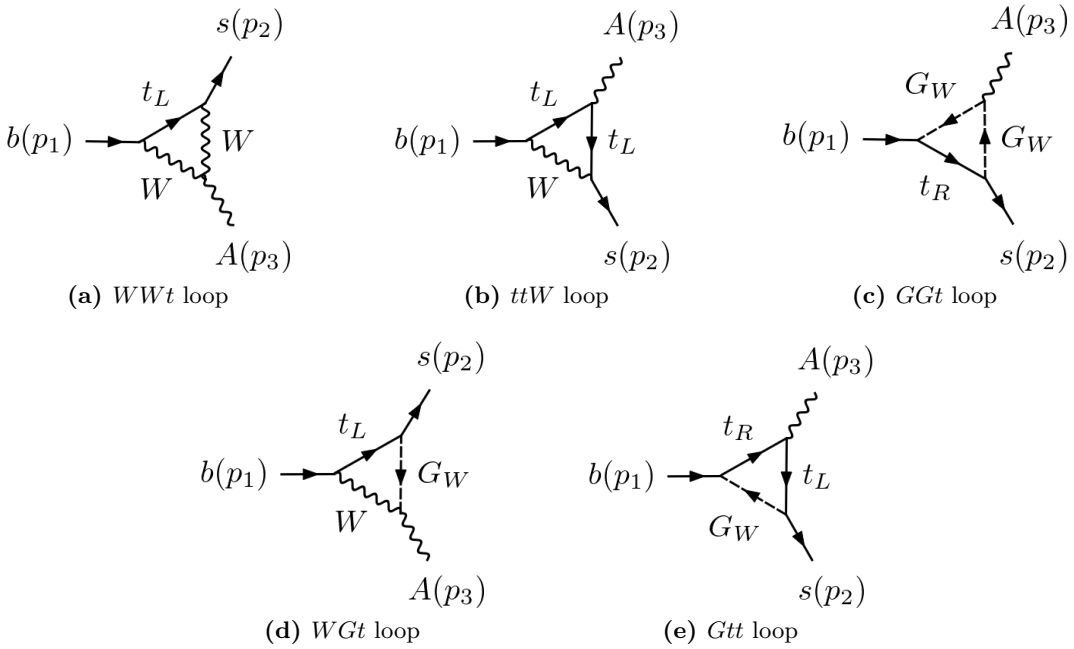
$$\langle s\gamma | \, Q_7^{(\prime)} \, | b \rangle = \frac{e}{16\pi^2} m_b \epsilon_\mu(q) \bar{s} \sigma^{\mu\nu} q_\nu P_{R(L)} b, \tag{7.81}$$

$$\langle s\gamma | \, Q_8^{(\prime)} \, | b \rangle = \frac{e}{16\pi^2} m_b \epsilon_\mu(q) \bar{s} \sigma^{\mu\nu} q_\nu P_{R(L)} T^A b, \tag{7.82}$$

with $\epsilon_\mu(q)$ the polarization tensor of the photon in $Q_7^{(\prime)}$ and the gluon in $Q_8^{(\prime)}$ respectively.

After multiplying the amplitude by i, coefficients can simply be identified in the result from the effective Hamiltonian expression using the operator definitions. In the following the leading contributions at the weak scale, i.e. one-loop calculations of $C_7^{(0)}(\mu = M_W)$ and $C_8^{(0)}(\mu = M_W)$, are presented.

9. We do not consider long distance effects and form factors in this definition, only the perturbative quantities.



(a) WWt loop **(b)** ttW loop **(c)** GGt loop

(d) WGt loop **(e)** Gtt loop

Figure 7.15 – Some of the diagrams contributing to $C_7^{(\prime)}$ in the Standard Model using the Feynman gauge. Counting separately t_L and t_R contributions there are in total 14 diagrams: 1 diagram of both types (a) and (b) and 4 diagrams of types (c), (d) and (e). There are also 10 mass correction diagrams that do not contribute to $C_7^{(\prime)}$.These diagrams have been generated automatically by `GRAFED`, see section 2.5 for more details.

C_7 in the SM

Diagrams contributing to $C_7^{(\prime)}$ in the Standard Model are presented in figure 7.15. We perform the calculation in the Feynman gauge, i.e with Goldstone boson G^+ contributions and a simple W propagator

$$\frac{-ig_{\mu\nu}}{p^2 - M_W^2}.$$

The calculation is done with on-shell massive quarks. The analytical result for $C_7^{(0)}(M_W)$ that we obtain with `MARTY` reads

$$\begin{aligned} C_7^{(0)}(M_W) = &-\frac{m_s^2}{2}\left(C_1^W + C_{11}^W + C_{12}^W + -\frac{2}{3}C_{12}^t\right) \\ &+ M_W^2\left(C_1^W + -\frac{2}{3}C_{11}^t + -C_{12}^W + -\frac{2}{3}C_{12}^t + -C_{22}^W\right) \\ &- \frac{m_t^2}{2}\Big(C_0^W + \frac{2}{3}C_0^t + C_1^W + \frac{4}{3}C_1^t + 2C_2^W \\ &\quad + \frac{2}{3}C_{11}^t + C_{12}^W + \frac{2}{3}C_{12}^t + C_{22}^W\Big), \end{aligned} \tag{7.83}$$

where integral functions have been defined as follows:

$$\begin{aligned} C_I^t &\equiv C_I(m_b^2, m_s^2, 0, m_t^2, M_W^2, m_t^2), \\ C_I^W &\equiv C_I(m_b^2, 0, m_s^2, m_t^2, M_W^2, M_W^2), \end{aligned} \tag{7.84}$$

with I representing any kind of indices. The m_s^2 contribution in equation 7.83 is usually neglected in the literature as $m_s^2 \ll m_t^2, M_W^2$. For the numerical analysis we set the following values for SM masses:

$$\begin{aligned} m_b &= 4.18 \text{ GeV}, \\ m_s &= 95 \text{ MeV}, \\ M_W &= 80.379 \text{ GeV}. \end{aligned} \tag{7.85}$$

The result of `MARTY` is plotted in figure 7.16 and has been validated using analytical expressions in [91].

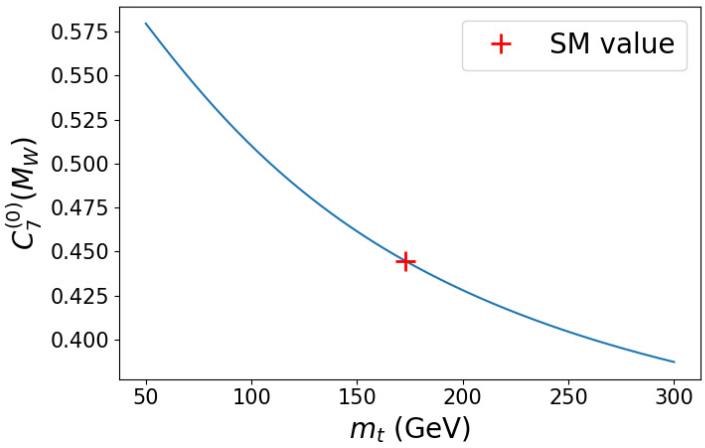


Figure 7.16 – Numerical result for top quark contributions of $C_7^{(0)}(M_W)$ as a function of the top quark mass m_t. The red cross indicates the SM value that is $C_7^{(0)}(M_W) = 0.44$ for $m_t = 173$ GeV.

C_8 in the SM

The diagram contributing to $C_8^{(\prime)}$ in the Standard Model is presented in figure 7.17. We perform this time the calculation in the unitary gauge, i.e without Goldstone boson contributions and a modified W-propagator

$$-i\frac{g^{\mu\nu} - \frac{p^\mu p^\nu}{M_W^2}}{p^2 - M_W^2}.$$

The calculation is done with on-shell massive quarks and because of the modified W-propagator we need to define

$$s_{12} \equiv p_b \cdot p_s = \frac{m_b^2 + m_s^2}{2}, \tag{7.86}$$

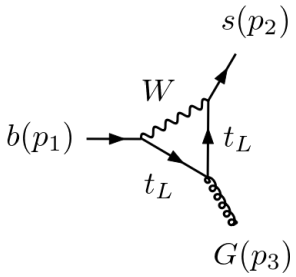


Figure 7.17 – Only diagram contributing to $C_8^{(\prime)}$ in the Standard Model using the unitary gauge. There are also 2 mass correction diagrams that do not contribute to $C_8^{(\prime)}$. This diagram has been generated automatically by `GRAFED`, see section 2.5 for more details.

where the exact expression is derived from $1 \to 2$ kinematics in the center of mass frame. The analytical result for $C_8^{(0)}(M_W)$ that we obtain with `MARTY` reads

$$\begin{aligned} C_8^{(0)}(M_W) = \frac{1}{12} &+ C_{00} + 3C_{002} - M_W^2(C_{11} + C_{12}) \\ &- \frac{m_t^2}{2}(C_0 + 2C_1 + C_2 + C_{11} + C_{12}) \\ &- s_{12}(C_2 + C_{12} + 2C_{22} + C_{122} + C_{222}) \\ &+ \frac{m_b^2}{2}(C_2 + 2C_{12} + 2C_{22} + C_{112} + 2C_{122} + C_{222}) \\ &+ \frac{m_s^2}{2}(C_2 + C_{12} + 2C_{22} + C_{222}), \end{aligned} \tag{7.87}$$

where integral functions have been defined as follows

$$C_I \equiv C_I(m_b^2, m_s^2, 0, m_t^2, M_W^2, m_t^2), \tag{7.88}$$

with I representing any kind of indices. One can see that there is now integral functions with three indices such as C_{002} because the unitary gauge introduced more momenta through the W-propagator. This represents a good test for `MARTY` as we already did the calculation of C_7 in the Feynman gauge. The result in the Feynman gauge would have been expressed differently but would be mathematically equivalent. While we present the unitary gauge for this calculation we do not recommend in general to use it as the Feynman gauge is far more convenient for simplifications in `MARTY`.

The numerical result of `MARTY` for $C_8^{(0)}(M_W)$ is plotted in figure 7.18 and has been validated using analytical expressions in [91].

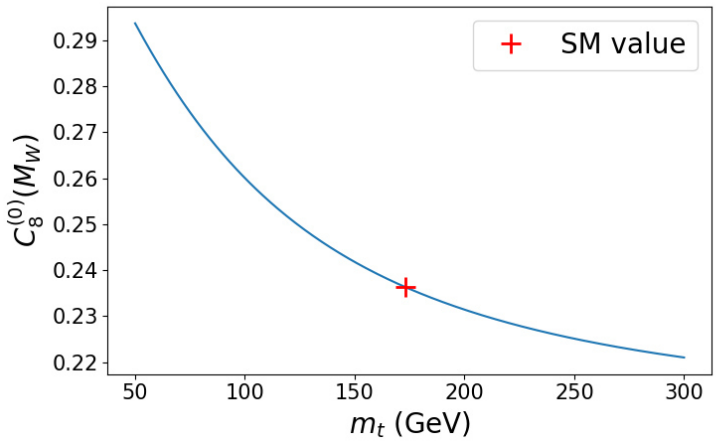


Figure 7.18 – Numerical result for the top quark contribution of $C_8^{(0)}(M_W)$ as a function of the top quark mass m_t. The red cross indicates the SM value that is $C_8^{(0)}(M_W) = 0.236$ for $m_t = 173$ GeV.

C_7 in the MSSM

The following example follows closely the section 6.2 of the main `MARTY` publication [83].

We consider in this example one of the supersymmetric contributions to C_7 i.e. diagrams with stop and charginos shown in figure 7.19. We perform the calculation on-shell in the Feynman-'t Hooft gauge. The fermion-flow reversal in the diagrams is due to fermion-number violating interactions between charginos and SM fermions. At the end of the calculation the fermion flow is regular but may get a sign due to charge conjugation matrix C which appears as we discussed in section 4.6. This sign must be correctly derived to reproduce interference patterns between diagrams.

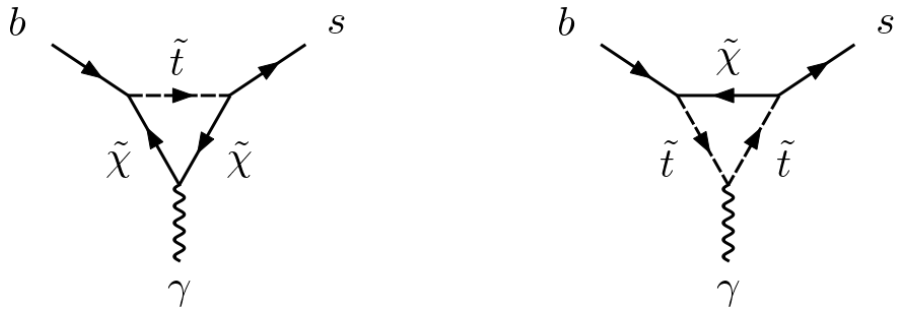


Figure 7.19 – Two types of contribution for C_7 in the pMSSM, with stops $\tilde{t}$ and charginos $\tilde{\chi}$. These diagrams have been generated using `GRAFED`, see section 2.5 for more details.

Contributions to C_7 come from chargino and stop loops and depend on two pMSSM parameters: μ (a parameter of the Higgs super-potential) and M_2 (the Wino mass). More

details on pMSSM parameters are given in [63]. The chargino mass matrix reads

$$M_\chi = \begin{pmatrix} 0 & X^T \\ X & 0 \end{pmatrix}, \tag{7.89}$$

with

$$X = \begin{pmatrix} M_2 & \sqrt{2}\sin\beta M_W \\ \sqrt{2}\cos\beta M_W & \mu \end{pmatrix}, \tag{7.90}$$

β being the angle between the two Higgs doublets' Vacuum Expectation Values (VEVs).

The stop squared mass matrix reads

$$M_{\tilde{t}}^2 = \begin{pmatrix} m_{Q_3}^2 + m_t^2 + \Delta_{\tilde{u}_L} & v(A_t^* \sin\beta - \mu y_t \cos\beta) \\ v(A_t \sin\beta - \mu^* y_t \cos\beta) & m_{u_3}^2 + m_t^2 + \Delta_{\tilde{u}_R} \end{pmatrix}, \tag{7.91}$$

where m_{Q_3} and m_{u_3} are soft supersymmetry breaking parameters, A_t is a trilinear coupling, y_t the top Yukawa and finally

$$\begin{aligned} \Delta_{\tilde{u}_L} &= \left(\frac{1}{2} - \frac{2}{3}\sin^2\theta_W\right)\cos(2\beta)M_Z^2, \\ \Delta_{\tilde{u}_R} &= \frac{2}{3}\sin^2\theta_W \cos(2\beta)M_Z^2. \end{aligned} \tag{7.92}$$

The numerical values of SM and pMSSM parameters used to evaluate C_7 are presented in table 7.1.

Parameter	Value
M_W	80.379 GeV
m_b	4.18 GeV
m_s	95 MeV
m_t	173.34 GeV
A_t	500
m_{Q_3}	1000 GeV
m_{u_3}	1000 GeV
$\tan\beta$	50
μ	$[-800, 800]$ GeV
M_2	$[-1000, 1000]$ GeV

Table 7.1 – Numerical values of supersymmetric and SM parameters used to evaluate C_7. M_2 and μ are varied in the given ranges. Other pMSSM parameters are irrelevant for the calculation presented here.

The results are shown in figures 7.20 and 7.21. `MARTY`'s output is compared with the analytical formula given in [140] and with `SuperIso` [90–93]. Numerical evaluations have been done for two different spectra. This first one (figure 7.20) is a tree-level spectrum computed by `MARTY` using GSL [141] for numerical diagonalization, and the result is compared with the analytical formula in [140]. The second spectrum (figure 7.21) is calculated by `SOFTSUSY` [142,143] with two-loop order corrections which are known to be important for the charginos [63]. For this spectrum, we compare `MARTY` with the output of `SuperIso`.

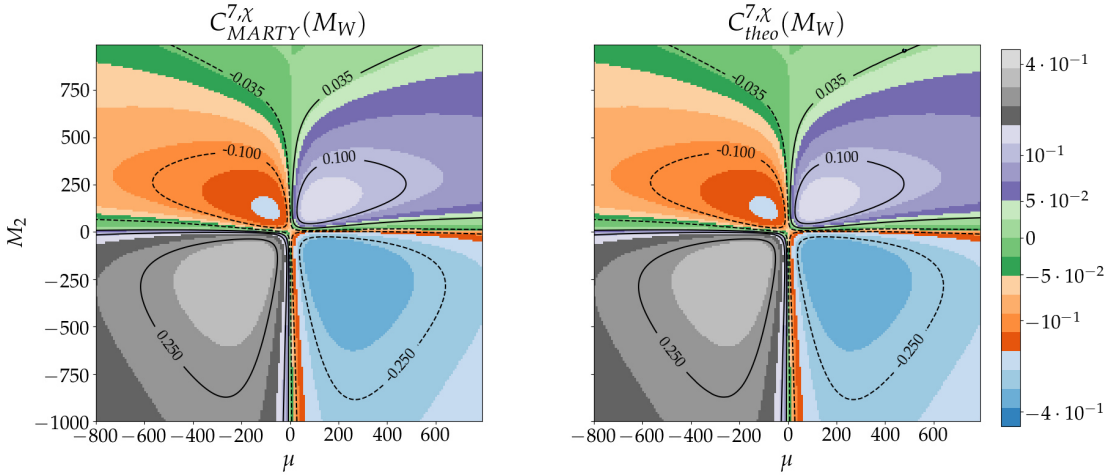


Figure 7.20 – Results for C_7 (chargino and stop contributions) in the pMSSM, from `MARTY` on the left and from the analytical formula [140] on the right, using the spectrum generated by `MARTY` at tree-level. The results match to four digits in average.

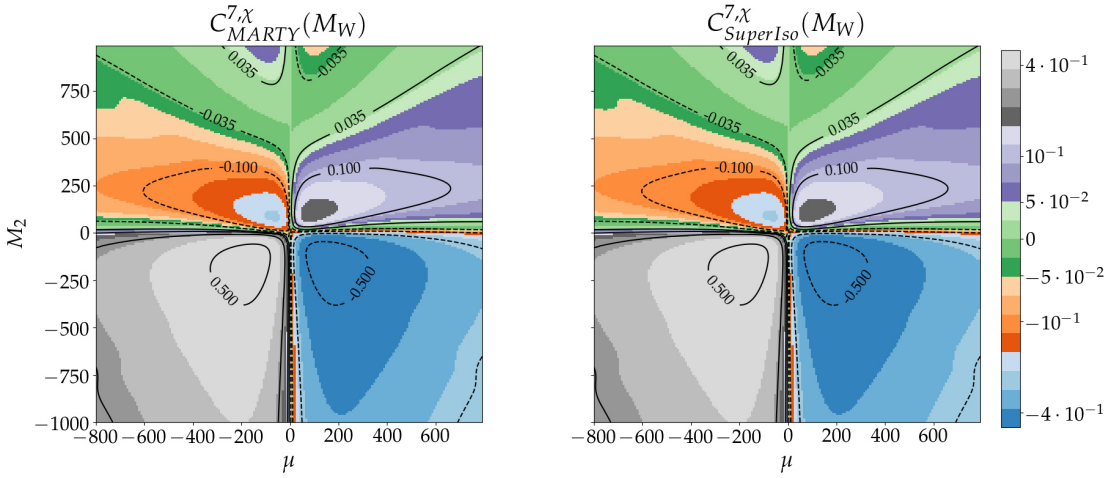


Figure 7.21 – Results for C_7 (chargino and stop contributions) in the pMSSM, from `MARTY` one the left and from the output of `SuperIso` [90–93] on the right, using the spectrum generated by `SOFTSUSY` [142, 143] with two-loop corrections. The results match to four digits in average.

7.4.2 4-fermions operators

Presentation

In this section we present the calculation of the Wilson coefficient of a 4-fermion operator in $b \to s\mu^+\mu^-$ transitions. These processes are important to describe in BSM scenarios as experimental predictions show tensions with the SM [43–48]. The effective operator describing such transitions is presented in figure 7.22.

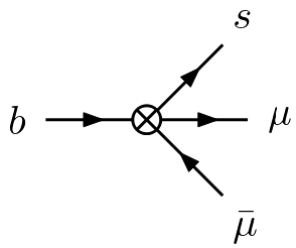


Figure 7.22 – Effective operator for $b \to s\mu\bar{\mu}$ transitions. The interaction represented by $\otimes$ is generic and does not assume any particular contribution. This diagram has been generated using `GRAFED`, see section 2.5 for more details.

C_9 in the SM

The leading order of $b \to s\mu\bar{\mu}$ is at one-loop with contributions from several diagrams. We perform the calculation in the Feynman gauge. First, diagrams contributing to C_7 such as those in figure 7.15 also contribute to C_9 multiplying by the tree-level coupling $A\mu\bar{\mu}$. These contributions also exist replacing the photon by a Z-boson, the Goldstone boson G_Z or the Higgs boson h^0. Considering only top quark contributions, there are box diagrams as shown in figure 7.23 and mass corrections that are presented in figure 7.24.

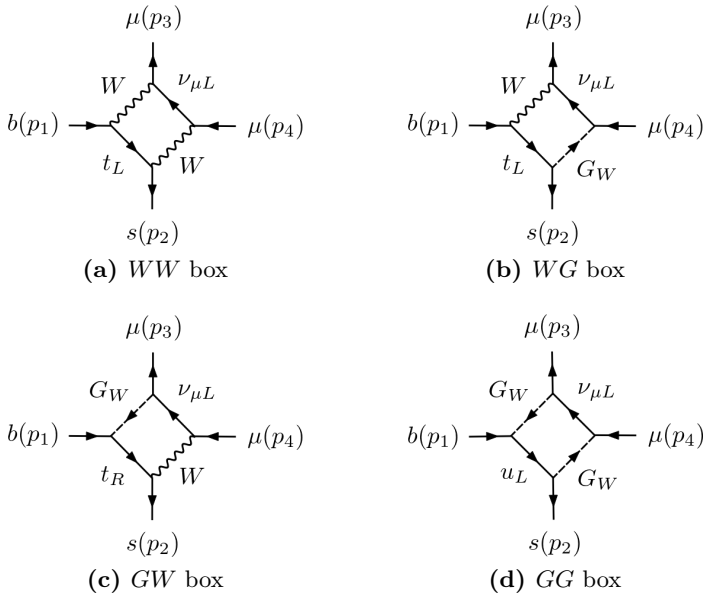
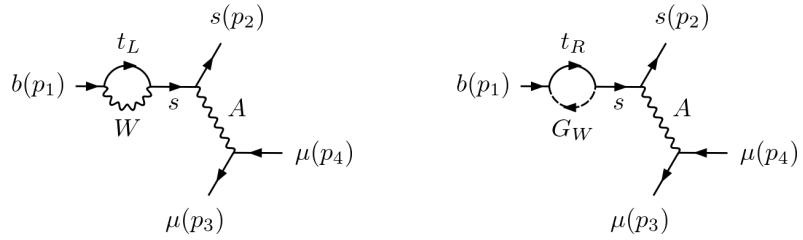


Figure 7.23 – Box diagram contributions to C_9 in the Standard Model in the Feynman gauge.These diagrams have been generated automatically by `GRAFED`, see section 2.5 for more details.

One has some liberty when calculating the C_9 Wilson coefficient and the muon mass in particular can be set to zero. As the neutrino is massless, setting $m_\mu = 0$ means that G_Z and h^0 contributions vanish because diagrams in figures 7.15 and 7.24 become proportional to m_μ or m_{ν_μ} when replacing the photon by G_Z or h^0. Similarly, Goldstone



(a) b propagator corrected by W **(b)** b propagator corrected by G

Figure 7.24 – Mass corrections contributing to C_9 in the SM in the Feynman gauge. Similar corrections to the s external line must be considered. The photon A can be replaced by other bosonic mediators such as Z, G_Z and h^0.These diagrams have been generated automatically by GRAFED, see section 2.5 for more details.

contributions in box diagrams vanish and one is left with three different types of contributions: The box diagram involving two W-bosons, penguins and mass corrections with a photon mediator and penguins and mass corrections with a Z-boson mediator.

Numerical values for parameters used in the calculation are the following:

$$\begin{aligned} M_W &= 80.379\ \text{GeV}, \\ M_Z &= 91.188\ \text{GeV}, \\ m_b &= 4.18\ \text{GeV}, \\ m_s &= 95\ \text{MeV}. \end{aligned} \tag{7.93}$$

The results of MARTY's calculations for the three contributions above as a function of the top quark mass m_t are presented in figure 7.25 and have been validated using analytical expressions in [91].

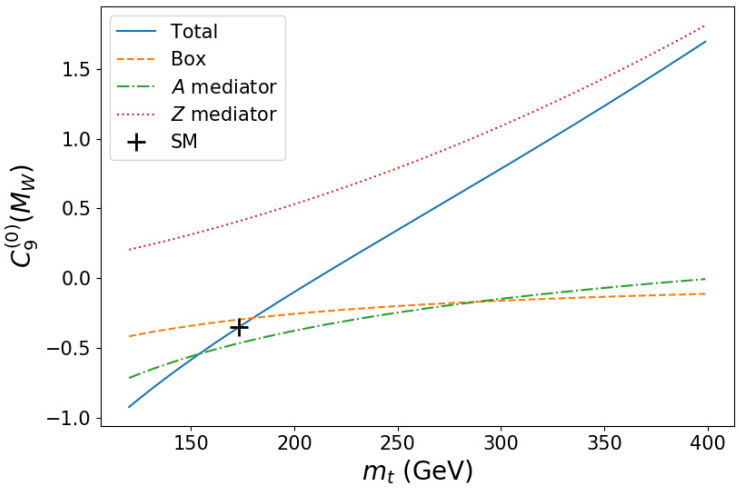


Figure 7.25 – Results for $C_9^{(0)}(M_W)$ in the Standard Model. The three types of contributions are plotted as a function of the top quark mass m_t together with the total contribution (blue plain line). A black cross represents the SM value $C_9^{(0)}(M_W) = -0.35$ at $m_t = 173$ GeV.

7.5 Performance

The calculations presented in this chapter require model building procedures for the simple scenarios, symbolic calculations and simplifications, the automated library generation, the compilation of the generated libraries and finally the numerical evaluation of the calculated quantities. [10] Each individual tree-level calculation presented in this chapter takes typically a fraction of a second, and one-loop results require more time. One-loop calculations and in particular the extraction of Wilson coefficients of 4-fermion operators can take a time of the order of one or several minutes. Running the entire test suite presented in this chapter, all the results presented above are derived in less than 3 min, without parallelization and on a standard laptop.

Besides being able to perform the general calculations at one-loop, MARTY performs them in a very short time thanks to the C++ speed and a lot of work dedicated to make CSL and MARTY as optimized as possible. When models become large, the number of diagrams for one process together with the size of expressions in the calculations grow very quickly and performance becomes crucial.

10. The numerical evaluation using the libraries generated by MARTY are done only for squared amplitude and Wilson coefficient calculations.

CHAPTER 8

Analytical results in NMFV-MSSM

8.1 Introduction

Analytical calculations in general BSM scenarios are a strong limitation to NP phenomenology. This is even more pronounced considering the fact that loop-level calculations, which are very time-consuming, are required for many observables. This is in particular the case in the MSSM in which loop-level analytical results are mostly known in simplified scenarios but not in the general MSSM. Motivated by experimental tensions with the Standard Model predictions from the LHC and Fermilab experiments, we present the full one-loop analytical contributions of the general MSSM to Wilson coefficients relevant for flavor anomalies, namely C_7 and C_9, together with the anomalous muon magnetic dipole moment $(g-2)_\mu$. The analytical results have been obtained with `MARTY` with general mixings and we restricted the numerical analysis to a particular set of Non-Minimal Flavor Violating (NMFV) MSSM scenarios.

While the Standard Model of particle physics predicts very well the majority of the measurements done in particle colliders since several decades, there are experimental tensions with the SM predictions in several channels as discussed in chapter 1. Here we are particularly interested in

— **The anomalous muon magnetic moment** $(g-2)_\mu$ that measures the quantum corrections to the tree-level relation of the coupling between the electromagnetic field and the muon spin ($g_\mu = 2$ at tree-level).

— **Flavor anomalies**, in particular in $b \to s\mu^+\mu^-$ transitions. The $b \to s\gamma$ is also phenomenologically motivated since it is well constrained experimentally.

In the following we present the full one-loop contributions to $(g-2)_\mu$ and Wilson coefficients relevant for the study of flavor anomalies in the general MSSM. Then, we present the numerical evaluation of the analytical results obtained by `MARTY` in a particular set of NMFV-MSSM scenarios.

8.1.1 $(g-2)_\mu$

As explained in chapter 1, the anomalous magnetic dipole moment is parametrized by a_μ defined in the $\mu \to \mu\gamma$ amplitude following

$$i\mathcal{M}(\mu \to \mu\gamma) \ni a_\mu \frac{ie}{4m_\mu} (\bar{\mu}\sigma^{\mu\nu}\mu) F_{\mu\nu}, \tag{8.1}$$

with m_μ the muon mass, e the electromagnetic coupling constant and $F_{\mu\nu}$ the photon field strength that we define in momentum space as

$$F_{\mu\nu} \equiv i(p_\mu \epsilon_\nu(p) - m_\nu \epsilon_\nu(p)), \tag{8.2}$$

where ϵ_μ is the photon polarization vector and p its momentum.

The combined experimental average for a_μ is [60]

$$a_\mu^{EXP} = 116\,592\,061(41) \times 10^{-11}, \tag{8.3}$$

and the SM prediction is [61]

$$a_\mu^{SM} = 116\,591\,810(43) \times 10^{-11}. \tag{8.4}$$

The tension between experiments and the theoretical prediction is therefore

$$a_\mu^{EXP} - a_\mu^{SM} = (251 \pm 59) \times 10^{-11}, \tag{8.5}$$

that corresponds to a 4.2σ tension. [1]

8.1.2 Flavor anomalies

Rare B-meson decays relying on the $b \to s$ transition at the parton level also present large tensions with the SM, in particular in the muon sector (see e.g. [43–48]). These tensions are persistent, consistent and could be the sign of Lepton Flavor Universality Violation (LFUV). Theoretically, observables are described in terms of hadronic matrix elements and Wilson coefficients. Matrix elements cannot be calculated perturbatively because they contain long-distance effect from transitions between different hadronic bound states but are model-independent. The BSM dependence is therefore contained in the Wilson coefficients only, that can be calculated with perturbation theory.

A model-independent global fit of experimental data [144] with 20 Wilson coefficients involved in $b \to s$ transitions presents two major features. In the $b \to s\gamma$ and $b \to s\mu^+\mu^-$ transitions we define the Wilson coefficients C_7 and C_9 respectively following:

$$\begin{aligned} i\mathcal{M}(b \to s\gamma) &\ni +i\frac{4G_F}{\sqrt{2}} V_{tb}V_{ts}^* \frac{e}{16\pi^2} m_b C_7 (\bar{s}\sigma^{\mu\nu} P_R b) F_{\mu\nu}, \\ i\mathcal{M}(b \to s\mu\mu) &\ni +i\frac{4G_F}{\sqrt{2}} V_{tb}V_{ts}^* \frac{e^2}{16\pi^2} C_9^\mu (\bar{s}\gamma^\mu P_L b) (\bar{\mu}\gamma_\mu \mu), \end{aligned} \tag{8.6}$$

1. Considering the lattice QCD calculations [62], the tension is however substantially reduced.

with G_F the Fermi coupling constant and V_{ij} CKM elements. From the global fit we have that the following corrections to C_7 and C_9:

$$\begin{aligned}\delta C_7 &= 0.05 \pm 0.03 \ (C_7^{SM} \approx -0.3),\\ \delta C_9^{\mu} &= -1.16 \pm 0.17 \ (C_9^{\mu,SM} \approx 4.2),\end{aligned} \tag{8.7}$$

would fit better the data than the SM. In particular, C_7 must not acquire a large shift because the $b \to s\gamma$ process is well constrained experimentally while a significant shift of C_9^{μ} (from about 25%) is the main leverage we have to address flavor anomalies.

8.1.3 NMFV-MSSM scenarios

The general SUSY-breaking Lagrangian is (see the review [63] for more details):

$$\begin{aligned}\mathcal{L}_{\text{soft}} = &-\frac{1}{2}\left(M_1\tilde{B}\tilde{B} + M_2\tilde{W}\tilde{W} + M_3\tilde{g}\tilde{g} + \text{c.c.}\right)\\ &-\left(\tilde{\bar{u}}\mathbf{a_u}\tilde{Q}H_u - \tilde{\bar{d}}\mathbf{a_d}\tilde{Q}H_d - \tilde{\bar{e}}\mathbf{a_e}\tilde{L}H_d + \text{c.c.}\right)\\ &-\tilde{Q}^{\dagger}\mathbf{m_Q^2}\tilde{Q} - \tilde{L}^{\dagger}\mathbf{m_L^2}\tilde{L} - \tilde{\bar{u}}\mathbf{m_{\bar{u}}^2}\tilde{\bar{u}}^{\dagger} - \tilde{\bar{d}}\mathbf{m_{\bar{d}}^2}\tilde{\bar{d}}^{\dagger} - \tilde{\bar{e}}\mathbf{m_{\bar{e}}^2}\tilde{\bar{e}}^{\dagger}\\ &- m_{H_u}^2 H_u^{\dagger}H_u - m_{H_d}^2 H_d^{\dagger}H_d - (bH_uH_d + \text{c.c.})\,,\end{aligned} \tag{8.8}$$

and contains in total 105 free parameters:

— 3 Higgs parameters $(m_{H_u}^2, m_{H_d}^2, b)$ that can be redefined more conventionally with $(M_A^2, \mu, \tan\beta)$.

— 3 gaugino masses M_1, M_2 and M_3.

— 54 trilinear couplings between Higgs (H_u and H_d) and sfermions in 3 general complex 3×3 matrices $\mathbf{a_u}$, $\mathbf{a_d}$ and $\mathbf{a_e}$.

— 45 sfermion mass parameters in 5 hermitian matrices $\mathbf{m_Q^2}$, $\mathbf{m_{\bar{u}}^2}$, $\mathbf{m_{\bar{d}}^2}$, $\mathbf{m_L^2}$ and $\mathbf{m_{\bar{e}}^2}$.

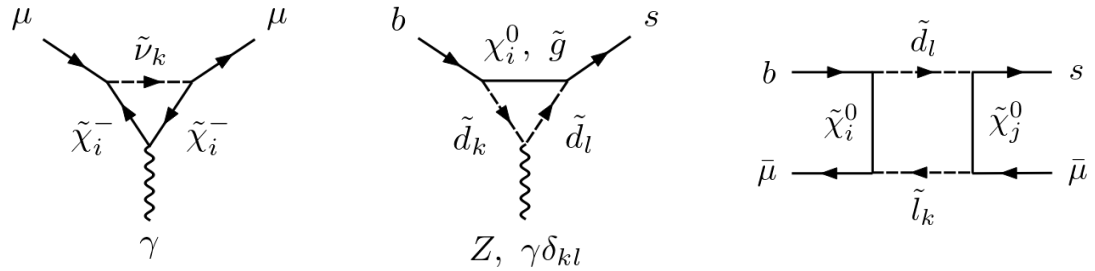
We consider in the following scenarios without universality assumption at the GUT scale. The pMSSM has 19 free parameters and forbids off-diagonal elements in SUSY-breaking matrices for trilinear couplings and sfermion masses. Vanishing off-diagonal elements for the 8 matrices defined above imply Minimal Flavor Violation (MFV), i.e. flavor violation only caused by the Standard Model Yukawa couplings. In this kind of scenarios such as the pMSSM, analytical calculations have been performed for several one-loop quantities such as C_7, C_9 [145] and $(g-2)_{\mu}$ [146].

In NMFV scenarios however, some calculations have been performed at the one-loop level (see e.g. [147]) but the general contributions to C_7, C_9 and $(g-2)_{\mu}$ are not known. In the following sections we present the methods that we used to derive analytically these quantities in the general MSSM with 105 parameters, together with their evaluation in a particular subset of NMFV scenarios with 42 parameters.

8.2 Methods

8.2.1 Theoretical calculations

In figure 8.1 examples of Feynman diagrams contributing to $(g-2)_\mu$, C_7 and C_9^μ in the general MSSM are presented.



(a) $\tilde{\chi}^+$ penguins in $(g-2)_\mu$ **(b)** $\tilde{\chi}^0/\tilde{g}$ penguins in C_7 and C_9 **(c)** $\tilde{\chi}^0$ boxes in C_9^μ

Figure 8.1 – Examples of contributions in NMFV-MSSM scenarios. Only a selection is presented, other chargino, neutralino and Higgs diagrams also contribute to C_7, C_9 and $(g-2)_\mu$.

In order to derive the full one-loop NMFV contributions to C_7, C_9^μ and $(g-2)_\mu$, a large number of Feynman diagrams must be calculated. We performed the analytical calculation in the unconstrained MSSM with general mixings. This means that diagrams must be summed over all particle families: 2 charginos $\tilde{\chi}^+_{1,2}$, 4 neutralinos $\tilde{\chi}^0_{1,2,3,4}$, 6 sleptons $\tilde{l}_{1,2,3,4,5,6}$, 6 up squarks $\tilde{u}_{1,2,3,4,5,6}$, 6 down squarks $\tilde{d}_{1,2,3,4,5,6}$ and 3 sneutrinos $\tilde{\nu}_{1,2,3}$. For the diagram shown in figure 8.1c for example, there are $4\times4\times6\times6\times2 = 1152$ independent diagrams, where the factor of 2 comes from the two possible contractions for any given ordered pair of neutralinos (counting the crossed diagrams).

We used `MARTY` to calculate automatically all the involved Feynman diagrams and extract the coefficients $(g-2)_\mu$, C_7 and C_9^μ. The number of diagrams for each contribution is presented in table 8.1. As `MARTY` counts left and right Dirac projectors P_L and P_R as independent vertices, the number of diagrams is larger that what a standard counting method would imply.

	$\tilde{\chi}_i^+$	$\tilde{\chi}_i^0$	$\tilde{g}$	H^+	H^0, A^0
$(g-2)_\mu$	96	96	0	1	2
C_7	240	96*	24*	24	0
$C_9/\gamma-$penguins	240	96*	24*	24	0
$C_9/Z-$penguins	624	1344*	240*	78	0
C_9/boxes	864	13824*	0	12	0

Table 8.1 – Number of diagrams for each contribution calculated by `MARTY`. NMFV-specific contributions are the starred orange numbers. By definition, C_7 and $(g-2)_\mu$ only receive contributions from γ-penguin diagrams. There are in total 17949 Feynman diagrams.

8.2.2 Numerical evaluation

The mathematical expressions resulting from the sum of thousands of one-loop diagrams are too large for any analytical purpose. In order to obtain predictions, `MARTY` generates a numerical C++ library containing functions evaluating the results given a general MSSM scenario. From a set of values for the SUSY-breaking parameters presented in equation 8.8, we are therefore able to evaluate the exact values of C_7, C_9^μ and $(g-2)_\mu$ at the one-loop level in the library generated by `MARTY`.

While `MARTY` also generates a tree-level spectrum generator to calculate masses and mixings from the initial model parameters, loop corrections are known to be large and we therefore use `SPheno` [77, 78] to produce a more precise spectrum including loop-level corrections and phenomenological constraints. Finally, the values of the Wilson coefficients are given to `SuperIso` [90–93] to apply Renormalization Group Equations and evolve the coefficients down to the b mass scale. This allows us to compare our results to standard analyses such as [144] that consider the Wilson coefficients at the scale of B-meson decays, i.e. $\mu = m_b$.

8.2.3 Random scan

To sample the MSSM parameter space, we used a random scan in 42 dimensions with NMFV only in the squark sector to reduce the number of free parameters. Input parameter ranges are presented in table 8.2.

Parameter	Scanned range	Parameter	Scanned range
$\tan\beta$	$[2, 60]$	$(a_e)_{33}$	$[-100, 100]$ GeV
μ	$[-100, 1000]$ GeV	$(a_{u/d})_{11}$	$[-0.1, 0.1]$ GeV
M_1, M_2	$[100, 3000]$ GeV	$(a_{u/d})_{22}$	$[100, 100]$ GeV
M_3	$[100, 7000]$ GeV	$(a_{u/d})_{33}$	$[10^{-4}, 10^4]$ GeV
M_A	$[100, 5000]$ GeV	$(m^2_Q)_{23}$	$[0, 10^3]$ GeV^2
$(m^2_Q)_{ii}$	$[10^2, 10^7]$ GeV^2	$(m^2_{\tilde{d}})_{23}$	$[0, 10^3]$ GeV^2
$(m^2_{\tilde{u}})_{ii}$	$[10^2, 10^7]$ GeV^2	$(a_u)_{ij}, i \neq j$	$[-100, 100]$ GeV
$(m^2_{\tilde{d}})_{ii}$	$[10^2, 10^7]$ GeV^2	$(a_d)_{ij}, i \neq j$	$[-100, 100]$ GeV
$(m^2_L)_{ii}$	$[10^2, 10^6]$ GeV^2		
$(m^2_{\tilde{e}})_{ii}$	$[10^2, 10^5]$ GeV^2		

Table 8.2 – Input parameters for the random scan. Specific ranges have been chosen empirically to improve the scan efficiency. There are in total 42 free parameters, which include the 19 pMSSM parameters and 14 flavor violating parameters $(m^2_Q)_{23}$, $(m^2_{\tilde{d}})_{23}$, $(a_u)_{ij}$ and $(a_d)_{ij}$ for $i \neq j$.

The scan efficiency is of about 0.05%, corresponding to physical scenarios for which `SPheno` can calculate a spectrum. For such a low efficiency there is a large bias in the selected scenarios. Consequently, we also present some posterior distributions for elements of the spectrum in figure 8.2. The scan could be refined with better constraints on the input parameters to improve the efficiency. The following analysis is therefore more a proof of principle rather than a complete phenomenological study of the MSSM parameter space. There are two visible biases in the posterior distributions of spectrum

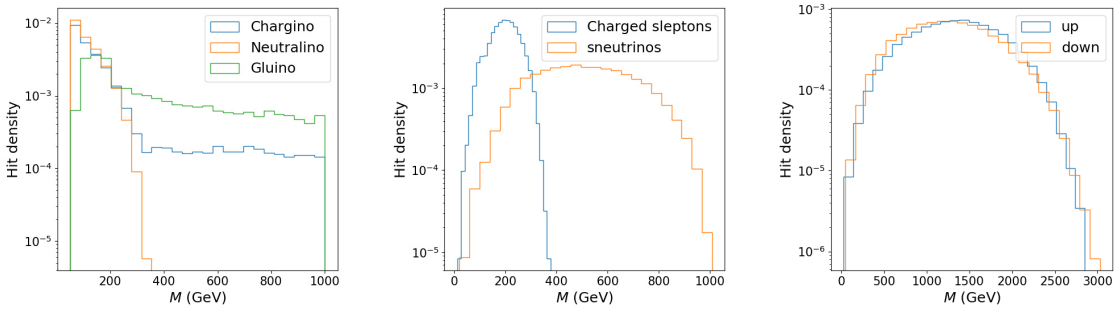


Figure 8.2 – Posterior distributions for gaugino, squark and slepton masses. For particle families, the distribution corresponds to the lightest particle of the family. Chargino and gluino mass distributions extend up to 3 TeV and 7 TeV respectively.

parameters:

— Charged sleptons are lighter than sneutrinos because the range for $m^2_{\tilde{e}}$ is smaller than the one of $m^2_{\tilde{L}}$. In particular, charged sleptons have masses around 100 GeV while sneutrinos are at the TeV scale.

— The lightest neutralino is always lighter that 400 GeV contrary to the lightest chargino. This is because we impose the condition to have a neutral Lightest Supersymmetric Particle (LSP) to be a dark matter candidate.

To improve the scan efficiency, we considered machine learning techniques to sample the parameter space. The purpose of these techniques is to create a sampling bias towards scenarios that generate valid scenarios, that therefore improves the scan efficiency. However, while these techniques can be implemented without much difficulty for the pMSSM with 19 parameters, the 43-dimensional space of the NMFV scenarios we present in this paper is too large for the machine learning-based sampling to be established. Indeed, in the absence of prior knowledge on the distribution of valid parameters, and because of the high number of dimensions, no efficient sampler can be constructed with the considered techniques such as Normalizing Flows or Hamiltonian Monte Carlo samplers.

8.3 Results

Using as input the NMFV-MSSM spectra obtained with `SPheno`, the numerical functions generated by `MARTY` evaluate the full 1-loop contributions to C_7, C_9^μ and $(g-2)_\mu$. As the scan is random, we show distributions for the different quantities that we calculated for the 70282 different scenarios. In the following, independent analyses of the Wilson coefficients and $(g-2)_\mu$ are presented. Then, the relation between these two sectors will be discussed.

8.3.1 Wilson coefficients

The distributions for the 1-loop contributions to the Wilson coefficients C_7 and C_9^μ are presented in figure 8.3. Both distributions are centered around zero as expected. While

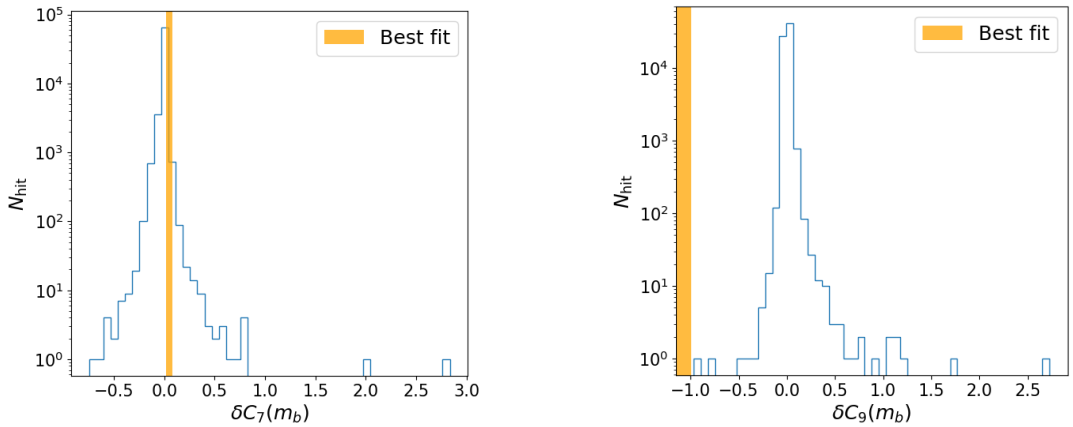


Figure 8.3 – Distribution of the Wilson coefficients δC_7 and δC_9^μ at the b mass scale. 1σ best fit regions from [144] are shown in orange.

the majority of δC_7 points are close to zero and the best fit region, many scenarios are already excluded because of a large shift to this coefficient. The analysis for δC_9^μ is different as the best fit region is shifted by -1 from the SM value. While it possible to obtain substantial C_9 shifts in our scenarios, only a handful of them predict $\delta C_9^\mu < 0.2$.

It is important to note that the best fit region for C_9^μ must not be considered as a discriminant criterion. First, the best fit depends on the set of coefficients that are considered as free parameters. Then, any scenario between the SM and the best fit can still fit better flavor observables and should be carefully considered.

A $2D$ distribution of $(\delta C_7, \delta C_9^\mu)$ is presented in figure 8.4. It is clear that the constraint on δC_7 exclude several scenarios with $\delta C_9^\mu < -0.15$. It seems possible to address both coefficients but a larger data set is required to explore the region with large negative δC_9.

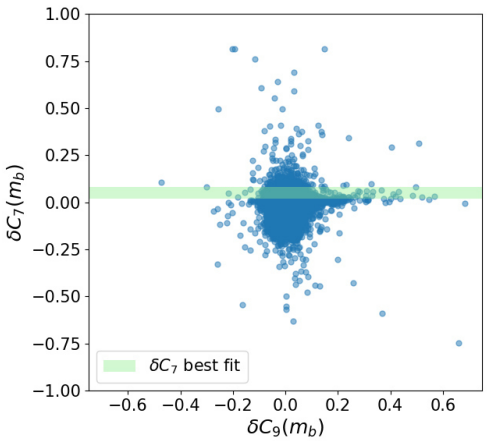


Figure 8.4 – Combined distribution of the Wilson coefficients δC_7 and δC_9^μ at the b mass scale. The best fit region for δC_7 is shown in green.

8.3.2 $(g-2)_\mu$

While our present analysis does not strictly consider NMFV parameters in the lepton sector [2] as discussed in table 8.2, the numerical results for $(g-2)_\mu$ are presented in the following. The mass distribution for charged sleptons is around the electroweak scale, i.e. a few hundred GeV (see figure 8.2). This implies significant contributions to $(g-2)_\mu$ that are shown in figure 8.5. As the experimental deviation is very small, it is not hard to address $(g-2)_\mu$ alone. The next section presents the relation between the results for $(g-2)_\mu$ and Wilson coefficients.

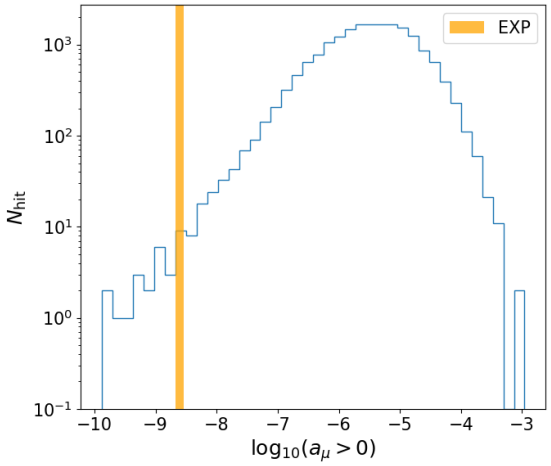


Figure 8.5 – Distribution of $\delta(g-2)_\mu$. Only scenarios with a positive shift are considered and the experimental measurement with its 1σ uncertainty is shown in orange.

8.4 Combined analysis

As shown in figure 8.1, the lepton and quark sectors are sensitive to the neutralino and chargino mass scales. However, while there are sleptons contributions in box diagrams for C_9^μ, these contributions are small and the latter coefficient is almost independent of the slepton masses. Figure 8.6 shows the dependence of $(g-2)_\mu$ and C_9^μ with respect to the relative slepton mass scale. [3]

This analysis shows that by rescaling the slepton masses (charged sleptons and sneutrinos), one can shift the value of $(g-2)_\mu$ and let Wilson coefficients C_7 and C_9^μ stable. It is therefore possible to search for a scenario that fits well flavor observables and adjust the slepton mass scale to address $(g-2)_\mu$.

2. There is no limitation for NMFV in the lepton sector, this choice has been made to reduce the number of free parameters and concentrate on flavor observables that are more difficult to address because of the C_9^μ shift.

3. C_7 is completely independent of the slepton sector.

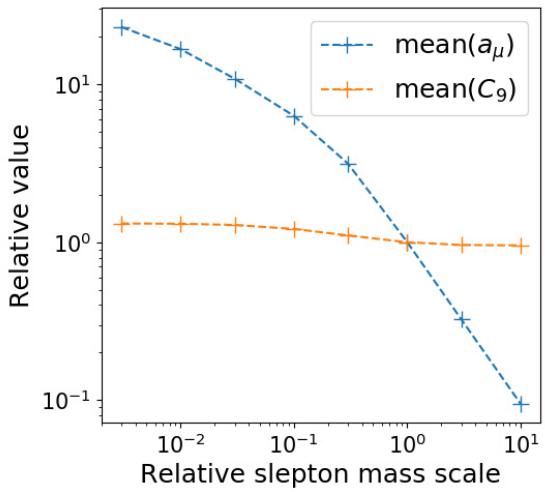


Figure 8.6 – Decoupling between Wilson coefficients and $(g-2)_\mu$ through slepton mass adjustment. The evolution of the relative mean absolute value of C_9 and $(g-2)_\mu$ for the entire data set is plotted as a function of the relative slepton mass scale. The initial, non-modified data set corresponds to the point at $(1, 1)$.

8.5 Discussion

We presented the full 1-loop analytical contributions in the general MSSM to $(g-2)_\mu$, C_7 and C_9^μ that we obtained using `MARTY`. By scanning the MSSM parameter space randomly by setting non-zero values for flavor violating parameters, we obtained using `SPheno` 70282 scenarios with their individual spectra. In these scenarios we showed that C_9^μ can be shifted towards the best fit region given in [144] but that we have only a few points that shift C_9^μ in the favored direction and let C_7 close to the SM prediction. Then, we discussed the scaling of $(g-2)_\mu$ with the slepton mass scale that allows us to address $(g-2)_\mu$ without modifying the predictions for flavor observables.

The present analysis is limited by the small sample of scenarios, i.e. 70282 model points. As a perspective, the scan should be optimized by searching a parameter set that is more likely to produce physical scenarios. In particular, by looking at the posterior distributions of the input parameters it is possible to refine the scan, improve the efficiency and generate more scenarios to analyse. Finally, experimental constraints could be studied to compare the obtained spectra with direct searches of SUSY particles, in particular from LHC measurements.

While the analysis in itself is not exhaustive and requires more statistics and phenomenological studies, the method that we presented is very promising for general BSM phenomenology. The procedure is completely model-independent and in particular `MARTY` is not limited to SUSY models or the quantities that we have evaluated. Furthermore, NMFV-MSSM scenarios generate a very large number of diverse contributions as presented in table 8.1. The fact the `MARTY` is able to handle this scenario proves that it can in practice be used in a large variety of BSM models for phenomenological one-loop analyses in many distinct domains of particle physics.

Conclusion

In this final chapter we first sum up the main project achievements through a reminder of what MARTY can bring to the high energy physics community. Then, we wrap-up this manuscript by introducing the ongoing projects that use MARTY for BSM phenomenology and the development perspectives that could make this tool even more general for elementary particle physics purposes.

MARTY, an innovation

MARTY is a general-purpose symbolic manipulation program, published under the terms of the GNU GPLv3 license, automating theoretical calculations from the Lagrangian for BSM scenarios. MARTY is specialized for analytical theoretical calculations from the Lagrangian and provide automated procedures to calculate amplitudes, squared amplitudes and Wilson coefficients for general BSM scenarios at tree-level and one-loop.

Such a level of generality has never been reached before, especially when considering the fact that MARTY is written in C++, relies only on the C++ standard library [4] and does not require a commercial software such as Mathematica [68]. Examples of results have been presented in chapter 7 to demonstrate the ability of MARTY to fulfill its requirements. This software program is the very first code for BSM phenomenology that provides all theoretical tools to go from the Lagrangian to fully simplified one-loop quantities. This generality makes MARTY very useful for all domains of particle physics such as dark matter, Higgs physics or precision physics for example, which all rely on theoretical one-loop calculations from the Lagrangian. While all the results presented in chapter 7 are derived in less than $3\ \mathrm{min}$ by MARTY and not limited to the SM or the MSSM, similar calculations in new scenarios could take weeks, months or years to perform by hand. This makes programs such as MARTY absolutely necessary for general BSM studies.

In addition, MARTY can derive Wilson coefficients for general BSM scenarios at the one-loop level. The latter feature was up to now lacking in existing codes that are usually specialized for specific scenarios such as SUSY models. Therefore, MARTY is also a solution to systematize analyses of BSM scenarios in flavor physics, in particular with respect to the flavor anomalies. In the last chapter we presented a new result in NMFV-MSSM scenarios in which we used MARTY to calculate $(g-2)_\mu$ and Wilson coefficients at the one-loop level.

MARTY is composed of three main C++ modules:

- CSL is the C++ Symbolic computation Library and is dedicated to the manipulation of general mathematical expressions in the program. We presented CSL in chapter 2.

4. At the numerical level, i.e. in C++ libraries that MARTY generates automatically, one-loop quantities require the C / Fortran LoopTools [73] library for the evaluation of integrals. However, MARTY itself does not need it to generate fully simplified analytical results.

- GRAFED is a Generating and Rendering Application for FEynman Diagrams allowing MARTY to display automatically diagrams on the screen and users to create custom ones. We presented GRAFED in section 2.5.
- The physics core of MARTY. It relies on CSL to manipulate general mathematical expressions and contains all the physics implementations for quantum fields, high energy physics models, group theory and calculations that we described respectively in chapters 3, 4, 5 and 6.

MARTY as a C++ software development project is large. It represents about 170 000 lines distributed in around 450 source files. Together with all this code, we released the documentation of CSL and MARTY under the form of Doxygen-generated [148] HTML documents and more than 300 pages of user manual. All of this material, the code and related publications can be found on the website <https://marty.in2p3/fr>. Furthermore, as we demonstrated through sample codes all along this thesis the user interface is minimal and does not require a broad knowledge of C++ to be used.

Ongoing projects

Several projects are already ongoing using MARTY as an automated tool for BSM phenomenology. As stated in chapter 1, MARTY can be applied in all domains of particle physics phenomenology. The ongoing projects based on MARTY belong to very different domains such as leptogenesis [149], dark matter and flavor physics.

In a first project [150], we aim to study the impact of a dark $U(1)_D$ sector on leptogenesis and baryonic asymmetry such as described for example in [151]. Some of these scenarios include fermion-number violating interactions that were presented in section 4.6.

In a second project [152], we are interested in the dark matter relic density that can be calculated by SuperIso Relic [94–96] from squared amplitude expressions, in SUSY and general non-SUSY BSM models.

Finally, observables relevant for flavor anomalies will be obtained [153] in several BSM scenarios such as NMFV-MSSM [63] and leptoquark models [55]. This will allow us to derive the associated flavor observables with SuperIso [90–93] and confront them to the data.

Development perspectives

MARTY has some limitations but can be extended beyond its present scope especially thanks to its independence and generality. Let us consider the example of particle types. For now only spins 0, $1/2$ and 1 can be described in MARTY. One can nevertheless implement the relevant definitions for spin $3/2$ [154] and 2 [155] particles without having to change the way MARTY describes other fields (see figure 3.2). Similarly, while we focused on quantum field theory in a $4D$ Minkowski space-time for simplicity nothing prevents a developer to also implement new relations for extra-dimensions phenomenology [156] without having to change the way MARTY works in general.

MARTY can be improved in several aspects and in particular at the numerical level. As we mentioned above, NLO-specific relations are not automated and one can simply access bare tree-level and one-loop quantities. One could implement such relations without

entering in MARTY's source code as it requires only to define appropriate generic counter-terms and find proper combinations of tree-level and one-loop results to obtain the final NLO quantity [29].

Finally, interfaces with other computer programs would be a great improvement. For example, a C++ program reading input model files of commonly used Mathematica packages such as FeynRules [69] or SARAH [76] would be a great help for users.

Final word

As it is an open-source C++ program, independent of any other external package, well-documented and fully general, we think that MARTY is excellent to be carried on by the high energy physics community for BSM theory and phenomenology. In return, MARTY would benefit from a joint effort from this community to make it even more general and interfaced with codes on the phenomenology sides as it was suggested in the introduction in figure 1.13.

Bibliography

[1] MissMJ and Cush, "Standard Model of Elementary Particles." https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg, 2019.

[2] J.J. Thomson, *XL. Cathode Rays*, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **44** (1897) 293.

[3] R. Frederick and C.C. L., *The Neutrino*, *Nature* **178** (1956) 446.

[4] A. Salam and J.C. Ward, *Weak and electromagnetic interactions*, *Il Nuovo Cimento* **11** (1959) 568.

[5] P.W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, *Physical Review Letters* **13** (1964) 508.

[6] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, *Phys. Rev. Lett.* **13** (1964) 321.

[7] G. Arnison et al., *Experimental observation of lepton pairs of invariant mass around 95 GeV/c2 at the CERN SPS collider*, *Physics Letters B* **126** (1983) 398.

[8] S. Weinberg, *A Model of Leptons*, *Physical Review Letters* **19** (1967) 1264.

[9] E.D. Bloom et al., *High-Energy Inelastic $e-p$ Scattering at 6° and 10°*, *Phys. Rev. Lett.* **23** (1969) 930.

[10] M. Breidenbach et al., *Observed Behavior of Highly Inelastic Electron-Proton Scattering*, *Phys. Rev. Lett.* **23** (1969) 935.

[11] D.P. Barber et al., *Discovery of three-jet events and a test of quantum chromodynamics at petra*, *Phys. Rev. Lett.* **43** (1979) 830.

[12] M. Kobayashi and T. Maskawa, *CP-Violation in the Renormalizable Theory of Weak Interaction*, *Progress of Theoretical Physics* **49** (1973) 652.

[13] J.J. Aubert et al., *Experimental Observation of a Heavy Particle J*, *Phys. Rev. Lett.* **33** (1974) 1404.

[14] J.E. Augustin et al., *Discovery of a Narrow Resonance in e^+e^- Annihilation*, *Phys. Rev. Lett.* **33** (1974) 1406.

[15] M.L. Perl et al., *Evidence for Anomalous Lepton Production in $e^+ - e^-$ Annihilation*, *Phys. Rev. Lett.* **35** (1975) 1489.

[16] CDF collaboration, *Observation of Top Quark Production in $\overline{p}p$ Collisions with the Collider Detector at Fermilab*, *Phys. Rev. Lett.* **74** (1995) 2626.

[17] D0 collaboration, *Observation of the Top Quark*, *Phys. Rev. Lett.* **74** (1995) 2632.

[18] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, *Physics Letters B* **716** (2012) 30.

[19] ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, *Physics Letters B* **716** (2012) 1.

[20] K.G. Wilson, *Confinement of Quarks*, *Phys. Rev. D* **10** (1974) 2445.

[21] M.B. Green and J.H. Schwarz, *Anomaly Cancellation in Supersymmetric D=10 Gauge Theory and Superstring Theory*, *Phys. Lett. B* **149** (1984) 117.

[22] A.M. Polyakov, *Fine Structure of Strings*, *Nucl. Phys. B* **268** (1986) 406.

[23] M.B. Green, J.H. Schwarz and E. Witten, *Superstring Theory. Vol. 1: Introduction*, Cambridge Monographs on Mathematical Physics (Jul, 1988).

[24] L. Smolin, *An invitation to quantum loop gravity*, World Scientific (Oct, 2004), 10.1142/9789812702340_0078, [hep-th/0408048].

[25] C. Rovelli, *Loop Quantum Gravity*, *Living Reviews in Relativity* **1** (1998) .

[26] C. Cohen-Tannoudji, B. Diu and F. Laloë, *Quantum mechanics; 1st ed*, Wiley, New York, NY (1977).

[27] R.P. Feynman, *Quantum electrodynamics*, *Frontiers in Physics* (1962) .

[28] W. Marciano and H. Pagels, *Quantum chromodynamics*, *Physics Reports* **36** (1978) 137.

[29] M.D. Schwartz, *Quantum Field Theory and the Standard Model*, Cambridge University Press (Mar, 2014).

[30] S. Weinberg, *The Quantum theory of fields. Vol. 1: Foundations*, Cambridge University Press (Jun, 2005).

[31] S. Weinberg, *The quantum theory of fields. Vol. 2: Modern applications*, Cambridge University Press (Aug, 2013).

[32] S. Weinberg, *The quantum theory of fields. Vol. 3: Supersymmetry*, Cambridge University Press (Jun, 2013).

[33] K.S. Hirata et al., *Observation of a small atmospheric ν_μ/ν_e ratio in Kamiokande*, *Physics Letters B* **280** (1992) 146.

[34] Y. Fukuda et al., *Evidence for Oscillation of Atmospheric Neutrinos*, *Physical Review Letters* **81** (1998) 1562–1567.

[35] PARTICLE DATA GROUP collaboration, *Review of Particle Physics*, *PTEP* **2020** (2020) 083C01.

[36] WMAP collaboration, *First year Wilkinson Microwave Anisotropy Probe (WMAP) observations: Preliminary maps and basic results*, *Astrophys. J. Suppl.* **148** (2003) 1 [astro-ph/0302207].

[37] WMAP collaboration, *Wilkinson Microwave Anisotropy Probe (WMAP) three year results: implications for cosmology*, *Astrophys. J. Suppl.* **170** (2007) 377 [astro-ph/0603449].

[38] WMAP collaboration, *Five-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Interpretation*, *Astrophys. J. Suppl.* **180** (2009) 330 [0803.0547].

[39] WMAP collaboration, *Seven-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Interpretation*, *Astrophys. J. Suppl.* **192** (2011) 18 [1001.4538].

[40] WMAP collaboration, *Nine-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Parameter Results*, *Astrophys. J. Suppl.* **208** (2013) 19 [1212.5226].

[41] PLANCK collaboration, *Planck 2013 results. XVI. Cosmological parameters*, *Astron. Astrophys.* **571** (2014) A16 [1303.5076].

[42] PLANCK collaboration, *Planck 2015 results. XIII. Cosmological parameters*, *Astron. Astrophys.* **594** (2016) A13 [1502.01589].

[43] R. Aaij et al., *Measurement of Form-Factor-Independent Observables in the Decay* $B^0 \to K^{*0}\mu^+\mu^-$, *Physical Review Letters* **111** (2013) .

[44] R. Aaij et al., *Angular analysis of the* $B^0 \to K^{*0}\mu^+\mu^-$*-decay using 3 fb-1 of integrated luminosity*, *Journal of High Energy Physics* **2016** (2016) .

[45] R. Aaij et al., *Test of lepton universality with* $B^0 \to K^{*0}\ell^+\ell^-$*-decays*, *Journal of High Energy Physics* **2017** (2017) .

[46] R. Aaij et al., *Measurement of CP-Averaged Observables in the* $B^0 \to K^{*0}\mu^+\mu^-$*-Decay*, *Physical Review Letters* **125** (2020) .

[47] LHCB collaboration, *Test of lepton universality in beauty-quark decays*, 2103.11769.

[48] LHCB collaboration, *Branching fraction measurements of the rare* $B_s^0 \to \phi\mu^+\mu^-$ *and* $B_s^0 \to f_2'(1525)\mu^+\mu^-$ *decays*, 2105.14007.

[49] P. Langacker, *The Physics of Heavy Z' Gauge Bosons*, *Rev. Mod. Phys.* **81** (2009) 1199 [0801.1345].

[50] A.J. Buras and J. Girrbach, *Left-handed Z' and Z FCNC quark couplings facing new* $b \to s\mu^+\mu^-$ *data*, *Journal of High Energy Physics* **2013** (2013) .

[51] B.C. Allanach, J.M. Butterworth and T. Corbett, *Collider constraints on Z' models for neutral current B-anomalies*, *Journal of High Energy Physics* **2019** (2019) .

[52] D. Marzocca, *Addressing the B-physics anomalies in a fundamental Composite Higgs model*, *Journal of High Energy Physics* **2018** (2018) .

[53] M. Chala and M. Spannowsky, *Behavior of composite resonances breaking lepton flavor universality*, *Physical Review D* **98** (2018) .

[54] A. Carmona and F. Goertz, *Recent B physics anomalies: a first hint for compositeness?*, *The European Physical Journal C* **78** (2018) .

[55] W. Buchmuller, R. Ruckl and D. Wyler, *Leptoquarks in Lepton - Quark Collisions*, *Phys. Lett. B* **191** (1987) 442.

[56] A.J. Buras, J. Girrbach-Noe, C. Niehoff and D.M. Straub, $B \to K^{(*)}\nu\bar{\nu}$ *decays in the Standard Model and beyond*, 1409.4557.

[57] J. Fuentes-Martín et al., *Vector leptoquarks beyond tree level*, *Physical Review D* **101** (2020) .

[58] J. Fuentes-Martín et al., *Vector leptoquarks beyond tree level. II.* $\mathcal{O}(\alpha_s)$ *corrections and radial modes*, *Physical Review D* **102** (2020) .

[59] J. Fuentes-Martín et al., *Vector leptoquarks beyond tree level. III. Vectorlike fermions and flavor-changing transitions*, *Physical Review D* **102** (2020) .

[60] MUON $g-2$ COLLABORATION collaboration, *Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm*, *Phys. Rev. Lett.* **126** (2021) 141801.

[61] T.o. Aoyama, *The anomalous magnetic moment of the muon in the Standard Model*, *Phys. Rept.* **887** (2020) 1 [2006.04822].

[62] S. Borsanyi et al., *Leading hadronic contribution to the muon magnetic moment from lattice qcd*, *Nature* **593** (2021) 51–55.

[63] S.P. Martin, *A supersymmetry primer*, *Advanced Series on Directions in High Energy Physics* (1998) 1–98.

[64] M. Drees, R. Godbole and P. Roy, *Theory and phenomenology of sparticles: An account of four-dimensional N=1 supersymmetry in high energy physics*, World Scientific Publishing Co. Pte. Ltd. (2004), 10.1142/4001.

[65] MSSM WORKING GROUP collaboration, *The Minimal supersymmetric standard model: Group summary report*, in *GDR (Groupement De Recherche) - Supersymetrie*, Dec, 1998 [hep-ph/9901246].

[66] P. Binetruy, *Supersymmetry: Theory, experiment and cosmology*, OUP Oxford (2006).

[67] R. Contino, *The Higgs as a Composite Nambu-Goldstone Boson*, 1005.4269.

[68] Wolfram Research, Inc., "Mathematica, Version 12.1." https://www.wolfram.com/mathematica.

[69] A. Alloul et al., *FeynRules 2.0 - A complete toolbox for tree-level phenomenology*, *Comput. Phys. Commun.* **185** (2014) 2250 [1310.1921].

[70] C. Degrande, *Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians: a proof-of-principle*, *Comput. Phys. Commun.* **197** (2015) 239 [1406.3030].

[71] C. Degrande et al., *UFO - The Universal FeynRules Output*, *Comput. Phys. Commun.* **183** (2012) 1201 [1108.2040].

[72] T. Hahn, *Generating Feynman diagrams and amplitudes with FeynArts 3*, *Computer Physics Communications* **140** (2001) 418–431.

[73] T. Hahn and M. Perez-Victoria, *Automatized one loop calculations in four-dimensions and D-dimensions*, *Comput. Phys. Commun.* **118** (1999) 153 [hep-ph/9807565].

[74] B. Ruijl, T. Ueda and J. Vermaseren, *FORM version 4.2*, 1707.06453.

[75] J.A. Evans and D. Shih, *FormFlavor Manual*, 1606.00003.

[76] F. Staub, *Exploring new models in all detail with SARAH*, *Adv. High Energy Phys.* **2015** (2015) 840780 [1503.04200].

[77] W. Porod, *SPheno, a program for calculating supersymmetric spectra, SUSY particle decays and SUSY particle production at e^+e^- colliders*, *Computer Physics Communications* **153** (2003) 275.

[78] W. Porod and F. Staub, *SPheno 3.1: extensions including flavour, CP-phases and models beyond the MSSM*, *Computer Physics Communications* **183** (2012) 2458–2469.

[79] A. Semenov, *LanHEP: A Package for the automatic generation of Feynman rules in field theory. Version 3.0*, *Comput. Phys. Commun.* **180** (2009) 431 [0805.0555].

[80] A. Pukhov et al., *CompHEP: A Package for evaluation of Feynman diagrams and integration over multiparticle phase space*, hep-ph/9908288.

[81] CompHEP collaboration, *CompHEP 4.4: Automatic computations from Lagrangians to events*, *Nucl. Instrum. Meth. A* **534** (2004) 250 [hep-ph/0403113].

[82] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *Journal of High Energy Physics* **2014** (2014) .

[83] G. Uhlrich, F. Mahmoudi and A. Arbey, *MARTY – Modern ARtificial Theoretical phYsicist: A C++ framework automating theoretical calculations Beyond the Standard Model*, *Computer Physics Communications* **264** (2021) 107928.

[84] G. Uhlrich, F. Mahmoudi and A. Arbey, *MARTY: a new C++ framework for automated symbolic calculations in Beyond the Standard Model physics*, *PoS* **ICHEP2020** (2021) 928.

[85] G. Uhlrich, F. Mahmoudi and A. Arbey, *Semi-automated BSM model building procedures in MARTY-1.1 through a 2HDM example*, *PoS* **TOOLS2020** (2021) 042.

[86] G. Uhlrich, "MARTY – User manual." https://marty.in2p3.fr/doc/marty-manual.pdf, 2020.

[87] G. Uhlrich, "Documentation of MARTY." https://marty.in2p3.fr/doc/marty/html/index.html, 2020.

[88] G. Uhlrich, "CSL – User manual." https://marty.in2p3.fr/doc/csl-manual.pdf, 2020.

[89] G. Uhlrich, "Documentation of CSL." https://marty.in2p3.fr/doc/csl/html/index.html, 2020.

[90] F. Mahmoudi, *SuperIso: A Program for calculating the isospin asymmetry of $B \to K^* \gamma$ gamma in the MSSM*, *Comput. Phys. Commun.* **178** (2008) 745 [0710.2067].

[91] F. Mahmoudi, *SuperIso v2.3: A Program for calculating flavor physics observables in Supersymmetry*, *Comput. Phys. Commun.* **180** (2009) 1579 [0808.3144].

[92] F. Mahmoudi, *SuperIso v3.0, flavor physics observables calculations: Extension to NMSSM*, *Comput. Phys. Commun.* **180** (2009) 1718.

[93] S. Neshatpour and F. Mahmoudi, *Flavour Physics with SuperIso*, *PoS* **TOOLS2020** (2021) 036 [2105.03428].

[94] A. Arbey and F. Mahmoudi, *SuperIso Relic: A Program for calculating relic density and flavor physics observables in Supersymmetry*, *Comput. Phys. Commun.* **181** (2010) 1277 [0906.0369].

[95] A. Arbey and F. Mahmoudi, *SuperIso Relic v3.0: A program for calculating relic density and flavour physics observables: Extension to NMSSM*, *Comput. Phys. Commun.* **182** (2011) 1582.

[96] A. Arbey, F. Mahmoudi and G. Robbins, *SuperIso Relic v4: A program for calculating dark matter and flavour physics observables in Supersymmetry*, *Comput. Phys. Commun.* **239** (2019) 238 [1806.11489].

[97] P. Athron et al., *GAMBIT: the global and modular beyond-the-standard-model inference tool*, *The European Physical Journal C* **77** (2017) .

[98] C. Balázs et al., *ColliderBit: a GAMBIT module for the calculation of high-energy collider observables and likelihoods*, *The European Physical Journal C* **77** (2017) .

[99] T. Bringmann et al., *DarkBit: a GAMBIT module for computing dark matter observables and likelihoods*, *The European Physical Journal C* **77** (2017) .

[100] F.U. Bernlochner et al., *FlavBit: a GAMBIT module for computing flavour observables and likelihoods*, *The European Physical Journal C* **77** (2017) .

[101] P. Athron et al., *SpecBit, DecayBit and PrecisionBit: GAMBIT modules for computing mass spectra, particle decay rates and precision observables*, *The European Physical Journal C* **78** (2018) .

[102] G.D. Martinez et al., *Comparison of statistical sampling methods with ScannerBit, the GAMBIT scanning module*, *The European Physical Journal C* **77** (2017) .

[103] J.J. Renk et al., *CosmoBit: a GAMBIT module for computing cosmological observables and likelihoods*, *Journal of Cosmology and Astroparticle Physics* **2021** (2021) 022–022.

[104] J. Frederick P. Brooks, "No Silver Bullet — Essence and Accident in Software Engineering." `http://worrydream.com/refs/Brooks-NoSilverBullet.pdf`.

[105] J.S. Cohen, *Computer Algebra and Symbolic Computation: Elementary Algorithms*, A K Peters/CRC Press (2002), 10.1201/9781439863695.

[106] The Sage Developers, "SageMath, the Sage Mathematics Software System." `https://www.sagemath.org/`.

[107] B. Stroustrup and H. Sutter, "C++ Core Guidelines." `https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`.

[108] S. Meyers, *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*, O'Reilly Media, Inc. (2014).

[109] The Qt Company, "Qt Open Source Model." `https://www.qt.io/download-open-source`, 2020.

[110] J.P. Ellis, *TikZ-Feynman: Feynman diagrams with TikZ*, *Computer Physics Communications* **210** (2017) 103–123.

[111] P. Cvitanović, *Group Theory*, Princeton University Press (2008).

[112] A. Denner et al., *Feynman rules for fermion-number-violating interactions*, *Nuclear Physics B* **387** (1992) 467.

[113] H.K. Dreiner, H.E. Haber and S.P. Martin, *Two-component spinor techniques and Feynman rules for quantum field theory and supersymmetry*, *Physics Reports* **494** (2010) 1–196.

[114] R. Feger, T.W. Kephart and R.J. Saskowski, *LieART 2.0 – A Mathematica application for Lie Algebras and Representation Theory*, *Computer Physics Communications* **257** (2020) 107490.

[115] P. Cvitanovic, *Group theory for Feynman diagrams in non-Abelian gauge theories*, *Phys. Rev. D* **14** (1976) 1536.

[116] H. Lehmann, K. Symanzik and W. Zimmermann, *Zur Formulierung quantisierter Feldtheorien*, *Il Nuovo Cimento* **1** (1955) 205.

[117] G.C. Wick, *The Evaluation of the Collision Matrix*, *Phys. Rev.* **80** (1950) 268.

[118] J.C. Romão and J.P. Silva, *A resource for signs and feynman diagrams of the standard model*, *International Journal of Modern Physics A* **27** (2012) 1230025.

[119] S. Okubo, *Casimir Invariants and Vector Operators in Simple Lie Algebra*, *J. Math. Phys.* **18** (1977) 2382.

[120] A. Denner, *Techniques for calculation of electroweak radiative corrections at the one loop level and results for W physics at LEP-200*, *Fortsch. Phys.* **41** (1993) 307 [0709.1075].

[121] G. Sulyok, *A closed expression for the UV-divergent parts of one-loop tensor integrals in dimensional regularization*, *Physics of Particles and Nuclei Letters* **14** (2017) 631–643.

[122] G. Passarino and M. Veltman, *One Loop Corrections for e^+e^- Annihilation Into $\mu^+\mu^-$ in the Weinberg Model*, *Nucl. Phys. B* **160** (1979) 151.

[123] R.K. Ellis et al., *One-loop calculations in quantum field theory: From Feynman diagrams to unitarity cuts*, *Physics Reports* **518** (2012) 141–250.

[124] T. van Ritbergen, A. Schellekens and J. Vermaseren, *Group theory factors for Feynman diagrams*, *Int. J. Mod. Phys. A* **14** (1999) 41 [hep-ph/9802376].

[125] A.J. Buras, *Weak Hamiltonian, CP violation and rare decays*, in *Les Houches Summer School in Theoretical Physics, Session 68: Probing the Standard Model of Particle Interactions*, Jun, 1998 [hep-ph/9806471].

[126] C.C. Nishi, *Simple derivation of general Fierz-type identities*, *American Journal of Physics* **73** (2005) 1160–1163.

[127] G. Uhlrich, "MARTY test suite." https://gitlab.in2p3.fr/marty/test-suite/, 2020.

[128] A. Denner et al., *Compact Feynman rules for Majorana fermions*, *Phys. Lett. B* **291** (1992) 278.

[129] G. Cacciapaglia, A. Deandrea and J. Llodra-Perez, *$H \to \gamma\gamma$ beyond the Standard Model*, *Journal of High Energy Physics* **2009** (2009) 054–054.

[130] E. Fuchs and G. Weiglein, *Breit-Wigner approximation for propagators of mixed unstable states*, *Journal of High Energy Physics* **2017** (2017) .

[131] M. Wen-Gan et al., *Forward-backward asymmetry with Z' effects in the process $e^+e^- \to \mu^+\mu^-$*, *Journal of Physics G: Nuclear and Particle Physics* **20** (1994) 1391–1398.

[132] Fernández, "Physics at LEP-1 and LEP-2." https://cds.cern.ch/record/850586/files/008cartagena1.pdf, 2000.

[133] A.D. Martin et al., *Parton distributions for the LHC*, *The European Physical Journal C* **63** (2009) 189–285.

[134] L.D. Faddeev and V.N. Popov, *Feynman Diagrams for the Yang-Mills Field*, *Phys. Lett. B* **25** (1967) 29.

[135] G. Barnich, F. Brandt and M. Henneaux, *Local BRST cohomology in gauge theories*, *Physics Reports* **338** (2000) 439–569.

[136] N. Dragon and F. Brandt, *BRST symmetry and cohomology*, in *Strings, gauge fields, and the geometry behind: The legacy of Maximilian Kreuzer*, World Scientific (2013).

[137] P. Aurenche, J.-P. Guillet and E. Pilon, "QED, QCD en pratique." https://cel.archives-ouvertes.fr/cel-01440544v2/file/cours-QCD.pdf, Oct., 2016.

[138] MUON G-2 collaboration, *Final report of the E821 muon anomalous magnetic moment measurement at BNL*, *Phys. Rev. D* **73** (2006) 072003.

[139] A.J. Buras, *Gauge Theory of Weak Decays: The Standard Model and the Expedition to New Physics Summits*, Cambridge University Press (2020), 10.1017/9781139524100.

[140] M. Ciuchini et al., *Next-to-leading QCD corrections to $B \to X_s\gamma$ in supersymmetry*, *Nuclear Physics B* **534** (1998) 3–20.

[141] GNU Project, "GNU Scientific Library (GSL)." https://www.gnu.org/software/gsl/.

[142] B. Allanach et al., *The inclusion of two-loop SUSYQCD corrections to gluino and squark pole masses in the minimal and next-to-minimal supersymmetric standard model: SOFTSUSY3.7*, *Computer Physics Communications* **219** (2017) 339–345.

[143] B. Allanach, *Softsusy: A program for calculating supersymmetric spectra*, *Computer Physics Communications* **143** (2002) 305–331.

[144] T. Hurth and F. Mahmoudi and D. Martinez Santos and S. Neshatpour, *More Indications for Lepton Nonuniversality in $b \to s\ell^+\ell^-$*, 2104.10058.

[145] C. Bobeth, A.J. Buras and T. Ewerth, *$B \to X_s\ell^+\ell^-$ in the MSSM at NNLO*, *Nuclear Physics B* **713** (2005) 522–554.

[146] S.P. Martin and J.D. Wells, *Muon anomalous magnetic dipole moment in supersymmetric theories*, *Physical Review D* **64** (2001) .

[147] A. Dedes, J. Rosiek and P. Tanedo, *Complete one-loop MSSM predictions for $B_0 \to \ell^+\ell^-$ at the Tevatron and LHC*, *Physical Review D* **79** (2009) .

[148] D. van Heesch, "Doxygen." https://www.doxygen.nl, 1997.

[149] S. Davidson, E. Nardi and Y. Nir, *Leptogenesis*, *Physics Reports* **466** (2008) 105.

[150] A. Dasgupta and G. Uhlrich, *A Boltzmann equation solver for leptogenesis in general BSM scenarios using MARTY*, *work in progress* (2021) .

[151] D. Borah, A. Dasgupta and S.K. Kang, *Leptogenesis from dark matter annihilations in scotogenic model*, 1806.04689.

[152] A. Arbey, F. Mahmoudi, M. Palmiotto and G. Uhlrich, *Dark matter relic density in general BSM scenarios*, *work in progress* (2021) .

[153] A. Boussejra, F. Mahmoudi and G. Uhlrich, *Flavor anomalies in the context of Non-Minimal Flavor Violating MSSM scenarios*, *to appear* (2021) .

[154] N.D. Christensen et al., *Simulating spin- 3/2 particles at colliders*, *The European Physical Journal C* **73** (2013) .

[155] Y.M. Zinoviev, *On massive spin 2 interactions*, *Nuclear Physics B* **770** (2007) 83.

[156] D. Hooper and S. Profumo, *Dark matter and collider phenomenology of universal extra dimensions*, *Physics Reports* **453** (2007) 29–115.