Finding Renormalization Group Invariants Using Computer Algebraic Methods

Rob Verheyen

Supervisors: Prof. dr. R.H.P. Kleiss & Prof. dr. W.J.P. Beenakker

Department of Theoretical High Energy Physics
Institute for Mathematics, Astrophysics and Particle Physics
Radboud University Nijmegen

Contents

1 Introduction

Theoretical particle physics has caught up with experimental particle physics some 40 years ago. Previously, particle colliders were making unexpected discoveries on a somewhat regular basis, while nowadays they are mostly looking for new physics that has already been predicted. With the discovery of the Higgs boson, which was already predicted back in 1964, the final piece of the Standard Model (SM) has been found. Physicists have however long been aware that the SM cannot be the complete story. There are several problems with it and there are missing ingredients such as gravity and dark matter. It is for this reason that theoreticians have been developing new physical models that include the SM along with some of these missing ingredients in those 40 years.

Unfortunately, these new models always include physics that only becomes visible at energy scales much larger than what is currently available in experiment. One of the most important of these new models is an extension to the SM known as the *Minimal Supersymmetric Standard Model* (MSSM). It adds the principle of supersymmetry to the SM in order to fix some problems and to allow for the incorporation of gravity and dark matter. Amongst other things, it can be shown to predict that the gauge couplings unify at a a scale of $\mathcal{O}(10^{16}GeV)$, which is a scale far from reachable with the current experimental setup. This unification points to the appearance of new physics at that scale, which we were able to predict even though it happens at an unreachable scale.

How is it possible to make these kinds of predictions? The theory helps out by providing the so-called *renormalization group equations* (containing the so-called β-functions). These are differential equations that describe the flow of the parameters of a theory as a function of the energy scale. The renormalization group equations can be used in several ways to attain high-scale results. A new method that is completely algebraic has been developed recently, making use of objects known as *renormalization group invariants*. These invariants allow for a method of probing the high energy scale that has multiple advantages over other methods. However, finding invariants of a set of β-functions is no easy task. In this thesis, we discuss an algebraic method of finding these renormalization group invariants. The method is functional for any set of renormalization group equations, but there will be specific emphasis on those of the MSSM, since it is currently the theory of most interest.

The next two chapters therefore consist of introductions to the SM, supersymmetry and the MSSM. Chapter 4 discusses the concept of renormalization thoroughly. It also introduces the method of renormalization group invariants in comparison with other methods. Next, in chapter 5 a new, computer-driven algebraic method is constructed that is able to find invariants of arbitrary sets of β-functions. Chapter 6 discusses the details of the implementation of the methods described in chapter 5. It describes various solutions to problems that were encountered in building a program that can find invariants. Finally, in chapter 7, the results of the application of this program to the β-functions of multiple theories such as the SM and several versions of the MSSM are shown.

1.1 Conventions

We will make use of the following conventions:

- We work in natural units, taking $\hbar = c = 1$. As a result, all dimensionful physical quantities have a mass dimension.
- We make use of the Einstein summation convention, meaning that all repeated indices are summed over unless explicitly mentioned.
- We take the Minkowski metric to be $\eta^{\mu\nu} = diag(1, -1, -1, -1)$.
- As is customary, the language of Dirac spinors is used in the introduction of the SM in chapter 2. On the other hand, chapter 3 switches to the use of Weyl spinors. The relation between those spinors is described in Appendix A.
- Vectors are denoted as $\vec{x}$ and matrices as $\mathbf{x}$. An exception are standard, well-known matrices such as the Pauli matrices σ^i where $i = (1, 2, 3)$, or the gamma matrices γ^μ where $\mu = (0, 1, 2, 3)$. We will also refer to the Pauli matrices as $\sigma^\mu = \left(\mathbf{I}, \sigma^1, \sigma^2, \sigma^3\right)$ and $\bar{\sigma} = \left(\mathbf{I}, -\sigma^1, -\sigma^2, -\sigma^3\right)$.

2 The Standard Model

In this chapter, we briefly describe the SM (SM). This is a very short summary of the contents of the SM up to the point we need for further topics. The contents of this chapter are based on [23, 24].

2.1 Quantum Field Theory

Quantum Field Theory is the standard theoretical framework used to construct quantum mechanical models for particle physics, such as the SM. Its central objects are fields, physical quantities that have a value at every spacetime point. Particles are described as excitations of those fields, and the interactions of those particles correspond with interactions of their associated fields. Table 2.1 contains the types of fields that appear in the SM.

Field	Spin	Degrees of Freedom	Mass Dimension
Scalar ϕ	0	1(2) Real(Complex)	1
Dirac ψ	$\frac{1}{2}$	4	$\frac{3}{2}$
Real Vector A_μ	1	3(2) Massive(Massless)	1

Table 2.1: Types of fields that appear in the SM

An important object in Quantum Field Theory (QFT) is the action S, the integral of the Lagrange density[1] $\mathcal{L}$ over all of space-time:

$$S = \int d^4x \; \mathcal{L}(\phi, \partial_\mu \phi) \tag{2.1}$$

where $\phi(x)$ is a quantum field, and the Lagrangian is a function of that field and its derivatives. By invoking the principle of least action, the Euler-Lagrange equations of motion can be derived:

$$\partial_\mu \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi)} - \frac{\partial \mathcal{L}}{\partial \phi} = 0 \tag{2.2}$$

We can also define the conjugate momentum of a field:

$$\pi(x) = \frac{\partial \mathcal{L}}{\partial \dot{\phi}} \tag{2.3}$$

where $\dot{\phi} = \frac{\partial \phi}{\partial t}$. This procedure can easily be extended to include multiple fields of different types.

Up until this point, the theory is a classical field theory. We switch on quantum effects by introducing a commutator for the fields and their conjugate momenta. In the Heisenberg picture:

[1] Often called the Lagrangian.

$$\left[\hat{\phi}(x), \hat{\pi}(y)\right] = i\delta^4(y-x) \tag{2.4}$$

The fields are then expressed in their Fourier modes. For instance, a real scalar field is expanded by introducing creation and annihilation operators $\hat{a}_p$ and $\hat{a}_p^\dagger$:

$$\hat{\phi}(x) = \int \frac{d^3p}{(2\pi)^3} \frac{1}{\sqrt{2\omega_p}} \left(\hat{a}_p e^{ip\cdot x} + \hat{a}_p^\dagger e^{-ip\cdot x}\right) \tag{2.5}$$

Where the form of ω_p depends on the form of the Lagrangian. The action of the creation and annihilation operators can now be interpreted as creation and annihilation of particles associated with the field.

From here, the Hamiltonian is expressed in terms of $\hat{a}_p$ and $\hat{a}_p^\dagger$. Next, the Hamiltonian is used to find a transition amplitude between states, to which perturbation theory is applied. Eventually, the calculations of such transition amplitudes can be expressed in terms of Feynman diagrams and Feynman rules.

2.2 Gauge Theory and the Standard Model

The SM contains all fields from Table 1 in some way. The spin $\frac{1}{2}$ fields correspond with the matter particles: fermions. The spin 1 fields are the gauge bosons that mediate all interactions of the SM. There is a single spin 0 field known as the Higgs field, which is discussed in the next section.

Let us start by introducing a fermion. To this end, consider the Dirac Lagrangian for a massless $s = \frac{1}{2}$ fermion:

$$\mathcal{L} = i\overline{\psi}\gamma^\mu \partial_\mu \psi \tag{2.6}$$

where γ^μ are the Dirac matrices and $\overline{\psi} \equiv \psi^\dagger \gamma^0$. The fundamental interactions of the SM are then usually introduced by the *gauge principle*. If the field ψ is an N-tuple, then this Lagrangian is invariant under global (x-independent) unitary transformations from the group $SU(N)$:

$$\psi \to e^{i\alpha^a T^a}\psi \tag{2.7}$$

where α^a are real numbers and summation over a is implied. T^a is the set of N^2-1 generators of $SU(N)$ in the fundamental representation. This symmetry is then 'gauged' by demanding invariance of the Lagrangian with respect to a localized version of this transformation:

$$\psi \to e^{i\alpha^a(x) T^a}\psi \tag{2.8}$$

This transformation is now no longer a symmetry of the Dirac Lagrangian, which can be fixed by introducing a new (gauge) field. The derivative ∂_μ is modified

to a covariant derivative $D_\mu = \partial_\mu - igA^a_\mu T^a$, and a kinetic term for the new field is introduced:

$$\mathcal{L} = \overline{\psi}\gamma^\mu D_\mu \psi - \frac{1}{4} F^a_{\mu\nu} F^{a,\mu\nu} \quad \text{where} \quad F^a_{\mu\nu} \equiv \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu \quad (2.9)$$

The parameters f^{abc} are the structure constants of the group in question. The extra terms in the derivative are interaction terms that correspond with the force associated with the gauge group. The number of new gauge fields that are introduced is equal to the number of generators. For instance, the theory of QCD demands global $SU(3)$ gauge invariance, resulting in 8 new gauge fields.

The gauge group of the SM is $SU(3)\times SU(2)\times U(1)$. The group $U(1)$ is a special, simpler case of the above $SU(N)$ gauging process. It has a single, constant generator and works on singlet fields. $U(1)$ is an Abelian group, meaning the structure constants are $f^{abc} = 0$.

The SM contains more than a single fermionic field with a single gauge interaction. All fermionic and bosonic fields are introduced in Table 2.2.

Field	$SU(3) \times SU(2) \times U(1)$
$Q_L = \begin{pmatrix} u_L & d_L \end{pmatrix}$	$(\mathbf{3}, \mathbf{2}, \frac{1}{6})$
u_R	$(\mathbf{3}, \mathbf{1}, \frac{2}{3})$
d_R	$(\mathbf{3}, \mathbf{1}, \frac{-1}{3})$
$L_L = \begin{pmatrix} \nu_L & e_L \end{pmatrix}$	$(\mathbf{1}, \mathbf{2}, \frac{-1}{2})$
e_R	$(\mathbf{1}, \mathbf{1}, -1)$
g	$(\mathbf{8}, \mathbf{1}, 0)$
W	$(\mathbf{1}, \mathbf{3}, 0)$
B	$(\mathbf{1}, \mathbf{1}, 0)$
ϕ	$(\mathbf{1}, \mathbf{2}, \frac{1}{2})$

Table 2.2: SM Field Content

Table 2 contains the field content of the SM. For the $SU(N)$ gauge groups, the fields are either in the fundamental representation of N dimensions denoted by $\boldsymbol{N}$, or in the trivial representation $\mathbf{1}$. Any field in the fundamental representation can be viewed as an n-tuple that transforms with the unitary matrices associated with the fundamental representation of the group. Any field in the trivial representation is a so called singlet under that group transformation. The representations of $U(1)$ are characterized by a quantum number, called the hypercharge Y.

There are a total of five fermionic fields[2]. However, there are actually three versions of every one of these fermionic fields. These are known as *families,* all having the exact same properties, with the exception of their mass.

The fields u and d are the *quarks* of *up-type* and *down-type*, characterized by their interaction with the strong force of $SU(3)$. The field Q_L is a doublet

[2]We choose to disregard the right-handed neutrino.

under $SU(2)$ transformations, containing the *left-handed* version of the u_R and d_R field, which are right-handed. This distinction exists simply because the $SU(2)$ gauge bosons only couple to left-handed fields, so the right-handed fields are $SU(2)$ singlets. L_L and e_R are separate for a similar reason. They are the *leptons*, the fermionic fields that do not interact with the strong force. Again, L_L contains the left-handed version of the *electron-like* lepton e_R, and the left-handed version of the *neutrino*. The right-handed version of the neutrino is not included, since it has no interaction with any other field. All fields have a different charge, determining their electromagnetic properties. The charge is related to hypercharge by $Q = T_3 + Y$, where T_3 is the third component of the weak isospin. It is just a number that is 0 for all $SU(2)$ singlets and $\frac{1}{2}(\frac{-1}{2})$ for the first(second) component of a doublet.

The gauge bosons are in the adjoint representations of their corresponding groups that have dimensions equal to the number of generators. Thus, since the group $SU(3)$ has 8 generators, its gauge bosons are in the representation **8**. These are called the *gluons*, and they mediate $SU(3)$ interactions. The field W mediates the $SU(2)$ part of the gauge interactions. They only couple to fields that are $SU(2)$ doublets, or equivalently, are left-handed. The field B mediates the $U(1)$ part, coupling to particles with nonzero hypercharge.

Finally, the field ϕ is the Higgs doublet field. Its purpose is explained in the next section.

2.3 Electroweak Symmetry Breaking and the Higgs mechanism

From experiment, it is known that some of the fields in Table 2 are massive. These masses must be represented in the Lagrangian. As it turns out, manually inserting these terms while maintaining gauge invariance is impossible. Therefore, something else is required to let these particles acquire mass without breaking gauge invariance.

We introduce a new[3] scalar field ϕ, called the Higgs field, that is a complex $SU(2)$ doublet. Using that field, we can construct gauge invariant mass terms for the Dirac fields. These are the Yukawa terms, that include dimensionless coupling matrices called the Yukawa matrices. They are 3×3 complex matrices in family space. This means that they allow for interaction terms between family multiplets. In the language of Table 2.2, those terms are:

$$\mathcal{L}_{Yukawa} = -\bar{Q}_L \mathbf{Y}_u \phi^c u_R - \bar{Q}_L \mathbf{Y}_d \phi d_R - L_L \mathbf{Y}_e \phi e_R + h.c. \tag{2.10}$$

where $\phi^c \equiv i\sigma_2 \phi^*$ and a sum over family indices is implied. The new field should also have a gauge invariant kinetic term, such that the equations of motion are not trivial. Additionally, it is possible to include a potential for the scalar field while keeping the Lagrangian consistent:

[3] Well, it was already included in Table 2.

$$\mathcal{L}_\phi = (D_\mu \phi)^\dagger (D^\mu \phi) - V(\phi) \quad \text{where} \quad V(\phi) = \mu^2 \phi^\dagger \phi + \frac{\lambda}{2} \left(\phi^\dagger \phi\right)^2 \tag{2.11}$$

where $D_\mu = \partial_\mu - \frac{1}{2} i g' B_\mu - i g_2 T^a W_\mu^a$ is the *covariant* derivative to assure gauge invariance, and g' and g_2 are coupling constants[4] associated with respectively $U(1)$ and $SU(2)$.[5] The parameters μ^2 and λ are real coefficients.

If we now take $\lambda > 0$ and $\mu^2 < 0$, the scalar potential $V(\phi)$ has a minimum at $|\phi| = \sqrt{-\frac{\mu^2}{\lambda}} \equiv \frac{v}{\sqrt{2}}$. Since the Higgs field is a complex $SU(2)$ doublet, an infinite number of ground states exist. These ground states break $SU(2) \times U(1)$ invariance, since they transition into each other by the corresponding transformations. This phenomenon is known as *spontaneous symmetry breaking*. However, the states are still physically equivalent. The Higgs field is expanded around such a minimum[6] $< \phi >$:

$$\phi = < \phi > + \eta \tag{2.12}$$

Now, $SU(2)$ and $U(1)$ transformations are used to fix the Higgs field minimum:

$$< \phi > = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix} \tag{2.13}$$

Eq. (2.13) can now be inserted into (2.10) and (2.11). The terms proportional to v can then be interpreted as mass terms. For instance, since the lower component of the field L is the electron field, the Yukawa term becomes:

$$\mathcal{L}_{Yukawa,e} = -\frac{v}{\sqrt{2}} \bar{e}_L \mathbf{Y}_e e_R - \bar{e}_L \mathbf{Y}_e \eta e_R + h.c. \tag{2.14}$$

The first term in eq. (2.14) can now be recognized as a mass term. The second term represents the interaction between electrons and the Higgs field. Similarly, plugging eq. (2.13) into (2.11) results in mass terms for W^1, W^2, W^3 and B. This procedure also creates off-diagonal mass terms that mix W^3 and B. These terms can be placed in a matrix, which is then diagonalized to find the masses of the gauge bosons of the weak force W^+, W^- and Z. The Z-boson field is a linear combination of the W^3 and B fields. The other (orthogonal) combination remains massless and is associated with the electromagnetic photon field A.

[4] Later on, g' is rescaled to $g_1 = \sqrt{\frac{5}{3}} g'$ as used in unification models.

[5] $SU(3)$'s coupling constant is g_3, but it doesn't appear here since ϕ is a singlet under $SU(3)$ transformations.

[6] Usually called the vacuum expectation value (VEV)

2.4 Flavor Mixing and Counting Parameters

We have now seen all parameters of the SM. Table 2.3 summarizes them.

Name	Function	Parameters
g_1, g_2 ,g_3	Gauge coupling	$3 \times 1 = 3$
$\mathbf{Y}_e$, $\mathbf{Y}_d$, $\mathbf{Y}_u$	Yukawa coupling	$3 \times 18 = 54$
μ, λ	Higgs parameters	$2 \times 1 = 2$

Table 2.3: SM Parameters

It turns out that not all of the Yukawa parameters are physical. As we have seen, the Yukawa matrices will function as mass matrices. Even though they can in principle be general 3×3 complex matrices, after symmetry breaking they must be transformable into a real, diagonal form. Diagonalization of complex matrices can be done using bi-unitary transformations:

$$\mathbf{Y}'_X = \mathbf{U}_L^{(X)\dagger} \mathbf{Y}_X \mathbf{U}_R^{(X)} \tag{2.15}$$

for $X = u$, d and e. Unitarity of the transformation implies $\mathbf{U}_{R/L}^{(X)\dagger} \mathbf{U}_{R/L}^{(X)} = \mathbf{I}$. These transformations can be absorbed into the fermionic fields:

$$X'_{R/L} = \mathbf{U}_{R/L}^{(X)} X_{R/L} \tag{2.16}$$

This transformation ensures the primed fermionic fields are mass eigenstates. However, we need to be consistent throughout the entire SM Lagrangian. Thus, the transformation of eq. (2.16) has to be applied everywhere. There is only one place where, as a result of electroweak symmetry breaking, the unitary matrices do not cancel out. This occurs in the kinetic terms of the fermion fields, that now contain the gauge interactions of those fields. It is:

$$\mathcal{L} = \frac{g_2}{\sqrt{2}} W_\mu^+ \bar{u}_L \gamma^\mu d_L = \frac{g_2}{\sqrt{2}} W_\mu^+ \bar{u}'_L \mathbf{U}_L^{(u)} \gamma^\mu \mathbf{U}_L^{(d)\dagger} d'_L = \frac{g_2}{\sqrt{2}} W_\mu^+ \bar{u}'_L \mathbf{V} \gamma^\mu d'_L \tag{2.17}$$

The new matrix $\mathbf{V} = \mathbf{U}_L^{(u)} \mathbf{U}_L^{(d)\dagger}$ is the *CKM-Matrix*. Its existence indicates that the quark mass eigenstates are not necessarily interaction eigenstates. $\mathbf{V}$ is a complex 3×3 matrix. However, it is unitary, cutting its number of free parameters in half. It turns out that it is possible to factorize five complex phases from $\mathbf{V}$ that can also be absorbed into the quark fields. The remaining four parameters can then be used to parametrize $\mathbf{V}$. In this new basis, we managed to drastically reduce the number of parameters of the theory. Table 2.4 summarizes them again.

Name	Function	Parameters
g_1, g_2 ,g_3	Gauge coupling	$3 \times 1 = 3$
$\mathbf{Y}_e$, $\mathbf{Y}_d$, $\mathbf{Y}_u$	Yukawa coupling	$3 \times 3 = 9$
$\mathbf{V}$	CKM matrix	$1 \times 4 = 4$
μ, λ	Higgs parameters	$2 \times 1 = 2$

Table 2.4: SM Parameters after diagonalization

Later on, we will discuss the renormalization of these parameters. The renormalization group equations are usually derived from the undiagonalized theory. From these, the equations of the diagonalized Yukawa couplings and the CKM parameters can be derived [31, 32]. When searching for invariants of the renormalization group equations, it will turn out that the parameters of the CKM matrix pose problems. Therefore, it will be easier to use the undiagonalized formulation of the theory.

3 Supersymmetry and the Minimal Supersymmetric Standard Model

It has long been known that the SM may not be the final theory of particle physics, since it lacks a number of elements such as gravity and dark matter. Supersymmetry provides a theoretical extension to the SM. There are multiple advantages to supersymmetry which we will briefly discuss in this chapter. We will, again briefly, discuss the most important extension of the SM, the Minimal Supersymmetric SM (MSSM) until we have all information we need later on. This chapter is based on [25, 26].

3.1 Introduction

Supersymmetry is a symmetry that relates bosonic and fermionic fields. It has a generator Q:

$$Q\,|boson\rangle = |fermion\rangle \qquad Q\,|fermion\rangle = |boson\rangle \tag{3.1}$$

Since Q transforms bosons into fermions, it must be fermionic. It can be shown to have the following algebra:

$$\left\{Q_a, Q_b^\dagger\right\} = 2\,(\sigma^\mu)_{ab}\,p^\mu \qquad \{Q_a, Q_b\} = \left\{Q_a^\dagger, Q_b^\dagger\right\} = 0 \tag{3.2}$$

With $\sigma^\mu = (1, \vec{\sigma})$ and a and b are spinor indices. Q organizes bosons and fermions into *supermultiplets,* containing the same number of bosonic and fermionic degrees of freedom. Using (3.2), it can be shown that the fields in these supermultiplets must have the same mass. Q can also be shown to commute with the generators of the gauge couplings, indicating that the supermultiplets must also have the same gauge quantum numbers.

The SM does not contain a set of particles that can be organized into supermultiplets. Therefore, if supersymmetry is to be implemented, new particles need to be added. The MSSM introduces a new superpartner particle for all SM particles. These particles presumably have a higher mass than the regular SM particles, since otherwise they would have been found. This means that supersymmetry must be a broken symmetry, and therefore a breaking mechanism must be introduced.

The bosonic superpartners of the SM fermions are called *squarks* and *sleptons.* The fermionic superpartners of the gauge bosons are called *gauginos.* There will be some issues in the Higgs sector of the MSSM leading to an additional Higgs doublet field. The superpartners of Higgs bosons are called *Higgsinos.*

3.2 Motivation for Supersymmetry

There are several reasons to seriously consider supersymmetry as an extension to the SM. We discuss the most important advantages.

3.2.1 The Hierarchy Problem

The hierarchy problem can be formulated in several ways. One way is to argue that scalar fields are not naturally massless in a theory without supersymmetry. Since quantum field theory performs its calculations perturbatively, a parameter such as the mass of a particle receives corrections from higher order terms. As we will see in chapter 4, these corrections often diverge. These divergences are usually cut off at some large scale where we expect new physics to occur. In comparison to these kinds of corrections with typical choices of scale, the Higgs particle is essentially massless. All other particles of the SM can be protected against quantum corrections to their mass.

The gauge bosons are protected by gauge invariance. If they are to be massive, a third degree of freedom needs to be introduced. During electroweak symmetry breaking, this degree of freedom is produced by the Higgs field, but before this mechanism there is no possibility for any spin 1 field to acquire mass. Similarly, spin $\frac{1}{2}$ fields are protected by chiral symmetries. Before electroweak symmetry breaking, all fermions are massless, which allows for chiral $U(1)$ transformations of left- and right-handed components of the fields independently. This symmetry is broken as soon as a mass term is added, and thus these kinds of terms cannot be generated in higher orders of perturbation theory.

There is no such protection mechanism for spin 0 fields, such as the Higgs field. This is the essence of the hierarchy problem. Supersymmetry provides a very simple solution: the Higgs field would have a superpartner, the higgsino, which is a spin $\frac{1}{2}$ field. The mass of the higgsino is protected by chiral symmetry, and we have seen that the masses of superpartners must be equal. Hence, the Higgs mass is also protected.

3.2.2 Dark Matter

Astronomical observations such as galaxy rotation curves and gravitational lensing have shown that there must be much more matter in our universe than we can see. This *dark matter* is often assumed to be some particle that only interacts with SM particles through the weak force and gravity. These particles are called *Weakly Interacting Massive Particles* (WIMP), and are not included in the SM. Models incorporating supersymmetry such as the MSSM provide such a candidate, the *Lightest Supersymmetric Particle* (LSP). This is a supersymmetric particle that can not decay further due to the invocation of an extra symmetry called *R-parity*, that will be introduced later.

3.2.3 Gauge Coupling Unification

As we will discover in the next chapter, the gauge couplings, as well as other parameters, are not constants. Their values change as a function of the energy scale. If there is indeed a *Grand Unified Theory* (GUT), then the values of the coupling constants would have to unify at this scale. In the SM, a unification of

the coupling constants never occurs. If supersymmetry is incorporated, it does occur, at a scale of $\mathcal{O}(10^{16}\ GeV)$.

3.2.4 Gravity

If we are to ever achieve a true theory of everything, then that theory must include gravity. Currently, there is no consistent way to unify the SM with a quantized version of gravity. Supersymmetry can also open this door. In fact, the most promising theories that incorporate quantum gravity are superstring theories, which are inherently supersymmetric. To reach a gravitational theory from a quantum field theory perspective, the supersymmetry transformation must be made local. Equation (3.2) indicates that the supersymmetry generators are linked with p_μ. Promoting the supersymmetry transformation to a local symmetry is therefore connected with local spacetime translations, which are generated by p_μ and play an important rôle in general relativity. These kinds of theories are known as *supergravity* theories, including a supermultiplet that mediates gravity and contains the graviton and gravitino.

3.3 A Supersymmetric Lagrangian

Supersymmetry organizes fields in supermultiplets with equal bosonic and fermionic content. There are two types of supermultiplets. The first one we will introduce is the *chiral* multiplet, consisting of a left-handded Weyl fermion field ψ and a complex scalar field ϕ. Writing down the kinetic terms of the Lagrangian, we find:

$$\mathcal{L}_{chiral,kin} = \partial_\mu \phi^* \partial^\mu \phi + i\psi^\dagger \overline{\sigma}^\mu \partial_\mu \psi \tag{3.3}$$

It can be shown that the supersymmetry algebra for this model closes only on-shell. This is caused by the incompatibility of the fields ϕ and ψ in the off-shell regime. The scalar field ϕ always has two degrees of freedom, whether on-shell or off-shell. The Weyl spinor field ψ has two spin polarizations when it is on-shell, corresponding with its two degrees of freedom, but it is a general two-component complex object in the unconstrained off-shell region. To fix this mismatch, an auxiliary field F of mass dimension 2 is introduced. It is a complex scalar field that does not have a kinetic term in the Lagrangian. The only addition to the Lagrangian is:

$$\mathcal{L}_F = F^* F \tag{3.4}$$

This leads to the equations of motion $F = F^* = 0$. However, the field can be made to transform in such a way that the supersymmetry algebra closes automatically without the requirement of being on-shell. It also ensures the degrees of freedom match everywhere. Since a kinetic term is missing, the number of degrees of freedom is 0 on-shell. Off-shell, F is just a general complex

scalar that has two degrees of freedom. Together with ϕ, it leads to four bosonic degrees of freedom in total as required.

It is of course possible to include multiple supermultiplets in the theory. We label these by an index on the fields. Next, interaction terms are added. These terms allow for mixing between fields of a different index. To keep the theory renormalizable, we will see that the terms must have a field content with total mass dimension ≤ 4. Additionally, all terms must be invariant under supersymmetry transformations, since the kinetic terms already were invariant. The only remaining possibility is:

$$\mathcal{L}_{chiral,int} = -\frac{1}{2}W^{ij}\psi_i \cdot \psi_j + W^i F_i + h.c. \tag{3.5}$$

where W^{ij} and W^i are functional derivatives of the *superpotential* W:

$$W^i \equiv \frac{\delta W}{\delta \phi_i} \qquad W^{ij} \equiv \frac{\delta^2 W}{\delta \phi_i \delta \phi_j} \tag{3.6}$$

The superpotential is of course also restricted in its form:

$$W = \frac{1}{2}M^{ij}\phi_i\phi_j + \frac{1}{6}y^{ijk}\phi_i\phi_j\phi_k \tag{3.7}$$

In particular, the superpotential must be holomorphic to ensure supersymmetry invariance. We recognize a symmetric mass matrix M^{ij} and a Yukawa coupling y^{ijk}. By adding eq. (3.5) to the Lagrangian, the auxiliary fields acquire a nontrivial equation of motion that can be used to eliminate them from the Lagrangian altogether. The result is:

$$\mathcal{L}_{chiral} = \partial_\mu \phi_i^* \partial^\mu \phi_i + i\psi_i^\dagger \overline{\sigma}^\mu \partial_\mu \psi_i - \frac{1}{2}\left(W^{ij}\psi_i \cdot \psi_j + W^*_{ij}\psi^{i\dagger} \cdot \psi^{j\dagger}\right) - W^i W^*_i \tag{3.8}$$

The second type of supermultiplet we can add is called a *gauge multiplet*. It consists of a massless gauge boson field A^a_μ and a Weyl gaugino fermion λ^a, where the index a runs over the adjoint representation of the relevant gauge group. Counting degrees of freedom again, we find that A^a_μ and λ^a both have two degrees of freedom on-shell. Off-shell, λ^a has four as usual, but A^a_μ only has three, since one can be removed by gauge invariance. Another auxiliary field is required, this time with one degree of freedom off-shell. We introduce D^a, this time as a real bosonic field. Again, it has mass dimension 2 and no kinetic term. The Lagrangian is:

$$\mathcal{L}_{gauge} = -\frac{1}{4}F^a_{\mu\nu}F^{\mu\nu a} + i\lambda^{\dagger a}\overline{\sigma}^\mu \nabla_\mu \lambda^a + \frac{1}{2}D^a D^a \tag{3.9}$$

where $F^a_{\mu\nu}$ is the regular field strength tensor as defined in eq. (2.9). The gaugino field has its own covariant derivative:

$$\nabla_\mu \lambda^a = \partial_\mu \lambda^a + g f^{abc} A^b_\mu \lambda^c \tag{3.10}$$

The next step is to combine the chiral and gauge parts of the Lagrangian. For this to work, we also introduce a covariant derivative for the chiral supermultiplet as usual. The gauge transformations commute with supersymmetry, so the components of the chiral multiplets must be in the same representation of the gauge group. Let the generators of the gauge group transformation for this representation be T^a. The covariant derivatives for ϕ_i and ψ_i are:

$$D_\mu \phi_i = \partial_\mu \phi_i - i g A^a_\mu \left(T^a \phi\right)_i \qquad D_\mu \psi_i = \partial_\mu \psi_i - i g A^a_\mu \left(T^a \psi\right)_i \tag{3.11}$$

Additionally, it is possible for the gaugino and auxiliary field D^a to interact with the chiral fields. By restricting ourselves to renormalizable terms that are real and invariant under supersymmetry, only three possible terms remain with fixed constants. The full Lagrangian is:

$$\mathcal{L} = \mathcal{L}_{chiral} + \mathcal{L}_{gauge} - \sqrt{2} g \left(\phi_i^* T^a \psi_i\right) \cdot \lambda^a - \sqrt{2} g \lambda^{\dagger a} \cdot \left(\psi_i^\dagger T^a \phi_i\right) + g \left(\phi_i^* T^a \phi_i\right) D^a \tag{3.12}$$

The first two extra terms are the supersymmetrized version of the gauge interaction. The last term provides a nontrivial equation of motion for D^a, which can again be used to eliminate it from the Lagrangian. It is of course possible to introduce multiple gauge groups with their associated gauge bosons and gauginos. We label them with a u. Finally, the total Lagrangian is:

$$\begin{aligned}\mathcal{L} =& D_\mu \phi_i^* D^\mu \phi_i + i \psi_i^\dagger \overline{\sigma}^\mu D_\mu \psi_i - \frac{1}{4} F^a_{u\mu\nu} F_u^{\mu\nu a} + i \lambda_u^{\dagger a} \overline{\sigma}^\mu \nabla_\mu \lambda_u^a \\ & - \frac{1}{2}\left(W^{ij} \psi_i \cdot \psi_j + W^*_{ij} \psi^{i\dagger} \cdot \psi^{j\dagger}\right) - W^i W_i^* - \frac{1}{2} g_u^2 \left(\phi_i^* T_u^a \phi_i\right)^2 \\ & - \sqrt{2} g_u \left(\phi_i^* T_u^a \psi_i\right) \cdot \lambda_u^a - \sqrt{2} g_u \lambda_u^{\dagger a} \cdot \left(\psi_i^\dagger T_u^a \phi_i\right)\end{aligned} \tag{3.13}$$

This is the Lagrangian for a general supersymmetric theory. It is defined completely by its gauge group, its chiral multiplet content and its superpotential.

However, this is not the end of the story. We already know that supersymmetry must be a broken symmetry. If it were not, we would already have seen the supersymmetric partners of the SM particles. Similar to the electroweak symmetry breaking procedure of the SM, supersymmetry can be spontaneously broken. There are several different ways of doing this. As nobody is quite sure which method is correct, the way to proceed is to artificially add all possible supersymmetry breaking terms to the Lagrangian that are allowed. This procedure is called *soft supersymmetry breaking*. 'Soft' refers to the fact that only terms that have couplings of positive mass dimension are allowed. These terms

are automatically suppressed at high energies, where supersymmetry should be an exact symmetry. The allowed terms that remain are:

$$\mathcal{L}_{soft} = -\left(\frac{1}{2}M_a\lambda_a\cdot\lambda_a + \frac{1}{6}a^{ijk}\phi_i\phi_j\phi_k + \frac{1}{2}b^{ij}\phi_i\phi_j\right) + h.c. - \left(m^2\right)^i_j\,\phi^{j*}\phi_i \tag{3.14}$$

It can be shown that these terms do not introduce any divergences in quantum corrections to scalar masses, which is important because one of the most important reasons to introduce supersymmetry in the first place was to get rid of those divergences as encountered in the hierarchy problem. Note that the second and third term of $\mathcal{L}_{soft}$ have the same structure as terms in the superpotential W. However, because functional derivatives are applied to the superpotential, the resulting terms in the Lagrangian are very different.

By choosing the correct gauge group, field content and superpotential, we arrive at the MSSM.

3.4 The Minimal Supersymmetric Standard Model

We have to specify the three ingredients for a minimal supersymmetric model. First, the gauge group should of course be the same as for the SM: $SU(3) \times SU(2) \times U(1)$. The particle content of the MSSM is shown in Table 3.1.

Field	Bosonic	Fermionic	$SU(3) \times SU(2) \times U(1)$
$\mathcal{Q}$	$\tilde{Q}_L = \begin{pmatrix} \tilde{u}_L & \tilde{d}_L \end{pmatrix}$	$Q_L = \begin{pmatrix} u_L & d_L \end{pmatrix}$	$(\mathbf{3},\mathbf{2},\frac{1}{6})$
$\bar{\mathcal{U}}$	$\tilde{u}^*_R$	u^c_R	$(\bar{\mathbf{3}},\mathbf{1},\frac{-2}{3})$
$\bar{\mathcal{D}}$	$\tilde{d}^*_R$	d^c_R	$(\bar{\mathbf{3}},\mathbf{1},\frac{1}{3})$
$\mathcal{L}$	$\tilde{L_L} = \begin{pmatrix} \tilde{\nu_L} & \tilde{e}_L \end{pmatrix}$	$L = \begin{pmatrix} \nu_L & e_L \end{pmatrix}$	$(\mathbf{1},\mathbf{2},\frac{-1}{2})$
$\bar{\mathcal{E}}$	$\tilde{e}^*_R$	e^c_R	$(\mathbf{1},\mathbf{1},1)$
gluon, gluino	$\tilde{g}$	g	$(\mathbf{8},\mathbf{1},0)$
W boson, wino	$\tilde{W}$	W	$(\mathbf{1},\mathbf{3},0)$
B boson, bino	$\tilde{B}$	B	$(\mathbf{1},\mathbf{1},0)$
H_u	$\begin{pmatrix} H^+_u & H^0_u \end{pmatrix}$	$\begin{pmatrix} \tilde{H}^+_u & \tilde{H}^0_u \end{pmatrix}$	$(\mathbf{1},\mathbf{2},\frac{1}{2})$
H_d	$\begin{pmatrix} H^0_d & H^-_d \end{pmatrix}$	$\begin{pmatrix} \tilde{H}^0_d & \tilde{H}^-_d \end{pmatrix}$	$(\mathbf{1},\mathbf{2},-\frac{1}{2})$

Table 3.1: MSSM Field Content

Here, we switched to the left-handed notation[7], as is usually done in a super-

[7] This is briefly explained in Appendix A.

symmetric context. The most important difference with the SM is that we are forced to introduce a second Higgs doublet. To see why this is the case, consider the superpotential of the MSSM:

$$W_{MSSM} = \bar{\mathcal{U}}\mathbf{Y}_u Q H_u - \bar{\mathcal{D}}\mathbf{Y}_d Q H_d - \bar{\mathcal{E}}\mathbf{Y_e} L H_d + \mu H_u H_d \tag{3.15}$$

where we drop all relevant indices. Note that the superpotential contains supermultiplets instead of their scalar components. Since only the functional derivatives of the superpotential end up in the Lagrangian, this describes the exact same physics.

In order for the Yukawa couplings to give mass in the correct way to the up-type quarks using only a single Higgs, a term similar to the first term of eq. (2.10) would be required. This term involves ϕ^c, and thus the term in the superpotential would be $\bar{\mathcal{U}}\mathbf{Y}_u Q H_d^*$. This term is not allowed because the superpotential must be holomorphic[8]. A second Higgs doublet is therefore required. For a similar reason, the term $\mu H_u H_d$ is the only available option as the supersymmetric version of the Higgs mass, since $H_u^* H_u$ and $H_d^* H_d$ are not allowed.

However, there are other terms that are allowed in the superpotential, but that are not included. These terms are:

$$\mathcal{L}\mathcal{Q}\bar{\mathcal{D}} \qquad \mathcal{L}\mathcal{L}\bar{\mathcal{E}} \qquad \mathcal{L}H_u \qquad \bar{\mathcal{U}}\bar{\mathcal{D}}\bar{\mathcal{D}} \tag{3.16}$$

These terms are all gauge-invariant and holomorphic. However, there is no analogy for them in the SM since they violate either *Baryon number* (B) or *Lepton number* (L). Equivalently to the SM, the field $\mathcal{Q}$ has $B = \frac{1}{3}$ and the fields $\bar{\mathcal{U}}$ and $\bar{\mathcal{D}}$ have $B = -\frac{1}{3}$. For the leptonic fields, $\mathcal{L}$ has $L = 1$ and $\bar{\mathcal{E}}$ has $L = -1$. It is then easy to see that the terms in eq. (3.16) do not have $B = L = 0$. There would be significant consequences if the MSSM were to violate either baryon number or lepton number. A typical example is the experimental constraint on proton decay, which violates B and L by one unit. If the terms of eq. (3.16) existed and were unsuppressed, the proton would have a very short lifetime. This heavily contradicts experimental results that have established a lower bound of $\mathcal{O}(10^{32})$ years.

Conservation of baryon and lepton number are not imposed on the SM. Rather, they are accidental symmetries. Therefore, we should aim to assume the least possible number of symmetries to forbid the terms of eq. (3.16). It turns out that only a single symmetry needs to be imposed: the conservation of a multiplicative quantum number called R-parity:

$$P_R \equiv (-1)^{3(B-L)+2s} \tag{3.17}$$

Here s is the spin of the particle. The value of P_R is 1 for all SM particles and -1 for all of their supersymmetric partners. R-parity is the source for the dark matter candidates promised in section 3.2. It ensures that a supersymmetric

[8] This is equivalent with complex analytic.

particle must always decay into at least another supersymmetric particle. Since there is always one of these supersymmetric particles that is the lightest, this particle has no way to decay any further. If this LSP is electrically neutral and has no colour charge, it only interacts weakly with ordinary matter, making it a prime candidate for dark matter.

Finally, the allowed soft supersymmetry breaking terms are:

$$\begin{aligned}\mathcal{L}_{MSSM,soft} = &-\frac{1}{2}\left(M_1\tilde{B}\tilde{B} + M_2\tilde{W}\tilde{W} + M_3\tilde{g}\tilde{g} + h.c.\right)\\ &-\left(\tilde{u}_R^\dagger \mathbf{a}_u \tilde{Q}_L H_u - \tilde{d}_R^\dagger \mathbf{a}_d \tilde{Q}_L H_d - \tilde{e}_R^\dagger \mathbf{a}_e \tilde{L}_L H_d + h.c.\right)\\ &-\tilde{Q}_L^\dagger \mathbf{m}_Q^2 \tilde{Q}_L - \tilde{L}_L^\dagger \mathbf{m}_L^2 \tilde{L}_L - \tilde{u}_R^\dagger \mathbf{m}_u^2 \tilde{u}_R - \tilde{d}_R^\dagger \mathbf{m}_d^2 \tilde{d}_R - \tilde{e}_R^\dagger \mathbf{m}_e^2 \tilde{e}_R\\ &- m_{H_u}^2 H_u^* H_u - m_{H_d}^2 H_d^* H_d - (bH_u H_d + h.c.)\end{aligned} \tag{3.18}$$

Included are complex mass terms for the gauginos M_1, M_2 and M_3. The couplings $\mathbf{a}_u$, $\mathbf{a}_d$ and $\mathbf{a}_e$ are 3×3 complex matrices in family space. Contrary to the Yukawa couplings, they have mass dimension 1. These are usually referred to as the *trilinear couplings*. The matrices $\mathbf{m}_Q^2$, $\mathbf{m}_L^2$, $\mathbf{m}_u^2$, $\mathbf{m}_d^2$ and $\mathbf{m}_e^2$ are hermitian matrices in family space to ensure that the Lagrangian is real. They are mass terms for the sfermions. Also included are masses for the Higgs supermultiplets $m_{H_u}^2$and $m_{H_d}^2$ that have to be real. Finally, b is a complex mixing parameter of mass dimension two that contributes to the Higgs potential.

We have now seen all parameters of the MSSM. Table 3.2 summarizes them.

Name	Function	Parameters
g_1, g_2 ,g_3	Gauge coupling	$3\times1=3$
$\mathbf{Y}_e$, $\mathbf{Y}_d$, $\mathbf{Y}_u$	Yukawa coupling	$3\times18=54$
M_1, M_2, M_3	Gaugino masses	$3\times2=6$
$\mathbf{a}_u$, $\mathbf{a}_d$, $\mathbf{a}_e$	Trilinear couplings	$3\times18=54$
$\mathbf{m}_Q^2$, $\mathbf{m}_L^2$, $\mathbf{m}_u^2$, $\mathbf{m}_d^2$, $\mathbf{m}_e^2$	Sfermion masses	$5\times9=45$
$m_{H_u}^2$, $m_{H_d}^2$	Higgs masses	$2\times1=2$
μ, b	Higgs potential	$2\times2=4$

Table 3.2: MSSM Parameters

This results in a grand total of 168 parameters. Some of these can be rotated away in a fashion similar to the procedure in the SM, but a large number will still remain. Fortunately, experimental results can help to reduce this massive parameter space. A large number of the soft supersymmetry breaking parameters imply either flavor mixing or CP violation, which are both severely restricted by experimental results. This leads us to formulate the *phenomenological* MSSM (pMSSM), where the following simplifications are made:

- The matrices $\mathbf{m}_Q^2$, $\mathbf{m}_L^2$, $\mathbf{m}_u^2$, $\mathbf{m}_d^2$ and $\mathbf{m}_e^2$ are assumed to be diagonal and real. In addition, the first and second generation masses are degenerate.
- First and second generation Yukawa couplings are neglected.
- The trilinear couplings are proportional to the Yukawa couplings: $\mathbf{a}_x = A_x \mathbf{Y}_x$ where $x = u$, d, e.
- $Im(M_a) = 0$, $Im(\mathbf{Y}_x) = 0$, $Im(A_x) = 0$.

The pMSSM introduces no new sources of CP violation apart from the contributions that are produced by the CKM matrix of the SM. Additionally, the complex phases of the parameters μ and b can be absorbed into H_u and H_d. We note again that for later purposes, we will use the complex Yukawa couplings rather than the CKM matrix to find renormalization group invariants. We count the parameters again:

Name	Function	Parameters
g_1, g_2 ,g_3	Gauge coupling	$3 \times 1 = 3$
$\mathbf{Y}_e$, $\mathbf{Y}_d$, $\mathbf{Y}_u$	Yukawa coupling	$3 \times 1 = 3$
M_1, M_2, M_3	Gaugino masses	$3 \times 1 = 3$
A_u, A_d, A_e	Trilinear couplings	$3 \times 1 = 3$
$\mathbf{m}_Q^2$, $\mathbf{m}_L^2$, $\mathbf{m}_u^2$, $\mathbf{m}_d^2$, $\mathbf{m}_e^2$	Sfermion masses	$5 \times 2 = 10$
$m_{H_u}^2$, $m_{H_d}^2$	Higgs masses	$2 \times 1 = 2$
μ, b	Higgs potential	$2 \times 1 = 2$

Table 3.3: pMSSM Parameters

Now, only 26 parameters are left. The implementation of our methods of finding renormalization group invariants will allow for easy adjustment to the above simplifications. Therefore, it will be easy to switch between the MSSM and the pMSSM, or any desired intermediate forms.

4 Renormalization and Renormalization Group Invariants

In this chapter we discuss renormalization, β-functions and their invariants in detail. The goal of this chapter is to discuss everything with as much generality as possible. Since our invariant-finding techniques can be applied to any set of β-functions, it is beneficial to not unnecessarily commit to any theory. Of course, the MSSM is currently the most interesting application of the methods that will be discussed in the upcoming chapters. The contents of this chapter are based on [23, 26, 27, 28].

4.1 Introduction

As briefly mentioned in chapter 2, calculations in quantum field theories are done perturbatively. Perturbative (or quantum) corrections correspond with increasing numbers of loops in the Feynman diagrams used to calculate observable quantities such as cross sections or decay widths. The higher-order terms of these perturbative calculations often turn out to diverge. If divergences turn up during the calculation of observable quantities, the theory loses all predictive power. The central realization of renormalization is that these divergences are not a consequence of the theory itself, but rather of the way we parametrized the theory. Typically, a Lagrangian is written down that contains parameters undetermined by the theory itself. Since the only way of finding the value of these parameters is by performing measurements, it turns out to be much more sensible to express the theory in terms of these measured quantities, rather than in terms of the 'bare' parameters we introduced while writing down the theory. Renormalization shifts the divergences from the observable quantities to the relation between physical parameters and bare parameters.

4.2 Renormalization

To describe the process of renormalization, we consider an unspecified theory that only has a single bare parameter g_b. We calculate and measure some observable quantity $F(x)$, such as a cross section or decay width. The argument x of the quantity F can in principle be any physical variable. In quantum field theories it is typically the invariant mass p^2. Since we are using perturbation theory, let us write:

$$F(x) = g_b + g_b^2 F_1(x) + g_b^3 F_2(x) + ... \tag{4.1}$$

By redefinition of F, it is always possible to achieve the form of eq. (4.1). The problem we are facing is that the functions $F_i(x)$ diverge. An example would be:

$$F_1(x) = \alpha \int_0^\infty \frac{dt}{t+x} \tag{4.2}$$

Which is very similar to the Feynman integrals that are typically encountered in quantum field theories.

Since our theory has only a single parameter g_b, it should suffice to perform only a single measurement in order to fix this parameter. The theory is then completely predictive. We take this measured value to be $F(\mu)$. The problem now is that because of the divergences that occur in the theory, the relation between $F(\mu)$ and g_b will not be well defined[9]. Since we at least know that $F(\mu)$ is finite, the next step is to parametrize the theory in terms of it. With this in mind, we define:

$$F(\mu) = g_r \tag{4.3}$$

The new parameter g_r is the renormalized coupling constant, and the newly chosen parameter of the theory.

The first step is to *regularize* the perturbative expansion of eq. (4.1). The most straightforward way to do this is to introduce a cut-off scale Λ:

$$F_{1\Lambda}(x) = \alpha \int_0^\Lambda \frac{dt}{t+x} \tag{4.4}$$

In physical context, this makes sense because we cannot reasonably expect our current quantum field theories to remain valid to infinite energies, or equivalently, to infinitesimally short lengths. In fact, we know a theory like the SM cannot be correct up to infinitesimal length since the effects of gravity should start playing a rôle.

On the other hand, there are other ways to regularize the perturbative expansion of eq. (4.1). In the SM, a procedure called *dimensional regularization* (DREG) is often used. The divergent integrals are regularized by continuously shifting from 4 spacetime dimensions to $4 - 2\epsilon$. A similar method called *regularization by dimensional reduction* (DRED) is used for supersymmetric models where it is ensured that the spacetime index μ of vector fields A^a_μ still runs over 4 dimensions to maintain a match with the number of degrees of freedom of the associated gaugino.

Regardless of what regularization method we use, the regularized perturbative expansion becomes:

$$F_\Lambda(x) = g_b + g_b^2 F_{1\Lambda}(x) + g_b^3 F_{2\Lambda}(x) + ... \tag{4.5}$$

Such that:

[9] As terms in eq. (4.1) are infinite.

$$\lim_{\Lambda\to\infty} F_\Lambda(x) = F(x) \tag{4.6}$$

The goal is now to write the expansion of $F_\Lambda(x)$ in terms of the coupling g_r, of which we know the value, rather than in terms of g_b. This process is best done recursively. We simply start at order g_b:

$$F_\Lambda(x) = g_b + \mathcal{O}(g_b^2) \tag{4.7}$$

At this order, $F_\Lambda(x)$ is just a constant. We use eq. (4.3) to reparametrize:

$$F_\Lambda(\mu) = g_b + \mathcal{O}(g_b^2) = g_r \Longrightarrow g_b = g_r + \mathcal{O}(g_r^2) \tag{4.8}$$

Where we inverted the relation in the last step. Next, we continue to the next order, g_b^2:

$$F_\Lambda(x) = g_b + g_b^2 F_{1\Lambda}(x) + \mathcal{O}(g_b^3) \tag{4.9}$$

To get rid of g_b we expand it in terms of g_r, which we are free to do since g_b is an arbitrary constant anyway:

$$g_b = g_r + g_r^2 H_1 + g_r^3 H_2 + ... \tag{4.10}$$

Plugging eq. (4.10) into eq. (4.9) we find:

$$F_\Lambda(x) = g_r + g_r^2 H_1 + g_r^2 F_{1\Lambda}(x) + \mathcal{O}(g_r^3) \tag{4.11}$$

We again impose eq. (4.3):

$$g_r = g_r + H_1 g_r^2 + g_r^2 F_{1\Lambda}(\mu) + \mathcal{O}(g_r^3) \Longrightarrow H_1 = -F_{1\Lambda}(\mu) \tag{4.12}$$

Thus, eq (4.10) to g_r^2 order is:

$$g_b = g_r - F_{1\Lambda}(\mu) g_r^2 + \mathcal{O}(g_r^3) \tag{4.13}$$

So now the regularized, but previously divergent quantity $F_{1\Lambda}(\mu)$ turns up in the parametrization of g_b. This is exactly what we wanted, if it has now indeed disappeared from the expansion of $F_\Lambda(x)$. To check that, we just plug (4.12) into (4.11):

$$F_\Lambda(x) = g_r + g_r^2 \left(F_{1\Lambda}(x) - F_{1\Lambda}(\mu)\right) + \mathcal{O}(g_r^3) \tag{4.14}$$

Does this get rid of the divergent part of $F_{1\Lambda}$? At least in our example of (4.4) it does:

$$F_{1\Lambda}(x) - F_{1\Lambda}(\mu) = \alpha \int_0^{\Lambda} \frac{dt}{t+x} - \alpha \int_0^{\Lambda} \frac{dt}{t+\mu} = \alpha\,(\mu - x) \int_0^{\Lambda} \frac{dt}{(t+x)\,(t+\mu)} \tag{4.15}$$

which is clearly not infinite when $\Lambda \to \infty$. In general, requiring $F_{1\Lambda}(x) - F_{1\Lambda}(\mu)$ not to diverge means requiring that the *divergent part of* $F_{1\Lambda}(x)$ *does not depend on* x. This is the central point of the question of renormalizibility. In fact, it has to be true for *all* physically observable quantities that can be predicted by the theory. If we move up to the third order and follow the same procedure, we find:

$$\begin{aligned} F_{\Lambda}(x) = {} & g_r + g_r^2\,(F_{1\Lambda}(x) - F_{1\Lambda}(\mu)) \\ & + g_r^3\,(F_{2\Lambda}(x) - F_{2\Lambda}(\mu) - 2F_{1\Lambda}(\mu)\,(F_{1\Lambda}(x) - F_{1\Lambda}(\mu))) + \mathcal{O}(g_r^4) \end{aligned} \tag{4.16}$$

This imposes the identical constraint of finiteness on $F_{2\Lambda}(x) - F_{2\Lambda}(\mu)$. Let us explicitly check this new requirement on the divergent part of $F_{1\Lambda}(x)$:

$$\alpha \int_0^{\Lambda} \frac{dt}{t+x} = \alpha \log(\Lambda + x) - \alpha \log(x) = \alpha \log(\Lambda) + \alpha \log\left(\frac{1 + \frac{x}{\Lambda}}{x}\right) \tag{4.17}$$

It can in fact be shown that the functions $F_{i\Lambda}(x)$ cannot be more than logarithmically divergent to ensure renormalizability. Thus, by some simple power counting for integrals that can appear in a theory, one can predict the renormalizability of that theory.

Let us assume for a moment the coupling g_b has mass dimension d. By dimensional analysis, we expect that $\frac{g_r}{g_b} = \left(\frac{\mu}{\Lambda}\right)^d$. Thus, if $d < 0$, this leads to couplings that become large for lower scales μ. Since we are doing perturbation theory, we require the parameters to be small at low scales such that eq. (4.5) remains approximately valid at low perturbation order. Therefore, the requirement of renormalizability can be formulated as the *absence of couplings with negative mass dimension.* This can of course be determined more precisely by considering the possible divergent integrals that appear in the perturbative calculations.

The procedure of renormalization can be generalized to a multitude of parameters. The general recipe is:

1. Select some observable $F_k(x)$ for every $g_{b,j}$.
2. Write $F_k(x)$ as a perturbative expansion with terms $F_{ik}(x)$.
3. Regularize the terms using some regularization procedure to $F_{ik,\Lambda}(x)$.
4. Redefine g_{bj} by writing it as a power series of g_{rk} absorbing the divergences.
5. Use this definition to substitute g_{bj} by g_{rk} everywhere else.

4.3 The Renormalization Group

We have previously selected a specific value for x we called μ. The value of our observable F at $x = \mu$ is what we used as the renormalized coupling constant g_r. Obviously, the result of the renormalization procedure should not depend on what value we select for μ. Selecting a new μ' with an associated g_r' should lead to the same observable F. This freedom corresponds with an equivalence class between the pairs (μ, g_r) and (μ', g_r') at all loop orders. The consequence is the following group law: going from $(\mu, g_r) \to (\mu', g_r')$ should give the same result as $(\mu, g_r) \to (\mu'', g_r'') \to (\mu', g_r')$. The corresponding group is known as the *renormalization group*.

It is important to verify that g_r' is finite as long as g_r is. We check this by writing the perturbation of $F_\Lambda(x)$ up to second order:

$$\begin{aligned} F_\Lambda(x) &= g_r + g_r^2 \left(F_{1\Lambda}(x) - F_{1\Lambda}(\mu)\right) + \mathcal{O}(g_r^3) \\ &= g_r' + g_r'^2 \left(F_{1\Lambda}(x) - F_{1\Lambda}(\mu')\right) + \mathcal{O}(g_r'^3) \end{aligned} \tag{4.18}$$

It can easily be checked that equality is ensured up to second order for:

$$g_r' = g_r + g_r^2 \left(F_{1\Lambda}(\mu') - F_{1\Lambda}(\mu)\right) + \mathcal{O}(g_r^3) \tag{4.19}$$

and thus a transition to a different scale is finite as long as the renormalizability condition is fulfilled[10]. In fact, making the transitions $(\mu, g_r) \to (\mu'', g_r'') \to (\mu', g_r')$ does indeed lead to the same result as $(\mu, g_r) \to (\mu', g_r')$ by applying this procedure twice. The validity of eq. (4.19) of course depends on the size of the jump of $\mu \to \mu'$, simply because it is done perturbatively. Instead, the aim should be to take very small steps between different μ. To find the change in g_r for such small changes in μ, we define the *β-function*:

$$\beta_{g_r} \equiv \mu \frac{\partial g_r}{\partial \mu} = \frac{\partial g_r}{\partial \log(\mu)} \tag{4.20}$$

The effects of a large change in μ can then be found by integrating these β-functions. In general, these β-functions can be computed to any loop order by setting:

$$\mu \frac{d}{d\mu} F_\Lambda(x) = \left(\mu \frac{\partial}{\partial \mu} + \mu \frac{\partial g_{r,i}}{\partial \mu} \frac{\partial}{\partial g_{r,i}}\right) F_\Lambda(x) = \left(\mu \frac{\partial}{\partial \mu} + \beta_{g_{r,i}} \frac{\partial}{\partial g_{r,i}}\right) F_\Lambda(x) = 0 \tag{4.21}$$

where summation over i for all parameters of the theory is implied. Eq. (4.21) must of course be true since we have just seen that $F_\Lambda(x)$ should be independent of the value of μ we choose. Recall that we can select any physical observable for F.

[10] This time it is indicated by μ' instead of x.

In practice, the β-function is usually redefined as:

$$\beta_{g_r} \equiv \frac{\partial g_r}{\partial t} \quad \text{with} \quad t \equiv \log_{10}\left(\frac{\mu}{\mu_0}\right) \tag{4.22}$$

where μ_0 is some arbitrary scale to make the logarithm dimensionless. A factor of $16\pi^2$ is often added. A property of field theories is that the n-th order part of a β-function receives a factor $\left(16\pi^2\right)^{-n}$, so multiplying by $16\pi^2$ cancels one of these powers everywhere . We refrain from adding it here because the factors $\left(16\pi^2\right)^{-n}$ will be relevant later on. We also note that, without specifically showing it, the β-functions of a quantum field theory are always polynomial with coefficients in $\mathbb{Q}$, apart from the factors $\left(16\pi^2\right)^{-n}$.

4.4 Probing High Scales

The β-functions we defined in the previous chapter can be used to learn more about the behavior of a theory at unreachable scales. The typical situation is that the scales reachable by experiment are not even close to the scales where interesting physics is expected to occur. By using the β-functions of the physical parameters of a theory, we can find the values of these parameters at these experimentally unreachable scales. We are often looking for scales at which parameters unify or otherwise attain special values.

We discuss the unification of parameters of the following toy model:

$$\begin{aligned}
\beta_v &= \frac{1}{16\pi^2} v^3 \\
\beta_w &= \frac{1}{16\pi^2} v^2 \left(5w + 6x - 4y\right) \\
\beta_x &= \frac{1}{16\pi^2} v^2 \left(-w - 2x + 4z\right) \\
\beta_y &= \frac{1}{16\pi^2} v^2 \left(x + y - 5z\right) \\
\beta_z &= \frac{1}{16\pi^2} v^2 \left(w - 2y + 6z\right)
\end{aligned} \tag{4.23}$$

This model is constructed such that it is simple, yet illustrates the upcoming points well. It is also meant to be very similar to the first order β-functions of the pMSSM. The parameter v can be seen as one of the gauge couplings, while w, x, y, and z are similar to the sfermion masses. The factor $\frac{1}{16\pi^2}$ is a result of the first order calculation of the β-functions, so it is included here as well.

As we have seen in the previous chapter, the soft supersymmetry breaking terms are included to parametrize our ignorance of the specific supersymmetry breaking mechanism. A choice of supersymmetry breaking mechanism will usually lead to some kind of unification of sfermion masses, so we use our toy model to

search for these kinds of unifications. These unifications usually occur at the GUT scale, which is $\mathcal{O}(10^{16}\,GeV)$, or $t \approx 16$ if we choose $\mu_0 = 1\,GeV$. We therefore demand our 'masses' to unify at this scale, and evolve them down to the collider scale at around $t = 2$. Fig. 4.1 shows the running of the parameters. The β-functions were numerically solved using the odeint package [29].

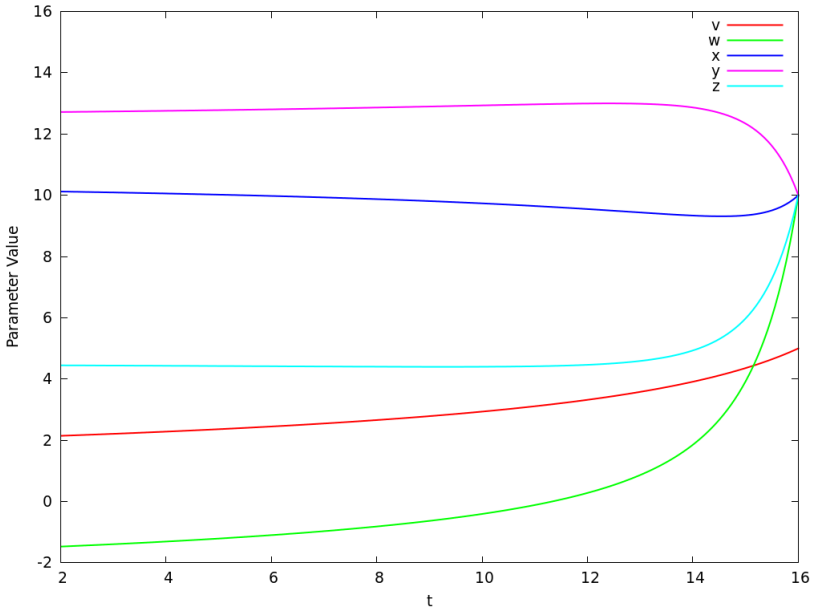

Figure 4.1: The running of the parameters of the toy model of eq. (4.23).

Using this example, we discuss the possible methods of finding the unification as if we do not know where it is yet. The methods are all more extensively discussed in [28].

4.4.1 The bottom-up method

Perhaps the most logical first idea is to measure the values of the physical parameters at the scale of current experiment, and use the β-functions to evolve them to the scale of interest. In our example, we would have to measure the values of all parameters at $t = 2$ (or higher), and evolve them until we find the unification.

There are some very clear advantages to this method. Most importantly, we do not need to know the scale of unification *or* the unifying values beforehand. Instead, we should just see them appear as the parameters evolve. On the other hand, in order for the calculation of this evolution to be possible at all, usually all parameters have to be measured. The renormalization group equations are usually heavily coupled differential equations, so even if one is interested in the

unification of a small set of parameters only, the rest is still required to evolve them. If we are considering a theory that includes new physics, the requirement of having to know all parameters can be very restrictive.

Perhaps more importantly is that, due to the polynomial nature of the β-functions, the evolution of parameters is often very sensitive to errors. Fig. 4.2 shows the same running as Fig. 4.1, but with the value of y at $t = 2$ changed by just 1%.

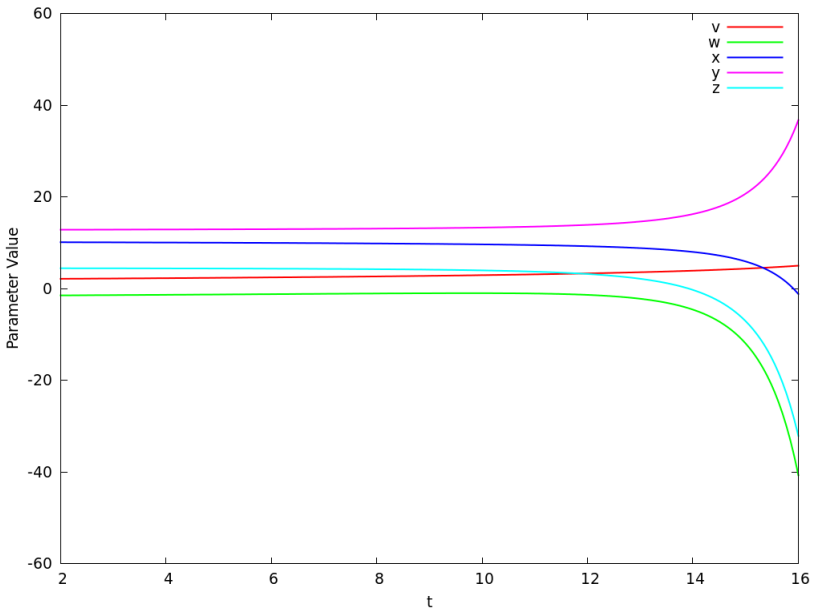

Figure 4.2: The running of the parameters of the toy model of eq. (4.23) with a 1% change in y.

It is clear that unification does not happen anymore at all. If there are uncertainties in multiple parameters, this effect is obviously enhanced even further.

4.4.2 The top-down method

The top-down method works the other way around. Unification at some scale is assumed, then the values of the parameters are evolved to the experimental scale to check for matching with experimental results. In case of our example, we would assume $w(16) = x(16) = y(16) = z(16) = 10$ and evolve the values of these parameters to $t = 2$.

The top-down method requires knowledge of the scale of unification and the value of the parameters at that unification. Since the theory does not predict these, the parameter space has to be scanned at some scale of unification. Values

for the parameters are selected, evolved down and checked for consistency with experimental data. The advantage is that the downward evolution does not suffer as much from numerical errors. Fig. 4.3 shows the evolution of the parameters of our toy model when we assume the unification scale is at $t = 16.5$ and the values unify at a value again differing by 1%.

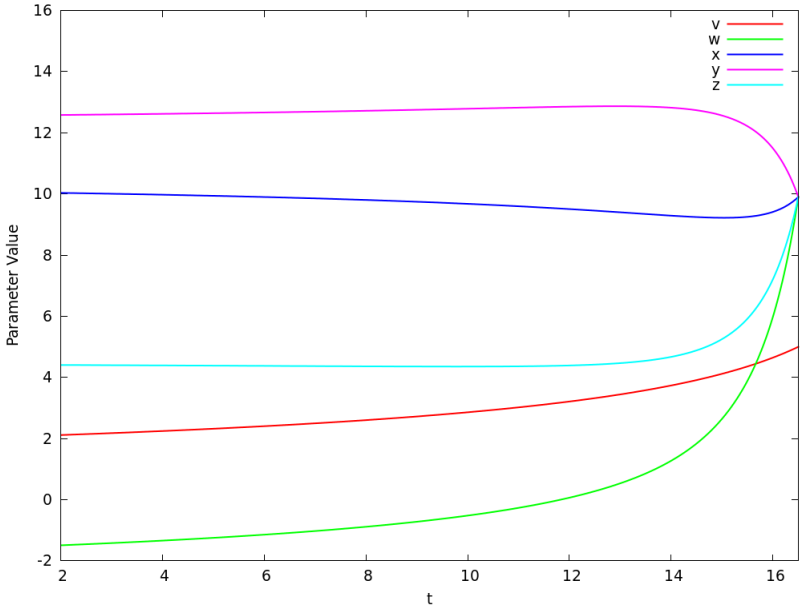

Figure 4.3: The running of the parameters of the toy model of eq. (4.23) unifying at $t = 16.5$ at a 1% lower value.

Unfortunately, the scanning procedure can be extremely time-consuming for models with large numbers of parameters, especially when several models of unification have to be tested.

4.4.3 Renormalization group invariants

The third method is that of *renormalization group invariants*, which are the central object of this thesis. Renormalization group invariants are combinations of parameters that are invariant under the flow of the renormalization group equations. That is, I is an invariant if:

$$\frac{d}{dt}I = 0 \tag{4.24}$$

We show the use of these invariants using our example system (4.23). This system has two simple invariants:

$$I_1 = w + 3x - 2z \qquad I_2 = x + 2y + z \tag{4.25}$$

These invariants are very simple since they follow directly from the linear dependence between the terms of the β-functions. In general, the β-functions are complex and lengthy polynomial functions, and determining their invariants is not simple at all. In fact, the next chapter will discuss a computer algebraic method to find invariants of arbitrary sets of β-functions.

Let us assume that w, x, y and z unify at some scale with unknown value s. We find:

$$I_1 = 2s \qquad I_2 = 4s \Longrightarrow 2I_1 - I_2 = 0 \tag{4.26}$$

The second part of eq. (4.26) is known as a *sum-rule*. We can now make use of the fact that I_1 and I_2 are invariant under renormalization group flow, which just means that:

$$\forall t:\ 2I_1 - I_2 = 2w(t) + 5x(t) - 2y(t) - 5z(t) = 0 \tag{4.27}$$

In particular, we can check for unification using just the values of the parameters we measured at $t = 2$. If eq. (4.26) does not vanish, we know unification cannot occur within the reach of the theory without having to do any extra work. We list some advantages of this method.

- Not all parameters of the theory need to be measured. In the above example, we never used the parameter v, and we do not need it to confirm the fact that the parameters w, x, y and z unify at some scale. This property can be particularly relevant when new physic is found and its parameters are only selectively measurable at that time.
- No information on the unification scale is required. We never had to use the fact that the parameters w, x, y and z unify only at $t = 16$. Due to the invariance of I_1 and I_2, the scale itself is completely irrelevant.
- This method is completely algebraic, while the other two methods are usually performed numerically. Algebraic methods are preferable to numerical ones for numerous reasons, of which the development of numerical errors described in the bottom-up method is a very important one. The top-down method also suffers from its numerical nature as it requires the scanning of a parameter space.

In the context of the MSSM, these renormalization group invariants can be used to determine the supersymmetry breaking mechanism. The goal of [28] is to determine the sum rules for these breaking mechanisms. However, the construction of sum-rules is in principle not limited to the MSSM, and can be applied anywhere. We will thus focus on finding these invariants for arbitrary sets of β-functions.

5 Methodology

We are now ready to develop a computer algebraic methodology that searches for invariants of arbitrary systems of renormalization group equations. Throughout this chapter, the described methods and techniques are designed to be applicable to any system of β-functions. However, in some cases there will be a focus on applications to the MSSM β-functions since it is currently the main application of interest. We will first detail a method that efficiently finds a specific set of invariants. Next, we attempt to expand this method to a more general case, and notice the need to adjust in order to maintain simplicity. Finally, we extend this method to reach an even larger class of invariants.

5.1 Monomial Searching (MS)

We first consider a very simple class of invariants. Let x_i for $i = 0...n$ be the renormalized parameters of the theory. The class of *monomial invariants* can then be represented as:

$$M = \prod_{i=1}^{n} x_i^{a_i} \tag{5.1}$$

where $\vec{a} \epsilon \mathbb{Z}^n$. Note that the mathematical definition from the field of commutative algebra defines a monomial to have $\vec{a} \epsilon \mathbb{N}^n$. We do not apply this restriction, but will still call the class of invariants of the form of M a monomial invariant.

In the case of a monomial invariant, our only job is to find the set of powers $\vec{a}$ such that M is an invariant for some set of β-functions β_i. To that end, let us consider the derivative of M with respect to the dimensionless variable t:

$$\frac{dM}{dt} = \sum_{i=1}^{n} a_i x_1^{a_1} ... x_i^{a_i - 1} ... x_n^{a_n} \beta_i = \sum_{i=0}^{n} \frac{a_i \beta_i}{x_i} \prod_{j=1}^{n} x_j^{a_j} = M \sum_{i=1}^{n} \frac{a_i \beta_i}{x_i} \tag{5.2}$$

If we disregard the obvious[11] case of $M = 0$, we can write this condition for invariance to be:

$$\sum_{i=1}^{n} \frac{a_i \beta_i}{x_i} = 0 \tag{5.3}$$

It was mentioned previously that the β-functions are always polynomial. Note that the factor $\frac{1}{x_i}$ has the potential to introduce negative powers in the expression. Again, while technically polynomials should have terms with powers in $\mathbb{N}$, we will call eq. (5.3) a polynomial equation.

[11] Although technically true: $M = 0$ is an invariant.

The crucial point now is that eq. (5.3) must be zero for all t. This just means that every monomial structure in eq. (5.3) must be canceled by the combination of powers $\vec{a}$ and the coefficients that originate from the β-functions.

As an example, let us consider a simple toy system of two β-functions, β_x and β_y:

$$\beta_x = xy + 3xy^2 \qquad \beta_y = 2y^2 + 6y^3 \tag{5.4}$$

On first sight, there is already some structure evident in this simple example. For instance, one might notice that the coefficients of β_y are twice those of β_x. These kinds of structures are not evident anymore at all as the number of renormalized variables increases and the β-functions become more complex. Let us now apply eq. (5.3) to this system.

$$\frac{a_x\beta_x}{x}+\frac{a_y\beta_y}{y} = \frac{a_x}{x}\left(xy+3xy^2\right)+\frac{a_y}{y}\left(2y^2+6y^3\right) = \left(a_x+2a_y\right)y+\left(3a_x+6a_y\right)y^2 = 0 \tag{5.5}$$

We can now see that eq. (5.3) can be reduced to a linear requirement by utilizing the fact that any invariant must be an invariant for all t. Our example reduces to a linear system of equations:

$$a_x + 2a_y = 0 \qquad 3a_x + 6a_y = 0 \tag{5.6}$$

Or in matrix notation:

$$\begin{pmatrix} 1 & +2 \\ 3 & +6 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} = 0 \tag{5.7}$$

We finally see that the computation of monomial invariants can be reduced to a problem of finding the nullspace of a system of equations. From elementary linear algebra [1], we know that any system of equations will only have a non-trivial nullspace if the matrix is not of full rank. In this case it is obvious the matrix is of rank 1, e.g. the rows are linearly dependent. A basis vector for the nullspace of this matrix might be:

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \tag{5.8}$$

Thus we learn that an invariant M for the system of β-functions described in eq. (5.4) might be represented as:

$$M = \frac{x^2}{y} \tag{5.9}$$

However, the nullspace of a matrix is always a subspace of the vectorspace of $\vec{a}$. Thus, we have not found a single invariant, but a continuum of invariants. In retrospect, this might be expected. Indeed, if M is an invariant, then of course M^p must also be an invariant since $\frac{d(M^p)}{dt} = pM^{p-1}\frac{dM}{dt} = 0$. In fact, the nullspace might have dimensionality higher than one. In that case, there will be multiple independent monomial invariants M_i. The linearity of the powers then just tells us that for all p_i, $\prod M_i^{p_i}$ will also be an invariant, and its powers are represented in the nullspace of the matrix we described earlier.

It should be mentioned that there are good reasons for letting $\vec{a}\epsilon\mathbb{Z}^n$. Since all β-functions have coefficients in $\mathbb{Q}$, it is always possible to find a number $F\epsilon\mathbb{Z}$ such that $\forall i\epsilon[1...n]$ one can redefine $\beta_i \to F\beta_i$ such that all coefficients in β_i are in $\mathbb{Z}$. This system of β-functions has the same invariants as the original system, since any derivative is just multiplied with F. Hence, the matrix as shown above will only include numbers in $\mathbb{Z}$. A consequence of this fact is that any basis vector of the nullspace can be written in terms of numbers in $\mathbb{Z}$.

The method described above provides a simple and elegant method of finding monomial invariants. It only requires the solution of a single system of linear equations, which is a task computers are very well suited for nowadays. The size of the system is a point of interest. It is easy to see that the number of columns of the corresponding matrix is equal to the number of renormalized variables in a theory. In practice, this number ranges from $\mathcal{O}(10)$ to $\mathcal{O}(100)$. For instance, the renormalizable parameters of the MSSM without any constraints add up to 168. However, it is by no means a requirement that the system is square. Depending on the complexity and forms of the β-functions, the number of rows might exceed the number of columns by several factors. This just means that a large portion of these rows must be linearly dependent in order for a monomial to exist. In other words, more complex β-functions usually lead to fewer monomial invariants.

5.2 Extending to Polynomial Invariants

Next, we attempt to generalize this method to polynomial invariants. To this end, we consider invariants of the following form:

$$P = \sum_{j=1}^{m} C_j \prod_{i=1}^{n} x_i^{a_{ij}} \tag{5.10}$$

$\mathbf{a}$ is now a matrix in $\mathbb{Z}^{n\times m}$ for the same reasons as discussed in the previous section. The number m is the number of allowed monomial terms in P. In principle, m could be any number in $\mathbb{N}$. We also introduce $\vec{C}\epsilon\mathbb{Z}^m$ as a set of coefficients for the separate monomial terms in this expression. There was no analogy for these coefficients in the monomial case because multiplication of an invariant with any number remains invariant. We again consider the derivative of this class of invariants:

$$\frac{dP}{dt} = \sum_{j=1}^{m} C_j \frac{d}{dt}\left(\prod_{i=1}^{n} x_i^{a_{ij}}\right) = \sum_{j=1}^{m} C_j \prod_{i=1}^{n} x_i^{a_{ij}} \left(\sum_{k=1}^{n} \frac{a_{kj}}{x_k}\beta_k\right) \tag{5.11}$$

It is now immediately obvious that the resulting equation is highly nonlinear. Not only are we forced to introduce a new set of parameters $\vec{C}$ that do not appear as powers and complicate the equation further, but we can also no longer divide out the product $\prod_{i=1}^{n} x_i^{a_{ij}}$ since it is different for different j. It follows that there is no direct analogy to the method explained in the previous section. We will have to look further.

At first sight, one might try to extend eq. (5.11) to a system of nonlinear equations in order to attempt to find a solution. One way this might be done is by exploiting the fact that eq. (5.11) has to hold for all t. This means that for any t_0, when all x_i are replaced by $x_i(t_0)$, eq. (5.11) has to vanish identically. Since only the running of all renormalized parameters is determined by theory, not their value, we expect the invariants to be independent of the actual values of the parameters. This means that any point in the space of parameters might be selected and inserted into eq. (5.11) to produce a valid equation. The only restriction is that setting a parameter to zero effectively removes the parameter from the theory, in which case eq. (5.11) should still hold, but information is lost.

Solving for a set of unknown variables will of course require at least as many equations as variables. In this case, the set of variables consists of $\vec{C}$ and $\mathbf{a}$, which amounts to $m(n+1)$ variables. Realistically, a valid method has to be able to reach large values of m without consequence. For instance, the pMSSM has known invariants at one-loop order that contain $\mathcal{O}(10)$ terms, corresponding with $\mathcal{O}(100)$ unknown variables in eq. (5.11).

Solving nonlinear systems of equations is a very difficult task. In fact, there really only is a single method for solving nonlinear equations in the most general sense [2]. This method is known as the Newton-Rhapson method. Several methods that fall in the same class as Newton-Rhapson exist for special cases[12], and there are extensions to improve the convergence properties. We will describe the method along with such an extension before discussing its application to the task of finding invariants.

Consider the following system of general equations:

$$\forall i \epsilon [1,n] : \; F_i(x_1, ..., x_n) = 0 \tag{5.12}$$

Since there are as many equations as variables, this system is 'square' in the algebraic sense. In vector notation, it is simply:

[12] For instance, Broyden's method.

$$\vec{F}(\vec{x}) = 0 \tag{5.13}$$

We write down the Taylor series of eq. (5.12):

$$F_i(\vec{x} + \delta\vec{x}) = F_i(\vec{x}) + \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j} \delta x_j + O\left(\delta\vec{x}^2\right) \tag{5.14}$$

Defining the *Jacobian matrix* $\mathbf{J}$ as:

$$J_{ij} = \frac{\partial F_i}{\partial x_j} \tag{5.15}$$

this can again be rewritten as:

$$\vec{F}(\vec{x} + \delta\vec{x}) = \vec{F}(x) + \mathbf{J} \cdot \delta\vec{x} + O\left(\delta\vec{x}^2\right) \tag{5.16}$$

The system of equations can now be linearized by neglecting terms of order $\delta\vec{x}^2$ and higher. We set eq. (5.16) to zero to obtain:

$$\mathbf{J} \cdot \delta\vec{x} = -\vec{F}(\vec{x}) \tag{5.17}$$

When evaluated at some $\vec{x}$, eq. (5.17) is a linear system that can be solved through any method of linear system solving.[13] The result is a correction to the solution vector, and it can thus be used as an iterative method to reach a root of the system. Starting from some $\vec{x}_0$ where $\vec{F}(\vec{x}) \neq 0$, at step p the solution vector is updated as:

$$\vec{x}_p = \vec{x}_{p-1} + \delta\vec{x} \tag{5.18}$$

Since Newton-Rhapson only takes first order effects into account, the starting point $\vec{x}_0$ has huge influence on the success of the algorithm. Especially for high n, $\vec{x}_0$ has to be very close to a root of the system in order for the method to converge. There is no absolute measure possible for this requirement, since it is highly dependent on the system $\vec{F}(\vec{x})$.

Although there are no other methods for root-finding of general nonlinear systems, there are several methods of finding minimal of such systems. One might formulate the root-finding problem as described above as a minimization problem by simply squaring all equations. Unfortunately, this process typically introduces a large number of local extrema diminishing the effectiveness of the method. However, the idea of minimization can be used to improve the convergence properties of Newton-Rhapson.

[13] Popular options are LU and QR decomposition, but regular Gaussian elimination works too.

Consider the following scalar function, where we drop the implied argument $\vec{x}$:

$$g = \frac{1}{2}\vec{F}\cdot\vec{F} \tag{5.19}$$

If $\vec{F} = 0$, then $g = 0$, but otherwise $g \geq 0$. Thus, the roots of $\vec{F}$ are a global minimum of g. Indeed, g is just the sum of the squared system of equations we described above. Now let us consider the derivative of g:

$$\frac{\partial g}{\partial x_i} = 2\left(\frac{1}{2}\sum_{j=1}^{n} F_j \frac{\partial F_j}{\partial x_i}\right) = \sum_{j=1}^{n} F_j J_{ij} \tag{5.20}$$

In vector notation, this is:

$$\nabla g = \vec{F}\cdot\mathbf{J} \tag{5.21}$$

Using eq. (5.17), we can now write down the following:

$$\nabla g \cdot \delta\vec{x} = \left(\vec{F}\cdot\mathbf{J}\right)\left(-\mathbf{J}^{-1}\cdot\vec{F}\right) = -\vec{F}\cdot\vec{F} \tag{5.22}$$

That is, the Newton-Rhapson step $\delta\vec{x}$ decreases g at first order. In other words, if one takes a small enough step in the direction of $\delta\vec{x}$, g is guaranteed to decrease. Since we are looking to find the global minimum of g, the following iterative procedure can be defined:

$$\vec{x}_p = \vec{x}_{p-1} + \lambda\delta\vec{x} \quad \text{such that} \quad g_p < g_{p-1} \tag{5.23}$$

for $0 < \lambda \leq 1$. The value $\lambda = 1$ is always tried first, since the full Newton-Rhapson step converges very quickly if $\vec{x}_{p-1}$ is very close to a root. If the full Newton-Rhapson step does not decrease g, λ is lowered until it does. We will not go into detail on how subsequent values for λ are selected, but we will mention that this *backtracking* method significantly improves the rate of convergence.

In practice, the algorithm selects low values for λ quite often, leading to slow walks through the space of variables. Additionally, the algorithm might still get stuck in local minima of g if the Newton-Rhapson step does not manage to reach another region with lower g. These problems can be tackled by selecting a different $\vec{x}_0$.

Newton-Rhapson with backtracking was applied to the pMSSM β-functions. The variable space includes both $\vec{C}$ and $\mathbf{a}$. To select starting points $\vec{x}_0$, van der Corput sequences were used to form a set of maximally self-avoiding points $\vec{x}_0$ [3]. This strategy has the benefit of allowing for easy selection of the size of the set. Apart from the fact that computation times rise quickly with the increase of the number of allowed terms in the polynomial form of an invariant (previously called m), the main problem with this method is the existence of an infinite

number of roots. Apart from trivial invariants, the main culprit is the coefficient vector $\vec{C}$. In terms of only $\vec{C}$, invariants behave exactly like elements of a vector space. That is, if $\vec{C}_1$ and $\vec{C}_2$ represent some invariants (with associated $\mathbf{a}_1$ and $\mathbf{a}_2$), then $\lambda_1\vec{C}_1+\lambda_2\vec{C}_2$ is also an invariant $\forall\lambda_1,\lambda_2\epsilon\mathbb{Z}$[14]. Newton-Rhapson will lead us to any invariant of this form. As a consequence, for some number of allowed terms m, the method will almost always converge to some linear combination of smaller invariants instead of to a single bigger one, simply because there are more of them. Even worse is the fact that false roots will occasionally pop up. Our method of requiring eq. (5.11) to vanish identically for random values of the parameters is not as strong as eq. (5.11) vanishing for all values. Of course, these false roots could be eliminated by comparing results originating from different sets of random values for the parameters, but this becomes much more tricky if the acquired results can be any linear combination of invariants.

The problem of linear equations of invariants can be remedied somewhat by introducing extra terms in $\vec{F}$ to ensure that known invariants are no longer roots. This might for instance be achieved by adding Gaussian functions that have sharp peaks at unwanted invariants. This procedure turns out to drastically decrease the effectiveness of the backtracking method, since it produces a much more complex functional landscape with more local extrema. All in all, Newton-Rhapson has too many downsides to be a reliable method of finding invariants. The method we describe next has the property of automatically taking care of the problem of linear combinations of invariants. It also no longer requires the use of random numbers to generate equations, and as such every found solution is guaranteed to be an independent invariant.

5.3 Dimensionalities

Before developing the new method for finding polynomial invariants, we first have to discuss a concept we will refer to as *dimensionalities*. Of course, mass dimension is a concept well-known from physics. We will derive a more general notion directly from the β-functions.

Consider another toy system:

$$\beta_x = 2xy + 10y^3 \qquad \beta_y = -x - y^2 \tag{5.24}$$

Let us now assign a dimension, an additive number, to all variables involved. We assign a 1 to y and a 2 to x. This way, the terms in β_x have dimension 3 and the terms in β_y have dimension 2. From now on, we will state that $dim\,(\beta_x) = 3$ and $dim\,(\beta_y) = 2$. Now consider the derivative of a general monomial in x and y:

$$M = x^p y^q \tag{5.25}$$

Which has dimension $2p + q$. Taking the derivative:

[14] In principle, we can of course even choose $\forall\lambda_1,\lambda_2\epsilon\mathbb{C}$

$$\frac{dM}{dt} = px^{p-1}y^q\beta_x + qx^p y^{q-1}\beta_y = 2px^p y^{q+1} + 10px^{p-1}y^{q+3} - qx^{p+1}y^{q-1} - qx^p y^{q+1} \tag{5.26}$$

All terms in the derivative have dimension $2p+q+1$. In general, the dimension of any term T differentiated with respect to the parameter x_i is $dim\,(T) - dim\,(x_i) + dim\,(\beta_x)$. That is, as long as $dim\,(\beta_{x_i}) - dim\,(x_i) = D$ for all i, then all terms in the derivative of a monomial term have the same dimension. In particular, that means that *monomial terms with different dimensionalities cannot produce identical monomial terms by differentiation*, because of the simple fact that monomial terms cannot be equal if they do not have the same dimensionalities. As a result, we are only interested in invariants that have terms of the same dimensionality. Invariants with terms with different dimensionalities exist, but since the derivatives of those terms cannot 'mix', terms of separate dimension must be separately invariant. This conclusion is the first step in solving the problem of linear combinations of invariants described in the previous section. For example, an invariant of eq. (5.24) is:

$$I = x^2 + 2xy^2 + 5y^4 \tag{5.27}$$

Which can be checked easily. As required, all terms have the same dimensionality of 4.

Let us now consider a selection of the one-loop β-functions from the pMSSM that are simple, but still illustrate this concept:

$$\beta_{g_a} = b_a g_a^3 \tag{5.28}$$

$$\beta_{M_a} = 2b_a M_a g_a^2 \tag{5.29}$$

$$\beta_{m^2_{u1,2}} = -\frac{32}{15}g_1^2 M_1^2 - \frac{32}{3}g_3^2 M_3^3 - \frac{4}{5}g_1^2 D_Y \tag{5.30}$$

$$\beta_{y_t} = y_t\left[6|y_t|^2 + |y_b|^2 - \frac{13}{15}g_1^2 - 3g_2^2 - \frac{16}{3}g_3^2\right] \tag{5.31}$$

Where $a = (1,2,3)$, $D_Y = m^2_{H_u} - m^2_{H_d} + Tr\left(\mathbf{m}_Q^2 - \mathbf{m}_L^2 - 2\mathbf{m}_u^2 + \mathbf{m}_d^2 + \mathbf{m}_e^2\right)$ and $b_a = \left(\frac{33}{5}, 1, -3\right)$. Note that for mass dimension, $dim_m\,(\beta_{x_i}) - dim_m\,(x_i) = 0$ for all i, and thus mass dimension classifies under our general definition of dimensionalities. However, assigning a 1 to the coupling constants g_a, y_t, y_b and y_τ, and a 0 to all other parameters, we find another dimensionality which we will call the *coupling dimension*. Note that now, $dim_c\,(\beta_{x_i}) - dim_c\,(x_i) = 2$ for all i. This particular dimensionality has an underlying source in physics. It is

related to the number of vertices that appear in Feynman diagrams at different perturbative orders. As a consequence, the constant we called D previously is not the same at different loop orders. It is equal to 2 for the first order as shown in eq. (5.28)-(5.31), and subsequent orders increase it in steps of 2. This also means that a nonzero dimensionality is assigned to every parameter of the theory, which will be relevant later.

It should be noted that there is no need for the dimensionality to be a consequence of underlying physics. The following method will not care at all if the dimensionality has a physical context, or if it is completely accidental.

5.4 Polynomial Searching (PS)

Let us now consider how we can build new invariants from a given set of known invariants. The possibilities are:

- Sums of invariants.
- Products of invariants.

The content established in the previous section helps us deal with the second point. Compared to the invariants themselves, a product of those invariants will (almost) always have a different dimensionality[15]. Thus, if our method can be limited to a predetermined set of dimensionalities, it will allow us to discern between 'primitive' invariants and products of those primitives.

The first point actually only concerns the linearity of $\vec{C}$. Therefore, if we are able to formulate a method that involves doing linear algebra on just $\vec{C}$, it should be able to find a basis for primitive invariants at a certain dimensionality. Let us now first redefine a monomial term as a function of its powers:

$$M(\vec{a}) = \prod_{i=1}^{n} x_i^{a_i} \tag{5.32}$$

Now let $\vec{d}$ be a vector of *dimensionalities* of size r. That is, all elements of the vector represent one of the dimensionalities that occur in a theory. For instance, the vector $\vec{d} = (1, 2)$ might correspond with mass dimension 1 and coupling dimension 2 in the MSSM. Using this vector and eq. (5.32), we now write down the definition of a set of monomials central to our new method:

$$\mathcal{M}(\vec{d}) = \{\vec{a}\epsilon\mathbb{Z}^n | \forall l\epsilon\,[1, ..., r] : \; dim_l\left(M\left(\vec{a}\right)\right) = d_l\} \tag{5.33}$$

That is, the set of monomial terms with fixed dimensionality $\vec{d}$. Because $\mathbb{Z}$ is infinite, it is in practice required to put a limit on $\vec{a}$. This might be achieved by something like:

[15] The only exception is if an invariant that has all dimensionalities zero exists. Our method will be able to find these invariants.

$$\mathcal{M}_p(\vec{d}) = \left\{ \vec{a}\epsilon\mathbb{Z}^n | \forall l\epsilon\left[1,...,r\right] :\ dim_l\left(M\left(\vec{a}\right)\right) = d_l,\ \sqrt{\vec{a}\cdot\vec{a}} \leq p \right\} \tag{5.34}$$

More complex limitations can be applied to $\vec{a}$. The limit p might be made to depend on $\vec{d}$, which often makes sense since higher elements in $\vec{a}$ are required to reach higher $\vec{d}$. In theories like the pMSSM the two available dimensionalities are completely separate: there are no parameters that have more than one nonzero dimensionality. It is therefore good practice to separate $\vec{a}$ into independent parts for each dimensionality and treat their limitations separately. Since there are many valid methods of restricting the set $\mathcal{M}(\vec{d})$, we will use $\mathcal{M}_p(\vec{d})$ to refer to such a set that is restricted in some way, rather than the specific example of eq. (5.34). Let us consider such a set $\mathcal{M}_p(\vec{d})$ of size s. We now consider a polynomial form that is a function of its coefficients $\vec{C}\epsilon\mathbb{Z}^s$ and of the dimensionality vector $\vec{d}\epsilon\mathbb{Z}^r$:

$$P_{\vec{d}}\left(\vec{C}\right) = \sum_{j=1}^{s} C_j M_j \quad \text{with} \quad M_j\epsilon\mathcal{M}_p(\vec{d}) \tag{5.35}$$

Finally, denoting the powers of the i-th monomial term as $\vec{a}^i$, we consider the derivative of $P_{\vec{d}}\left(\vec{C}\right)$ which of course looks very similar to eq. (5.11):

$$\frac{dP_{\vec{d}}\left(\vec{C}\right)}{dt} = \sum_{j=1}^{s} C_j \frac{d}{dt}\left(\prod_{i=1}^{n} x_i^{a_j^i}\right) = \sum_{j=1}^{s} C_j \left[\prod_{i=1}^{n} x_i^{a_j^i}\left(\sum_{k=1}^{n} \frac{a_j^k}{x_k}\beta_k\right)\right] \tag{5.36}$$

The part of eq. (5.36) in brackets is a well-defined polynomial function. Therefore, we can now employ the same tactic of section 5.1 by demanding eq. (5.36) to vanish identically for all $\vec{x}$. In the same way, this results in a system of linear equations that all equate zero. The problem of finding invariants is thus reduced to the problem of determining a nullspace of a (potentially very large) integer matrix.

As an example, let us reconsider the toy system of eq. (5.24):

$$\beta_x = 2xy + 10y^3 \qquad \beta_y = -x - y^2 \tag{5.37}$$

We assigned dimensionality of 2 to x and 1 to y. There is only one dimensionality, so $r = 1$ and $\vec{d}$ is just a scalar. For simplicity, without specifying the mechanism of restricting $\mathcal{M}_p(\vec{d})$, consider the set $\mathcal{M}_p(4)$[16]:

$$\mathcal{M}_p(4) = \left\{x^2, xy^2, y^4\right\} \tag{5.38}$$

We insert eq. (5.38) into eq. (5.36) and demand it to vanish:

[16]Which just happens to only contain the terms we already saw in the invariant of this system.

$$
\begin{aligned}
0 &= C_1 2x\beta_x + C_2 y^2 \beta_x + C_2 2xy\beta_y + C_3 4y^3 \beta_y \\
&= (10C_2 - 4C_3)\, y^5 + (20C_1 - 4C_3)\, xy^3 + (4C_1 - 2C_2)\, x^2 y
\end{aligned} \tag{5.39}
$$

The corresponding system is:

$$
\begin{pmatrix} 0 & 10 & -4 \\ 20 & 0 & -4 \\ 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = 0 \tag{5.40}
$$

The basis vector $\vec{C} = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$ spans the nullspace of this system, which corresponds with the invariant:

$$
I = x^2 + 2xy^2 + 5y^4
$$

that we found earlier. For simplicity we neglected to include any terms in $\mathcal{M}_p(4)$ that do not appear in the invariant. Usually, large numbers of terms will be included that do not appear in any invariant.

This method has all the advantages we mentioned earlier. It is linear in $\vec{C}$ unlike the previously described procedure that used Newton's method, and thus deals with linear combinations of invariants naturally. It also fixes $\vec{d}$, which guarantees a finite number of invariants as long as $\mathcal{M}(\vec{d})$ is made to be finite.[17] A possible downside is of course the potential to generate massive systems of which the nullspace has to be found. The number of columns is directly related to the choice of the method of restricting the size of $\mathcal{M}(\vec{d})$. We should always strive to use a $\mathcal{M}_p(\vec{d})$ of a largest as possible size, since it reaches the most invariants. The number of rows is not necessarily equal to the number of columns, and in general depends on the complexity of the β-functions. For instance, searching at the two-loop level among MSSM β-functions produces many more rows than searching at one-loop level. We will describe how it is possible to deal with the very large systems in question in the next chapter.

5.5 Factorized Polynomial Searching (FPS)

A potential downside of the method described in the previous section is the required limitation on the allowed powers of the monomial terms. There is no direct relation between the powers that appear in the β-functions and the powers of their invariants. Consider the following simple, but illustrative system:

$$
\beta_x = -x \qquad \beta_y = 60x^2 + 58y \tag{5.41}
$$

[17] Products with invariants that have $\vec{d} = 0$ are restricted by the condition that makes $\mathcal{M}_p(\vec{d})$ finite.

The powers of these β-functions do not exceed 2, yet an invariant of this system is:

$$I = x^{60} + x^{58}y \tag{5.42}$$

This is of course caused by the large coefficients of β_y. Again, in the context of larger systems of more complex β-functions, it is very unclear if there is some maximum allowed powers for invariants. However, rewriting the invariant we just found points us in a direction that might help:

$$I = x^{58}\left(x^2 + y\right) \tag{5.43}$$

The high power of x can be factorized. Considering the derivative clears up the roles of the factorized monomial and the remaining polynomial part:

$$\frac{dI}{dt} = 58x^{57}\left(x^2+y\right)\beta_x + x^{58}\left(2x\beta_x + \beta_y\right) = x^{58}\left(58x\beta_x + \frac{58y\beta_x}{x} + 2x\beta_x + \beta_y\right) \tag{5.44}$$

I is of course only an invariant if the term inside the brackets vanishes. The monomial factorization contributes only the first term inside the brackets, while the other two are produced by the polynomial part. The power of the factorized monomial appears outside the brackets. We therefore see that there are more degrees of freedom available that can make the derivative vanish. By simply changing the power, the factor 58 can be tweaked to let the terms inside the brackets vanish without any other consequence. We thus acquire a strictly larger reach of invariants by factorizing a monomial.

To formalize the method, we define $\vec{b}\epsilon\mathbb{Z}^n$ as the vector of powers of the factorized monomial. The form of an invariant is now:

$$P_{\vec{d}}\left(\vec{C},\vec{b}\right) = \prod_{l=1}^{n} x_l^{b_l} \sum_{j=1}^{s} C_j M_j \quad \text{with} \quad M_j \epsilon \mathcal{M}_p(\vec{d}) \tag{5.45}$$

The derivative is:

$$\begin{aligned}\frac{dP_{\vec{d}}\left(\vec{C},\vec{b}\right)}{dt} &= \prod_{l=1}^{n} x_l^{b_l} \sum_{j=1}^{s} C_j \frac{d}{dt}\left(\prod_{i=1}^{n} x_i^{a_j^i}\right) + \frac{d}{dt}\left(\prod_{l=1}^{n} x_l^{b_l}\right)\sum_{j=1}^{s} C_j \prod_{i=1}^{n} x_i^{a_j^i} \\ &= \prod_{l=1}^{n} x_l^{b_l}\left(\sum_{j=1}^{s} C_j \prod_{i=1}^{n} x_i^{a_j^i} \sum_{k=1}^{n}\left(\frac{a_j^k}{x_k}\beta_k + \frac{b_k\beta_k}{x_k}\right)\right)\end{aligned} \tag{5.46}$$

This of course vanishes only if:

$$\sum_{j=1}^{s} C_j \prod_{i=1}^{n} x_i^{a_j^i} \sum_{k=1}^{n} \left(\frac{a_j^k}{x_k} \beta_k + \frac{b_k \beta_k}{x_k} \right) = 0 \tag{5.47}$$

Eq. (5.47) is similar to the requirement of letting eq. (5.36) vanish. The factorization produced the extra term $\frac{b_k \beta_k}{x_k}$ that represents more degrees of freedom. Unfortunately, we can no longer reduce this equation to a linear system like we did in the previous section. Terms with just C_j appear, but additional terms that have $C_j b_k$ are also present. The equations are polynomial, and in particular they are quadratic. This is not quite as bad as the equations we encountered in section 5.2, but neither is it simple to solve them. We now discuss some attempts that were made to solve eq. (5.47) before presenting a new way of looking at it.

5.5.1 Linearization

It is possible to cast the quadratic system produced by eq. (5.47) into a linear form by the process of *linearization*[18]. Define:

$$h_{jk} = C_j b_k \tag{5.48}$$

Then the system of eq. (5.47) is indeed linear in the variables $\vec{C}$ and $\mathbf{h}$. A downside is that this produces a system that is of size $s\,(n+1)$ while the systems of the previous sections were of size s. Aside from the possible objections from a computational standpoint, in many cases this system will be underdetermined. The linearization process is well-known in the field of cryptography, and more advanced techniques have been developed to produce additional equations, such as the XSL algorithm [4]. Problems arise when one attempts to retrieve the vectors $\vec{b}$ from solutions of $\vec{C}$ and $\mathbf{h}$. It is required that:

$$\forall j : \; b_k = \frac{h_{jk}}{C_j} \tag{5.49}$$

A general basis vector of the nullspace of the linearized system will not fulfill the requirement of eq. (5.49). Instead, one has to search for those linear combinations of solution vectors that do obey eq. (5.49). This effect is only enhanced by the existence of many directions in the nullspace that do not correspond with an invariant whatsoever, but are introduced as a result of the linearization method. Let $\left(\vec{C}_i, \mathbf{h}_i\right)$ for $i = (1, ..., m)$ be a basis of the nullspace of the system of eq. (5.47). We can then write eq. (5.49) as:

$$\forall j : \; b_k = \frac{\sum_{i=1}^{m} \lambda_i \left(h_i\right)_{jk}}{\sum_{i=1}^{m} \lambda_i \left(C_i\right)_j} \tag{5.50}$$

[18] Not to be confused with the linearization that occurs in the Newton-Rhapson algorithm.

for $\vec{\lambda}\epsilon\mathbb{Z}^m$. Eq. (5.50) is just another quadratic system of equations in $\vec{\lambda}$. This problem originates directly from the absence of terms that are linear in the components of $\vec{b}$, and it forces us to find another solution.

5.5.2 Gröbner bases

Quadratic systems can be solved generally by a method that can be seen as a generalized version of Gaussian elimination. The objective is to compute a so-called *Gröbner basis* for a system of polynomials,[19] then employ backward solving to retrieve solutions for the variables [5, 6].

Let $F = \{f_1, ..., f_m\}$ be a set of polynomials in $\mathbb{Q}[x_1, ..., x_n]$, meaning it can contain variables $x_1, ..., x_n$ and coefficients in $\mathbb{Q}$. The *ideal* generated by F is defined as:

$$< F >= \left\{ \sum_{i=1}^{m} h_i f_i | h_1, ..., h_m \epsilon \mathbb{Q}[x_1, ..., x_n] \right\} \tag{5.51}$$

The set F is a basis for the ideal $< F >$, analogous to the basis of a vector space in linear algebra. There are other possible bases for an ideal, and the Gröbner basis is one of them that has some special properties. To define the Gröbner basis, we first need some other mathematical tools.

We first need a method of writing down polynomials consistently. In linear algebra we do this by placing the variables in a vector. Here, since there is in principle an infinite number of terms one can build, we instead introduce *monomial ordering.* If we define a monomial as $M = \prod_{i=1}^{n} x_i^{\alpha_i}$, then a monomial ordering $>$ is called *admissible* if:

1. $\forall \alpha \epsilon \mathbb{N}^n : \ \alpha > 0$
2. $\forall \alpha, \beta, \gamma \epsilon \mathbb{N}^n : \ \alpha > \beta \Longrightarrow \alpha + \gamma > \beta + \gamma$

That is, every monomial is larger than 0, and if some monomial is larger than some other monomial, then this remains true if both are multiplied by some third monomial. There are many possible monomial orderings. One of the most well-known examples is the *lexicographic order*: $\alpha > \beta \Longleftrightarrow$ the left-most nonzero entry of $\alpha - \beta$ is positive. By choosing a monomial ordering, we can define for arbitrary polynomials f and g:

- $LT(f)$: The *leading term* of f as the term ordered as highest.
- $LM(f)$:The *leading monomial* of f as the monomial of $LT(f)$ without its coefficient.
- $LC(f)$: The *leading coefficient* of f as the coefficient of $LT(f)$.

[19] Note that in this case, we mean the mathematical polynomial, e.g. all powers are positive.

- $LCM(f,g)$: The *least common multiple* of f and g as the lowest monomial divisible by $LM(f)$ and $LM(g)$.

Next, we define *polynomial reduction.* A polynomial g can be reduced by a polynomial f if $LT(g)$ can be canceled out by subtracting a suitable multiple of f. To be more precise, let $m = \frac{LM(g)}{LM(f)}$ and $c = \frac{LC(g)}{LC(f)}$. Then g reduces to h modulo f if and only if m exists.[20] In fact, m and c exactly form the suitable multiple of f we need, and thus $h = g - cmf$. We denote this process by $g \rightarrow_f h$. If m does not exist, we call g *irreducible* by f.

If we instead have a set of polynomials $F = \{f_1, ..., f_m\}$, we can define $g \rightarrow_F^+ h$ as the continuous reduction of g by all polynomials in the set F until it is irreducible for all of them. Then, h is the *normal form* of g modulo F.

We also need the so-called *S-polynomial* of two polynomials f and g. It is:

$$S - poly(f,g) = \frac{LCM(f,g)}{LT(f)} f - \frac{LCM(f,g)}{LT(g)} g \tag{5.52}$$

$S - poly(f,g)$ is in the same ideal as f and g. It is built such that the leading terms cancel out exactly.

We are now finally ready to define a Gröbner basis. Different definitions that are all equivalent can be used. We will define the Gröbner basis in terms of the objects we defined previously. Let $F = \{f_1, ..., f_n\}$ be a set of polynomials. The following statements are equivalent:

- F is a Gröbner basis.
- $\forall f_i, f_j \epsilon F : \; S - poly(f_i, f_j) \rightarrow_F^+ 0$
- $\forall f \epsilon < F >: \; f \rightarrow_F^+ 0$

Although the second statement is in fact contained in the third, it will form the basic principle of the *Buchberger* algorithm that finds a Gröbner basis for an arbitrary set of equations:

Algorithm 1 The Buchberger Algorithm

Input: $F = \{f_1, ..., f_n\}$
Output: A Gröbner basis $G = \{g_1, ..., g_m\}$for $< F >$

$G \leftarrow F$
DO
    $G' \leftarrow G$
    FOR all $\{p, q\}$ in G' with $p \neq q$
        $S \leftarrow NormalForm(S - Poly(p, q))$
        IF $S \neq 0$
            $G \leftarrow G \cup \{S\}$
WHILE $G' \neq G$

[20] E.g. there are no negative powers in $\frac{LM(g)}{LM(f)}$.

This algorithm keeps adding reduced S-polynomials to the basis until it is a Gröbner basis. The algorithm can be made much more efficient by some modifications, such as doing some preliminary work on F before starting. For our purposes, the most important aspect of the Buchberger algorithm is that the polynomials that are being added are constructed such that certain variables are eliminated. The order in which these variables are eliminated depends on the monomial ordering chosen. When the Buchberger algorithm terminates, the final polynomial that is added will be a polynomial in the 'lowest' variable only. From there, the solutions to this polynomial can be computed. The next variable can then be solved by substituting the solutions for the lowest variable, and so forth.

By quick inspection, it is obvious that the Buchberger algorithm can be very expensive. The algorithm has to compute the S-polynomial of every pair available in F initially, plus all pairs with new basis elements included. The normal form algorithm is quite expensive too: it continuously has to try to reduce the polynomial by all elements of the current basis. Every time a suitable polynomial is found, the reduction is done and the process has to start over until there are no suitable polynomials any longer. Typically, the Buchberger algorithm also has the tendency to produce very large Gröbner bases. It turns out to be very difficult to reach a solid estimate on the complexity of the algorithm, and this is still an ongoing topic of research [6].

There are a number of improvements listed in [5, 6] which help. The most important are some rules that can determine if the S-polynomial will vanish before actually computing it. Additionally, the initial basis might be reduced with respect to itself before starting the Buchberger algorithm. The monomial ordering often has a big influence on the computation time and the size of the generated basis. Since our method is really only interested in finding the vector of powers $\vec{b}$, we are required to pick some *elimination* order as mentioned in [6] to ensure we can solve for the power vectors first, meaning we do not have full freedom in the selection of our monomial ordering. Code implementing the Buchberger algorithm can be found in [8]. As it turns out, the results differ immensely when searching at different dimensionality vectors $\vec{b}$. In some cases, computation times approach hours and the machine runs out of memory due to the sheer size of the Gröbner basis. Although some success was achieved, the method is clearly unsuitable as a support for software that is meant to find invariants for arbitrary sets of β-functions in a somewhat reasonable amount of time.

5.5.3 Eigenvalue interpretation

Let us consider the implications of eq. (5.47) again. In general, the resulting quadratic system might be written as:

$$\mathbf{A}\vec{C} + \sum_{i=1}^{n} b_i \mathbf{B}_i \vec{C} = \left(\mathbf{A} + \sum_{i=1}^{n} b_i \mathbf{B}_i \right) \vec{C} = 0 \tag{5.53}$$

Where $\mathbf{A}$ and $\mathbf{B}_i$ are generally non-square, integer matrices. Especially in the case where we choose some j such that $\forall i \neq j : \; b_i = 0$, eq. (5.53) is very similar to a *generalized eigenvalue problem in the second sense* [7]. Viewing the problem this way has the advantage of treating $\vec{b}$ and $\vec{C}$ separately. A typical eigenvalue problem represents a search for eigenvalues such that the matrix acquires a nontrivial nullspace. The eigenvalues are usually solved for first. Then, eigenvectors are found that span the nullspace of the system with the eigenvalue inserted. In our specific case, this is exactly what we are looking for. The vector $\vec{C}$ is still the vector of coefficients that must still display its linear properties even when factorizing a monomial. On the other hand, we expect to find a finite set of solutions for $\vec{b}$ corresponding with a finite number of possible monomials that factorize from an invariant.

Unfortunately, most numerical approaches to these kinds of problems are only relevant to the case of square matrices and only allow for the previously described case of a single nonzero b_i. Let us, for now, consider the case of a single nonzero b_i. We rename that b_i to b and $\mathbf{B}_i$ to $\mathbf{B}$ for convenience:

$$(\mathbf{A} + b\mathbf{B})\,\vec{C} = 0 \tag{5.54}$$

Since our systems always have more rows than columns, consider the following statement: *Any eigenvalue of an overdetermined system must be an eigenvalue of every fully determined subsystem of that system.* That is, we can select a number of rows from the matrix $(\mathbf{A} + b\mathbf{B})$ such that we are left with a square system. Then, all eigenvalues of the full system must be included in the eigenvalues of the subsystem. The validity of this statement can easily be seen by considering the implications if it were not true. If there is indeed some fully determined system that does not have an eigenvalue that the full system does have, then the full system can never have a nonzero nullspace when this eigenvalue is inserted. The fully determined subsystem is still present in the full system, preventing the existence of a nullspace.

In practice, this means we can just perform Gaussian elimination on the matrix $(\mathbf{A} + b\mathbf{B})$. This process will result in a square, upper triangular matrix where additional equations have been eliminated completely. The result thus depends on the initial order of the rows in the matrix, which can be manipulated freely. The candidate eigenvalues can be solved from the polynomials that appear on the diagonal, and then checked for validity simply by inserting them and solving the nullspace using the same machinery that would be used without factorizing a monomial.

In principle, this method could be used to solve cases of multiple nonzero b_i too. However, there is a very fundamental difference compared with the case of only a single nonzero b_i. After doing Gaussian elimination, polynomials appear on the diagonal of the resulting upper triangular matrix. Roots of those polynomials correspond with eigenvalues. In order to find a solution to multiple nonzero b_i, the statement about subsystems is still valid. However, it is not required that

every nonzero b_i appears on every diagonal element. In fact, this again depends on the order of the rows of the matrix. Consider the following illustrative example: we search through the β-functions of the pMSSM at $\vec{d} = (0, -3)$, where the -3 is the coupling dimension. We factorize a monomial of two variables: $g_1^{b_1} g_2^{b_2}$. $\mathcal{M}(\vec{d})$ is in principle a fairly large set. By doing preliminary eliminations that will be described in the next chapter, we are left with the following system:

$$\begin{pmatrix} b_1 - 2 & 0 & 0 \\ b_2 - 2 & 0 & 0 \\ 0 & 33b_1 - 99 & 5b_2 - 15 \\ 0 & 1 - b_2 & 0 \\ 0 & 0 & 1 - b_1 \end{pmatrix} \begin{pmatrix} C_{g_1^{-2} g_2^{-2}} \\ C_{g_1^{-3} g_2^{-1}} \\ C_{g_1^{-1} g_2^{-3}} \end{pmatrix} = 0 \tag{5.55}$$

where the coefficients are labeled by their corresponding invariants. We are looking for the invariant $\frac{5}{g_1^2} - \frac{33}{g_2^2} = g_1 g_2 \left(\frac{5}{g_1^3 g_2^1} - \frac{33}{g_1^1 g_2^3} \right)$, which would correspond with $b_1 = 1$, $b_2 = 1$. There are also three obvious invariants that just lead to the powers of g_1 and g_2 canceling out. For instance, consider the subsystems of rows $1, 4, 5$ and rows $1, 3, 4$, applying Gaussian elimination wherever needed:

$$1, 4, 5 \to \begin{pmatrix} b_1 - 2 & 0 & 0 \\ 0 & 1 - b_2 & 0 \\ 0 & 0 & 1 - b_1 \end{pmatrix} \tag{5.56}$$

$$\begin{aligned} 1, 3, 4 &\to \begin{pmatrix} b_1 - 2 & 0 & 0 \\ 0 & 33b_1 - 99 & 5b_2 - 15 \\ 0 & 1 - b_2 & 0 \end{pmatrix} \\ &\to \begin{pmatrix} b_1 - 2 & 0 & 0 \\ 0 & 33b_1 - 99 & 5b_2 - 15 \\ 0 & 0 & (1 - b_2)(5b_2 - 15) \end{pmatrix} \end{aligned} \tag{5.57}$$

Looking at the third column, we find that $b_1 = 1$ and $b_2 = 1$ or $b_2 = 3$. The case of 1 and 1 corresponds with the invariant we are looking for, while the case of 1 and 3 corresponds with one of the trivial invariants[21]. Next, the corresponding eigenvectors can be found. It can easily be checked that:

$$b_1 = 1, b_2 = 3 \to \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \to I = 1 \qquad b_1 = 1, b_2 = 1 \to \begin{pmatrix} 0 \\ 5 \\ -33 \end{pmatrix} \to I = \frac{5}{g_1^2} - \frac{33}{g_2^2} \tag{5.58}$$

In conclusion, this method is perfect for monomials with one parameter. If multiple parameters are factorized, solving for the sets of eigenfunctions requires finding consecutive subsystems that eventually give enough information to determine the viable sets of nonzero b_i. Since the matrices can grow large very

[21] In this case it is $\frac{g_1 g_2^3}{g_1 g_2^3} = 1$.

quickly and there is no natural guideline for which subsystems need to be selected[22], the above method is not very viable for the factorization of more than a single parameter.

We have described three methods of approaching the problem of factorized polynomial searching. The first two methods turned out not to be viable from a computational standpoint, and had the issue of treating $\vec{C}$ and $\vec{b}$ on equal footing. The third method of interpreting the problem as an eigenvalue problem fixes the latter issue, but requires the severe restriction of limiting ourselves to only a single factorized parameter to be computationally viable.

5.6 Invariant Filtering

The methods described in the previous two sections depend on a choice for the dimensionality vector $\vec{d}$. This of course means that we want to apply these techniques with as many choices for $\vec{d}$ as we can handle in order to reach as many invariants as possible. However, if we have found a set of invariants $\{I_1, ..., I_n\}$ with dimensionality vectors $\left\{\vec{d}_1, ..., \vec{d}_n\right\}$, then we can expect to find invariants of the form $\prod I_i^{\alpha_i}$ at dimensionality $\sum \alpha_i \vec{d}_i$. This is complicated further by the fact that invariants are not necessarily neatly separated: if we select a dimensionality vector where we find a single new invariant and a large number of products of old invariants, the output will be some linear combination of all of them, effectively hiding the new invariant. Some sort of algorithm that can filter out products of previous invariants is required.

To this end we introduce a *polynomial ordering*:

$$I_1 > I_2 \Longleftrightarrow HNP(I_1) > HNP(I_2) \tag{5.59}$$

Where the function HNP is *the Highest number of Nonzero Powers among monomial terms*. For instance, $xy^2z + z^3 > x^2y + yz$ since xy^2z has 3 nonzero powers while x^2y and yz both have two.

If the invariants are sorted starting from the lowest HNP, then it is (almost) guaranteed that invariants that contain products of other invariants are listed below those other invariants. To see why, consider $I_1 = \sum_{i=1}^{a} I_{1,i}$ and $I_2 = \sum_{j=1}^{b} I_{2,j}$. The product $I_1 I_2$ contains all terms $I_{1,i} I_{2.j}$. It is of course possible for cancellations to occur in those products such that values of i and j exist for which the number of nonzero powers in $I_{1,i} I_{2.j}$ is lower than $HNP(I_1)$ and/or $HNP(I_2)$, but as long as $I_1 \neq I_2$ it will never occur for all i and j. For general invariants there are two exceptions:

- An invariant is a power of another monomial invariant. These invariants are easy to find manually. They can also easily still be filtered out by adding the monomials found by monomial searching to the top of the list.

[22] This information is 'hidden' until Gaussian elimination is performed.

- An invariant is a linear combination of products of invariants such that it only contains terms such as mentioned above, where powers have canceled out. Even if it is possible to construct these invariants, it would be completely accidental if either of the methods we described previously were to find them.

The rest of the procedure is explained in algorithm 2:

Algorithm 2 Filter algorithm

Input: $I = \{I_1, ..., I_n\}$with dimensionality vectors $D = \left\{\vec{d}_1, ..., \vec{d}_n\right\}$ of size r.

Output: $I' = \left\{I_1^{'}, ..., I_m^{'}\right\}$ with $m \leq n$, a set of unique invariants.

$I' \leftarrow \emptyset$
$i \leftarrow 1$
WHILE $i \leq n$
    $J \leftarrow \emptyset$
    Find all $\vec{\alpha}$ with $\forall k \epsilon [1, r] : dim_k \left(\prod_{j=1}^{i-1} I_j^{\alpha_j}\right) = d_{i,k}$ & NOT $\prod_{j=1}^{i-1} I_j^{\alpha_j} > I_i$
    FOR ALL those $\vec{\alpha}$
        $J \leftarrow J \cup \left\{\prod_{j=1}^{i-1} I_j^{\alpha_j}\right\}$
    IF I_i is NOT a linear combination of the invariants of J
        $I' \leftarrow I' \cup \{I_i\}$
    $i \leftarrow i + 1$

Checking if an invariant is a linear combination of a set of invariants comes down to solving the nullspace of another linear system. All monomial terms are assigned a dimension in a vector space. The values of the coefficients of these terms are the values in their corresponding vectors. Then by elementary linear algebra, determining linear dependence is just a matter of determining the dimension of the nullspace of the matrix with the vectors of I_i and the contents of J as columns. As an example, consider step 3 in the algorithm:

$$J_1 = 2x^2 + 4y \quad J_2 = -x^2 + 2\frac{y^2}{x^2} \qquad I_3 = y + \frac{y^2}{x^2} \tag{5.60}$$

Assigning x^2 to the first dimension, y to the second and $\frac{y^2}{x^2}$ to the third, the corresponding vectors are:

$$J_1 = \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix} \quad J_2 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \qquad I_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \tag{5.61}$$

We place the vectors in a matrix and perform Gaussian elimination, noting that it is only square by accident in this case. The procedure is identical for non-square matrices.

$$\begin{pmatrix} 2 & -1 & 0 \\ 4 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & -1 & 0 \\ 0 & 2 & 1 \\ 0 & 2 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & -1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.62}$$

This matrix clearly has a nullspace[23] leading to the conclusion that I_3 is indeed a linear combination of the invariants in J.

5.7 The Full Method

We finally present an outline of the full algorithm that implements the methodology described in this chapter. Details of the implementation will be discussed in the next chapter.

Algorithm 3 The full algorithm

Input: A set of β-functions $B = \{\beta_1, ..., \beta_n\}$. A searching set of dimensionality vectors $D = \left\{\vec{d}_1, ..., \vec{d}_t\right\}$ of size r. Some method of restricting monomial powers labeled p.
Output: A set of unique invariants $I = \{I_1, ..., I_m\}$.

$I \leftarrow \emptyset$
Use eq. (5.3) to construct the system of equations for $\vec{a}$. (MS)
Solve the system of equations, finding the set of all monomial invariants M.
$I \leftarrow I \cup \{M\}$
$i \leftarrow 0$
WHILE $i \leq t$
    Compute $\mathcal{M}_p(\vec{d}_i)$. (PS)
    Use $\mathcal{M}_p(\vec{d}_i)$ and eq. (5.36) to construct the system of equations for $\vec{C}$.
    Solve the system, finding a set of polynomial invariants P_i.
    $I \leftarrow I \cup \{P_i\}$
    $j \leftarrow 1$
    WHILE $j \leq n$
        Use $\mathcal{M}_p(\vec{d}_i)$ and eq. (5.47) to construct the system of equations for $\vec{C}$ and the factorized power b_j. (FPS)
        Find the eigenvalues for b_j and corresponding eigenvectors for $\vec{C}$.
        Construct the resulting invariants Q_{ij}
        $I \leftarrow I \cup \{Q_{ij}\}$
        $j \leftarrow j + 1$
    $i \leftarrow i + 1$
Filter I using Algorithm 2.

[23] It is spanned by $\begin{pmatrix} 1 \\ 2 \\ -4 \end{pmatrix}$, but this is not particularly important now.

The acronyms MS, PS and FPS refer to the names of the sections of this chapter that contain the relevant methods. We will refer to these separate parts of the algorithm with the acronyms from now on. Note that this algorithm will usually add identical invariants to I multiple times. The filtering algorithm conveniently takes care of this.

Details about the implementation of the algorithm can be found in the next chapter. The program implementing this algorithm can be found at [8].

5.8 Two-loop Contributions and Pi-counting

As we have seen, β-functions can be computed up to some loop order. Consider the expression for a general β-function of some theory to higher loop orders:

$$\beta_{x_i} = \frac{1}{16\pi^2}\beta_{x_i}^{(1)} + \frac{1}{(16\pi^2)^2}\beta_{x_i}^{(2)} + ... \tag{5.63}$$

If we only consider invariants of the one-loop part of the β-function, the factor $\frac{1}{16\pi^2}$ is irrelevant. However, if the two-loop or higher parts are included, this factor is significant. Let us take the case of all β-functions up to two loop orders. We have already established that it is sufficient to let $\mathbf{a}\epsilon\mathbb{Z}^{m\times n}$ and $\vec{C}\epsilon\mathbb{Z}^n$. The factor $\frac{1}{\pi^2}$ is then the only irrational number appearing in the problem. It allows us to introduce a boundary between terms included in an invariant. Let I be an invariant of some set of β-functions up to two loop orders. Suppose it is possible to write:

$$I = I_1 + \frac{1}{16\pi^2}I_2 \tag{5.64}$$

Taking the derivative, we find:

$$\frac{dI}{dt} = \frac{1}{16\pi^2}I_1^{(1)} + \frac{1}{(16\pi^2)^2}\left(I_1^{(2)} + I_2^{(1)}\right) + \frac{1}{(16\pi^2)^3}I_2^{(2)} \tag{5.65}$$

where $I_i^{(j)}$ is the part of the derivative that involves the j-th loop order β-function $\beta_{x_i}^{(j)}$. The factors of $\frac{1}{16\pi^2}$ have been extracted and the terms are grouped by their powers of this factor. Since $I_i^{(j)}$ contains only factors from $\mathbb{Z}$, these separated terms cannot mix. As a consequence, we learn that $I_1^{(1)}$ and $I_1^{(2)} + I_2^{(1)}$ must be separately zero. The final term $I_2^{(2)}$ is preceded by a factor $\frac{1}{(16\pi^2)^3}$, indicating it is a three-loop order effect. This can be seen easily by considering the fact that it would mix with other contributions to the derivative if the three-loop parts of the β-function would be included. If we are interested in invariants of β-functions up to two-loop order, the contribution of $I_2^{(2)}$ should be neglected.

To see how our method changes by adding two-loop contributions, we consider another toy system:

$$\begin{aligned}
\beta_x &= \frac{1}{(16\pi^2)}\left(-2x^2+2y^2\right)+\frac{1}{(16\pi^2)^2}\left(-x^3-xy^2\right)\\
\beta_y &= \frac{1}{(16\pi^2)}\left(x^2-y^2\right)+\frac{1}{(16\pi^2)^2}xy^2\\
\beta_z &= \frac{1}{(16\pi^2)}\left(-5x^3+2x^2y+5xy^2-2y^3\right)
\end{aligned} \tag{5.66}$$

Two-loop contributions for z are not included since they will not be relevant for the example. It can easily be checked that the system of eq. (5.66) has two invariants at one-loop order:

$$F_1 = x+2y \qquad F_2 = 5x^2+4y^2-4z \tag{5.67}$$

We will look for a two-loop contribution to the invariant F_1. We can assign a dimension of 1 to x and y and 2 to z. It can then be checked that:

$$\dim\left(\beta_x^{(1)}\right)-\dim(x)=\dim\left(\beta_y^{(1)}\right)-\dim(y)=\dim\left(\beta_z^{(1)}\right)-\dim(z)=1 \tag{5.68}$$

as is required for a dimensionality. However, for the two-loop contributions, we find:

$$\dim\left(\beta_x^{(2)}\right)-\dim(x)=\dim\left(\beta_y^{(2)}\right)-\dim(y)=2 \tag{5.69}$$

A difference in dimensionality between loop orders is typical in theories like the SM or the (p)MSSM. In those theories, the constant for the coupling dimension is 2 for one-loop order and increases in steps of 2 for higher loop orders. If we want terms of $I_1^{(2)}$ and $I_2^{(1)}$ to mix, we will need to include terms of different dimensionalities. For our toy system, consider:

$$\mathcal{M}_p(1)=\{x,y\} \quad \text{and} \quad \mathcal{M}_p(2)=\left\{x^2,y^2,z,xy\right\} \tag{5.70}$$

Using these sets of monomial terms, we consider the requirement:

$$I_1^{(1)}=0 \quad \text{and} \quad I_1^{(2)}+I_2^{(1)}=0 \tag{5.71}$$

The linear equations that follow from both requirements in eq. (5.71) are placed in a matrix, using the terms of $\mathcal{M}_p(1)$ for I_1 and $\mathcal{M}_p(2)$ for I_2. The result is:

$$\begin{pmatrix}
-2 & 1 & 0 & 0 & 0 & 0\\
-2 & 1 & 0 & 0 & 0 & 0\\
-1 & 0 & -4 & 0 & -5 & 1\\
-1 & 1 & 4 & 0 & 5 & -1\\
0 & 0 & 0 & 2 & 2 & -2\\
0 & 0 & 0 & -2 & -2 & 2
\end{pmatrix}
\begin{pmatrix}
C_x\\ C_y\\ C_{x^2}\\ C_{y^2}\\ C_z\\ C_{xy}
\end{pmatrix}=0 \tag{5.72}$$

By a single step of Gaussian elimination and removing identical rows, the system can be reduced to:

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8 & 0 & 10 & -2 \\ 0 & 0 & 0 & 2 & 2 & -2 \end{pmatrix} \begin{pmatrix} C_x \\ C_y \\ C_{x^2} \\ C_{y^2} \\ C_z \\ C_{xy} \end{pmatrix} = 0 \tag{5.73}$$

Solving the nullspace yields the following invariants:

$$\begin{aligned} S_1 &= -4x - 8y + \frac{1}{16\pi^2}x^2 \\ S_2 &= \frac{1}{16\pi^2}\left(5x^2 + 4y^2 - 4z\right) \\ S_3 &= \frac{1}{16\pi^2}\left(x^2 + 4y^2 + 4xy\right) \end{aligned} \tag{5.74}$$

Why do we find three invariants while we only expect the two-loop continuation for F_1? The reason is that there is another way of solving eq. (5.71) by setting $I_1 = 0$ and $I_2^{(1)} = 0$, or in other words, the two-loop contribution to the invariant $I_1 = 0$ is a one-loop invariant. We search for I_2 at dimensionality 2, where both F_1^2 and F_2 are one-loop invariants. Thus, we find these too. In this case, the invariants are neatly separated. S_1 is in fact the two-loop continuation of $4F_1$ and the other two have $I_1 = 0$. Invariants like S_2 and S_3 can be eliminated by removing any invariants that have $I_1 = 0$. However, this does not entirely fix the problem. In general, any linear combination of the invariants S_1, S_2 and S_3 could have been found. As long as S_1 is included in such a linear combination, the invariant will not have $I_1 = 0$ and thus will not be filtered out by the above procedure. However, by finding a new basis for the nullspace, the unwanted invariants can still be located and removed.

6 Implementation

This chapter will discuss some noteworthy details of the implementation of the method outlined in the previous chapter. The implementation was programmed in C++11[24] making no use of any external libraries other than the C++ Standard Library. It consists of roughly 3700 lines of code including the β-functions of the SM and MSSM, which can easily be reduced to those of the pMSSM or other simplified models. The code has been built with the intent of usage on a 64 bit system[25]. The source code can be found at [8].

6.1 Computing $\mathcal{M}_p(\vec{d})$

Let us denote the j-th dimensionality of a parameter x_i with y_{ij}. Then $\vec{y}_i$ is the vector of the j-th dimensionalities of $\vec{x}$. The problem of computing $\mathcal{M}_p(\vec{d})$ can then be formulated as:

$$\text{Find all } \vec{a}\epsilon\mathbb{Z}^n \text{ subject to } p \text{ with } \forall j\epsilon\,[1,...,r]:\ \vec{a}\cdot\vec{y}_j = d_j \tag{6.1}$$

As explained previously, p can be any limit on $\vec{a}$ that keeps $\mathcal{M}_p(\vec{d})$ finite. Consider for instance the case where the requirement is that for a given $\vec{p}\epsilon\mathbb{Z}^n$ and $\mathbf{P}\epsilon\mathbb{Z}^{n\times n}$:

$$\forall i\epsilon\,[1,...,n]:\ \sum_{j=0}^{n} P_{ij}|a_i| < p_i \tag{6.2}$$

This particular implementation is somewhat generalized to allow for a comparison below. A very practical example of such an implementation is $\mathbf{P} = \mathbf{I}$, since it still allows us to tweak $\vec{p}$ to apply different limitations to the powers of different parameters. Eq. (6.1) is then:

$$\text{Find all } \vec{a}\epsilon\mathbb{Z}^n \text{ with } \forall i\epsilon\,[1,...,n]:\ |a_i| < p_i \text{ and } \forall j\epsilon\,[1,...,r]:\ \vec{a}\cdot\vec{y}_j = d_j \tag{6.3}$$

Using eq. (6.2), problem (6.1) is in fact a more complex form of a class of problems known as *Integer Programming* [13]*:*

$$\text{Maximize } \vec{a}\cdot\vec{b} \text{ with } \forall i\epsilon\,[1,...,n] : (\mathbf{A}\vec{x})_i < c_i \text{ and } \vec{a}\epsilon\mathbb{Z}^n \tag{6.4}$$

for $\mathbf{A}\epsilon\mathbb{Z}^{n\times n}$ and $\vec{b}$, $\vec{c}\epsilon\mathbb{Z}^n$. This class of problems is known to be NP-hard, meaning there is no efficient way to solve it. The most straightforward way to tackle the problem is by brute force: we generate all possible $\vec{a}$ that satisfy eq. (6.2), which is a finite set. They are then checked to satisfy $\forall j\epsilon\,[1,...,r] : \vec{a}\cdot\vec{y}_j = d_j$. If they do, they are added to $\mathcal{M}_p(\vec{d})$.

[24] Also known as C++0x.

[25] This can be relevant for the amount of memory assigned to standard data types.

The current implementation has a restriction p that is slightly different from what is described above. $\mathbf{P} = \mathbf{I}$ and p is just a single number such that eq. (6.2) becomes:

$$\forall i \epsilon [1, ..., n] : \ |a_i| < p \tag{6.5}$$

In addition, the number of nonzero values in $\vec{a}$ is restricted. This configuration gives the user even more control over the size of $\mathcal{M}_p(\vec{d})$. It also has the advantage that computation of candidate terms of $\mathcal{M}_p(\vec{d})$ can be done slightly more efficiently. While this problem is not as complex as eq. (6.4), the easiest approach is still to implement a brute force method, albeit somewhat more clever than what was mentioned before.

Let t be the number of allowed nonzero powers of parameters. Let n be the number of renormalized parameters as usual. The candidates for $\mathcal{M}_p(\vec{d})$ can then be calculated by generating the sets:

$$\mathcal{G} = \left\{ \vec{g} \epsilon \mathbb{N}^t | \forall i \epsilon [1, ..., r] : \ g_i \leq n \ , \ \forall i \epsilon [1, ..., r] : g_i > g_{i-1} \right\} \tag{6.6}$$

$$\mathcal{H} = \left\{ \vec{h} \epsilon \mathbb{Z}^r | \forall i \epsilon [1, ..., r] : \ |h_i| < p \right\} \tag{6.7}$$

$\mathcal{G}$ is the set of vectors that select the renormalized parameters with a nonzero power. The first requirement makes sure all elements of a $\vec{g}$ correspond with a parameter. The second ensures that every combination of parameters only appears once. $\mathcal{H}$ is the set of powers without yet specifying to what parameters these powers belong. A candidate element of $\mathcal{M}_p(\vec{d})$ is then a pair of a $\vec{g}$ and an $\vec{h}$. For instance, let $r = 3$. A possible monomial of an element of $\mathcal{M}_p(\vec{d})$ would be:

$$\vec{g} = \begin{pmatrix} 3 \\ 5 \\ 9 \end{pmatrix} \quad \text{and} \quad \vec{h} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} \rightarrow \frac{x_3^1 x_5^2}{x_9} \tag{6.8}$$

The sets $\mathcal{G}$ and $\mathcal{H}$ are easy to produce using a simple recursive function. The procedure of finding $\mathcal{M}_p(\vec{d})$ is now just a matter of checking every pair of elements of $\mathcal{G}$ and $\mathcal{H}$. An added benefit of this method is a reduction of required storage: the sets $\mathcal{G}$ and $\mathcal{H}$ require much less storage than a full set of candidates for $\mathcal{M}_p(\vec{d})$.

6.2 Sparse Matrix Storage and Handling

A good method of storing and handling large matrices is a cornerstone of the implementation of our method of finding invariants. As was stated previously, the method p of restricting powers should be chosen such that the set $\mathcal{M}_p(\vec{d})$ is as large as possible. Every element of the set $\mathcal{M}_p(\vec{d})$ corresponds with a column

in the matrices we generate in PS and FPS. The number of rows is dependent on the specific form of the β-functions, but it is typically a factor of $\mathcal{O}(10)$ larger than the number of columns. The number of monomial terms that are produced by taking the derivative of one of the terms from $\mathcal{M}_p(\vec{d})$ is typically $\mathcal{O}(10)$ to $\mathcal{O}(100)$ for one- and two-loop β-functions. Since the number of terms in $\mathcal{M}_p(\vec{d})$ can easily be $\mathcal{O}(10^5)$ or even $\mathcal{O}(10^6)$, most columns contain a factor of $\mathcal{O}(10^4)$ to $\mathcal{O}(10^5)$ more zeroes than nonzero elements. Obviously, it is extremely inefficient to store all of these zeroes as if the matrix is a regular matrix.

Matrices that contain many zeroes are called *sparse*. Efficient storage of sparse matrices can be done in several different ways. We will briefly discuss some of these methods before selecting one most suitable for our purpose [9, 10]. We will take the matrix to have n rows, m columns and a *sparsity* s, defined as $\frac{\#\text{nonzero elements}}{nm}$. When discussing required storage, we will assume all numbers to be stored in long format, which has a size of 8 bytes in C++ on a 64 bit machine. Those 8 bytes are referred to as a unit of storage.

Other criteria that are relevant include ease of access to a row or column, and ease of performing *fill-in*, which is the insertion of new nonzero elements. As an example of the storage techniques below, we show the following matrix in those formats:

$$\begin{pmatrix} 0 & 3 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 2 \end{pmatrix} \tag{6.9}$$

One very obvious way to store a sparse matrix is as a list of triplets of the form of $\begin{pmatrix} row \\ col \\ value \end{pmatrix}$ for every nonzero entry. In this format, the matrix of eq. (6.9) is:

$$\left(\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 2 \end{pmatrix} \right) \tag{6.10}$$

This method is typically known as the *Coordinate list (COO)*. Its storage requirement is $3nms$ disregarding any overhead storage required for the list of the triplets, since it should always be negligible. Depending on how it is used, COO allows for very easy insertion of new nonzero elements[26]. However, when a procedure such as Gaussian elimination is required, access to all elements in a row or a column is difficult. This issue can be partly remedied by sorting the list of tuples with respect to row, then column or column, then row. Finding a row or column then still requires calling some sort of searching algorithm to find the first triplet of a certain row or column. It also means that for every

[26] You just add it to the bottom of the list.

fill-in, a similar search has to be performed to find the correct place of insertion. Searching algorithms such as binary search have an average complexity of $\mathcal{O}(\log(size))$ [11], which in this case is $\mathcal{O}(\log(nms))$. Logarithmic complexity is not steep, but it is significant. Keeping in mind that processes like Gaussian elimination require $\mathcal{O}(n^2 m)$ on a regular matrix, COO is not a good type of storage for our purpose.

Next is the *Compressed Row Storage (CRS)*. This format stores only the doublets $\begin{pmatrix} col \\ value \end{pmatrix}$. The row structure is maintained through another list that contains the indices (or pointers) of the list of doublets that mark a new row. Eq. (6.9) is:

$$\left(\begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \end{pmatrix} \right) \qquad (2, 4) \tag{6.11}$$

Meaning doublet 2 and 4 start a new row. CRS has storage requirement of $2nms + n$ where we can again neglect overhead. This is lower than that of COO as long as $ms > 1$. Access to a row is significantly easier, since the i-th element of the list of indices directly indicates the starting doublet of the i-th row. If access to columns is required, it is of course also possible to introduce CCS which just flips the rôle of the row and the column. A major downside is that fill-in is a much more expensive procedure. If an element is filled into row i, then for all $i > n$, the elements of the index row must be increased by one. This procedure seems mundane, but turns out to be a major bottleneck.

Finally, the *List of Lists (LIL)*. It uses the same doublets as CRS, but instead of storing them in a single list, the doublets are stored in a separate list for every row. The matrix of eq. (6.9) is:

$$\begin{pmatrix} & \begin{pmatrix} 2 \\ 3 \end{pmatrix} & \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, & & \begin{pmatrix} 3 \\ 1 \end{pmatrix} \\ & \begin{pmatrix} 3 \\ 2 \end{pmatrix} & \end{pmatrix} \tag{6.12}$$

LIL adds extra structure to the storage compared to the previous methods. The storage requirement of this method is dependent on what kinds of lists are used. The most basic array requires no additional storage space, but operations such as insertion are always of $\mathcal{O}(size)$, since they basically come down to building a new list and replacing the old one. The C++ Standard Library contains a number of custom containers with different properties. The most efficient is the *vector*, which requires 3 longs[27] of overhead storage. Advantages include $\mathcal{O}(1)$ insertion and removal at the back of the container, which is very important in the procedure of creating the matrix, and an $\mathcal{O}(1)$ function that returns the

[27] Or 32 bytes.

size of the container [12]. Using vectors as our lists, the storage requirement is $n(2ms + 3)$.

On the other hand, we will need a method to access columns of a matrix stored in LIL for system solving methods later on. A possible way to do this is to insert links between elements of the same column. These links would be pointers, which are of the same size as a long in C++ on 64 bit machines. Adding two links to every element increases the storage requirement to $n(4ms+3)$. The downside is that it makes insertion much more expensive. If an element is ever inserted anywhere, a search through the column of that element has to be performed. When the correct location in the chain of links is found, the links of the new element and the surrounding elements have to be adjusted. Additionally, since an element is added to one of the rows, all pointers to the elements of that row become invalid and have to be adjusted. This whole procedure has to be performed every time an element is inserted, making it very inefficient.

A different way to approach this is to store the matrix twice: once in *row-dominant* form and once in *column-dominant* form. Since the elimination algorithms later on will only require access to the information of which rows have nonzero elements in a certain column, we do not actually have to store the full matrix in column dominant form. Instead, we only store the row indices. Our example matrix is now:

$$\begin{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 3 \\ 2 \end{pmatrix} \end{pmatrix} \qquad \begin{pmatrix} (2) \\ (1) \\ (2,3) \end{pmatrix}$$

To clarify, the second structure tells us that, for instance, in the third column, there is a nonzero element in rows 2 and 3. This is all the information the algorithms later on will need. The second storage structure is updated simultaneously with the first. The storage requirement of this method jumps up to $2nms + 3(m + n)$, which is quite steep. The advantage is that this implementation of LIL offers both fast access to rows and columns, and fast fill-in. It comes at the cost of increased memory usage, but this is compensated by better performance. LIL is therefore the implemented method.

6.3 System Creation and the Cantor Pairing Function

MS, PS and FPS all require some form of linear system building. In theory, the procedure is to write down the derivative of some monomial or polynomial, and then bring all identical terms together and require them to cancel. In reality, storing the full derivative polynomial is unnecessary. It would waste large amounts of storage at little to no avail. Instead, the program calculates all terms one at a time and immediately places them in the correct row of the

matrix. In order to be able to do that, it needs to 'remember' to which rows monomial terms were assigned previously. The object that takes care of this is a *map*. Maps are associative containers that store elements formed by a *key* and a *value*. At any point, the value corresponding with a key can be requested. In our case, we need monomial terms as keys to be mapped to an integer value representing the row number.

Let us consider the representation of monomial terms as introduced in the first section of this chapter: as a combination of elements from $\mathcal{G}$ and $\mathcal{H}$ that have been checked for dimensionality. The elements of these sets are vectors that contain relatively small integers. A theory like the MSSM has $\mathcal{O}(10)$ to $\mathcal{O}(100)$ parameters, so the integers of elements in $\mathcal{G}$ will never exceed those numbers. Given that the set $\mathcal{M}_p(\vec{d})$ needs to be kept manageable, the integers of elements from $\mathcal{H}$ will also be $\mathcal{O}(10)$. The smallest native data type of C++ is the char, which has a size of one byte. Especially in the case of elements from $\mathcal{H}$, but also for $\mathcal{G}$ in theories with less parameters, a byte of memory is much more than required. In case of the MSSM a larger data type is required for elements of $\mathcal{G}$, which are then immediately much larger than required. In addition, storage of vectors from $\mathcal{G}$ and $\mathcal{H}$ introduces some overhead memory usage we would rather avoid storing. Since the number of rows is typically a multiple of the number of columns in the relevant matrices, and this map has as many entries as the number of rows, it actually turns out that the map uses much more storage than the matrix itself. This memory usage is also the main bottleneck to the program, so it pays off to find a different way of storing elements of the map.

We define the *Cantor pairing function* on two positive integers [14]:

$$\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N} : \quad \pi\left(n_1, n_2\right) = \frac{1}{2}\left(n_1 + n_2\right)\left(n_1 + n_2 + 1\right) + n_2 \tag{6.13}$$

A pairing function is a process that uniquely encodes two natural numbers into one. The Cantor pairing function is a bijection that will allow us to reduce a monomial term of the form of an element of $\mathcal{G}$ and $\mathcal{H}$ to a single positive number. Recall that the size of elements from $\mathcal{G}$ and $\mathcal{H}$ is t. Call the maximum number of nonzero powers occurring in the set of β-functions s. The maximum number of nonzero powers that can then occur in a derivative monomial term is $s + t \equiv u$. Such a term can then be represented by two vectors $\vec{g}\epsilon\mathbb{N}^u$ and $\vec{h}\epsilon\mathbb{Z}^u$ in the same way we described monomial terms as elements of $\mathcal{G}$ and $\mathcal{H}$. The following algorithm maps these two vectors to a unique number. We denote the modulo operator with %.

Algorithm 4 Storage reduction using the Cantor pairing function.

Input: $\vec{g}\epsilon\mathbb{N}^u$, $\vec{h}\epsilon\mathbb{Z}^u$, $u\epsilon\mathbb{N}$
Output: A unique $x\epsilon\mathbb{N}$ for the vectors $\vec{g}$ and $\vec{h}$.

$v \leftarrow 2^{\lceil \log_2(u) \rceil}$
$\vec{g}' \leftarrow (g_1, ..., g_u, 0, ..., 0)\,\epsilon\mathbb{N}^v$
$\vec{h}' \leftarrow (2|h_1| + |h_1|\%2, ..., 2|h_u| + |h_u|\%2, 0, ..., 0)\,\epsilon\mathbb{N}^v$
$\vec{w} \leftarrow (\pi(g'_1, h'_1), ..., \pi(g'_v, h'_v))\,\epsilon\mathbb{N}^v$
$i \leftarrow 1$
WHILE $i \leq \log_2(v)$
    $j \leftarrow 1$
    WHILE $j \leq v \cdot 2^{1-i}$
        $w_{\frac{j+1}{2}} \leftarrow \pi(w_j, w_{j+1})$
        $j \leftarrow j + 2$
    $i \leftarrow i + 1$
$x \leftarrow w_1$

This algorithm first finds v, the closest powers of 2 rounded up for the value of u. The vectors $\vec{g}$ and $\vec{h}$ are then extended by adding zeroes until they are of this size. The vector $\vec{h}$ is converted to $\vec{h}'\epsilon\mathbb{N}^v$ by the doubling of the absolute values of the components and adding 1 for odd numbers. Next, the Cantor pairing function is used to produce a single unique vector $\vec{w}\epsilon\mathbb{N}^v$. The elements of $\vec{w}$ are then repeatedly concatenated pairwise until a single number is left, which is the unique identifying number x. The extension of the vectors to size v is required to make sure they exactly fit the procedure of pairwise concatenation. This procedure is preferred over just recursively pairing the two leftmost entries of $\vec{w}$ because it leads to smaller x. In any case, for x to be unique, we must ensure that all vectors are of the same length which is automatically guaranteed in this procedure.

The resulting number x is typically still very large. The current implementation stores x in an 16-byte integer, which is still lower than the overhead that would be introduced by storing $\vec{g}$ and $\vec{h}$. Even though Algorithm 4 needs to be called for every new monomial term, the performance actually increases because the lookup time of the map drastically decreases. In fact, the performance and the memory usage can be made even better by storing x in a smaller data type. The value of x is then modulated by the size of this data type, which results in values that are quite[28] evenly distributed, and thus Algorithm 4 can be used as a *hash function* [11]. Hash functions are used in cryptography to encrypt all sorts of data. A very important property of a hash function is that it returns a very different value if the function argument is altered only slightly, which is the case for Algorithm 4. Another important property is to avoid *collisions*, e.g. different function arguments that return the same value. This issue is similar to the well-known *birthday problem*: the probability of hitting an identical hash

[28] Proving that they are exactly evenly distributed is a whole other matter.

for separate function arguments increases very quickly with the number of used arguments. However, if we assume that the numbers are evenly distributed along a range of, say, an 8-byte integer, then for a probability of a collision of $\mathcal{O}(10^{-18})$, about $\mathcal{O}(10^{10})$ function arguments, or different monomial structures, are required [15]. This probability jumps up quite far if a 4-byte integer would be used, but it typically remains around $\mathcal{O}(10^{-6})$ for typical numbers of monomial structures. The current implementation uses an 8-byte number, but it is thus possible to switch to a 4-byte integer with fairly low risk of collisions if required from a memory perspective.

A possible use for this is replacing the regular map with a *hash map*. This map stores a hash of a key along with the key and value. This hash is used to reach $\mathcal{O}(1)$ lookup times, while regular maps have $\mathcal{O}(\log(\text{size}))$ lookup time. The current application does not use a hash map because the, although relatively small, extra storage required for the hashes is more relevant than the increase in performance.

6.4 System Solving and Markowitz Pivoting

The field of sparse system solving consists of a large number of different methods. These methods can generally be classified into two areas: iterative methods and direct methods. The category of iterative algorithms contains many subcategories that are suitable for matrices with certain properties, such as being symmetric. However, these methods always require working with matrices in $\mathbb{R}^{m\times n}$. The matrices we encounter in MS, PS and FPS are all in $\mathbb{Z}^{m\times n}$. In terms of both speed and accuracy of results, this is a massive advantage we would prefer to maintain. Thus, we consider direct methods [17]. Matrix decomposition such as LU and QR decomposition are usable on sparse matrices. Techniques that make these decompositions viable options are the Strassen algorithm for matrix multiplication [18] or multifrontal methods [19], which can make better use of the multithreading possibilities of modern computers and is still a very active field of research.

As it turns out, the matrices we deal with in PS and FPS have some special structure. This structure is much easier to exploit using a solving method much simpler than the above options. It will also maintain matrices in $\mathbb{Z}^{m\times n}$ without much trouble.

Our method will be a version of fraction-free Gaussian elimination with complete pivoting. When determining the nullspace of a matrix, the rows of that matrix can be interchanged freely. The columns can be interchanged as well, but this requires interchanging the elements of the solution vector in a similar fashion. Complete (or maximal) pivoting usually refers to the procedure of pivoting of rows and columns throughout the whole matrix to achieve optimal reduction of round-off errors. Since we are not dealing with round-off errors in $\mathbb{Z}^{m\times n}$ matrices at all, pivoting has a different purpose. Instead, it is used to reduce the amount of fill-in during elimination. The reduction of fill-in is extremely

important for all methods of sparse solving. Not only does fill-in take time, but it more importantly reduces the effectiveness of subsequent steps in the algorithm. The more entries are kept zero, the less computations the method will have to perform, and the less storage is used.

With the goal of keeping fill-in at a minimum, let us consider step k of the procedure of Gaussian elimination. That is, the first $k-1$ columns have been eliminated and the effective working area for the rest of the algorithm is the $(m-k)\times(n-k)$ submatrix in the bottom-right. Define $r_i^{(k)}$ as the number of nonzero entries in row i of this submatrix, and $c_j^{(k)}$ as the number of nonzero entries in the column j of this submatrix. The *Markowitz criterion* is then to select the entry to use for elimination that minimizes [17]:

$$\left(r_i^{(k)}-1\right)\left(c_j^{(k)}-1\right) \tag{6.14}$$

There usually are also requirements on the absolute value of this element to keep the algorithm numerically stable, but since we are working in $\mathbb{Z}^{m\times n}$ this is of no concern. The idea here is to select an entry that will produce the lowest amount of fill-in when used to eliminate a column. The entry with the lowest value for eq. (6.14) is selected, is then (implicitly) pivoted to the top-left of the matrix, and regular Gaussian elimination is performed. Note that eq. (6.14) is preferred over $r_i^{(k)}c_j^{(k)}$ because it effectively forces the algorithm to select entries that are row singlets or column singlets. These entries will not produce any fill-in, and will play a major role in our method. The Markowitz criterion is a purely empirical entity that has been shown to provide excellent results.

The concept of row singlets is very important for PS and FSP. It is the reason Markowitz pivoting is such a good strategy for these specific problems. Recall that rows correspond with monomial terms, and the number of rows is typically much larger than the number of columns. As it turns out, at least for the β-functions of the SM and MSSM, row singlets occur very often. This is to be expected, because the consequence of a row singlet is just that the variable corresponding with the column of that nonzero entry must be zero. In other words, some element of $\mathcal{M}_p(\vec{d})$ produced a derivative monomial term that is not produced by any other element of $\mathcal{M}_p(\vec{d})$. Obviously the coefficient of this element must then be zero. Since we expect most coefficients to be zero for any given invariant, the number of row singlets is expected to be quite high[29].

The following algorithm is the first step in the full procedure of Gaussian elimination that makes use of the large number of row singlets. Note that pivoting is done implicitly. Rows are not pivoted upward, but just placed in another matrix. Columns are not pivoted at all. The resulting matrix is not upper triangular, so the algorithm needs to store the information that certain columns have been eliminated. For any matrix $\mathbf{M}$, we denote the i-th row by M_i. If this

[29] It is of course possible to find a zero coefficient from a row that is not a singlet, but this requires some very specific cancellations in the invariant.

row is a singlet, then the column of the nonzero entry is c_i. The rows of the j-th element of c_i are $r\left(c_{i,j}\right)$ for $j\epsilon\left[1,...,s_{c_i}\right]$.

Algorithm 5 Gaussian elimination on row singlets

Input: A sparse matrix $\mathbf{A}\epsilon\mathbb{Z}^{m\times n}$
Output: A matrix $\mathbf{B}\epsilon\mathbb{Z}^{r\times n}$ containing the elimination rows, a smaller $\mathbf{A}\epsilon\mathbb{Z}^{p\times n}$, $\vec{o}\epsilon\mathbb{Z}_2^n$ containing an flag of elimination for all columns

$B \leftarrow \emptyset$
$\vec{o} \leftarrow (0,...,0)$
$i \leftarrow 1$
$Q \leftarrow \emptyset$
WHILE $i \leq m$
    IF A_i is a row singlet
        $Q \leftarrow Q \cup \{i\}$
        $i \leftarrow i+1$
WHILE $Q \neq \emptyset$
    $p \leftarrow$Some element of Q
    $Q \leftarrow Q - \{p\}$
    IF A_p is a row singlet
        $o_{c_p} \leftarrow 1$
        $B \leftarrow B \cup \{A_p\}$
        $j \leftarrow 1$
        WHILE $j \leq s_{c_i}$
            $c_{p,j} \leftarrow 0$
            IF $r\left(c_{p,j}\right)$ is a row singlet
                $Q \leftarrow Q \cup \{r\left(c_{p,j}\right)\}$
            $j \leftarrow j+1$

This algorithm just finds all row singlets in the original matrix, then eliminates the column of the nonzero entry. After eliminating the column, it checks if it made any new row singlets. If that is the case, it adds them to the rows to be checked in the future.

Algorithm 5 is extremely efficient. In the hypothetical case where it would eliminate the entire matrix, the complexity is $\mathcal{O}(nms)$ where s is the sparsity: for all columns, the elements of the column have to be wiped and the corresponding row needs to be checked. These actions are both constant in time[30]. There are an average of ms nonzero elements in a column and this procedure has to be repeated n times. The process becomes easier the further the elimination has progressed since there will be less active rows left, meaning that the complexity is technically multiplied with some constant $c < 1$. This constant depends on the specific form of the matrix.

For the matrices produced in PS, algorithm 5 turns out to eliminate almost the

[30]Meaning they do not rely on the size of the problem in any way.

entire matrix. In cases where matrices have $\mathcal{O}(10^6)$ rows, often only $\mathcal{O}(10^2)$ rows remain. Algorithm 5 has proven to be the single most important factor of efficiency in the current implementation. It has shifted the bottleneck of the program to the memory usage instead of the computer time, and it is usually so efficient that the procedure of solving matrices takes less time than the procedure of building them.

Algorithm 5 never completely solves the system, so a follow-up algorithm that does the rest of the elimination is required. A typical strategy is to do Markowitz pivoting until a small, dense matrix is left [20, 21]. Then, any regular system solving method can be applied to this matrix without taking fill-in or other sparse issues into account. After running algorithm 5, which is a form of Markowitz pivoting, the resulting matrices turn out to still be very sparse. It is therefore better to solve the system completely using regular Markowitz pivoting. Algorithm 6 performs this procedure, starting off where algorithm 5 finished and using the same data structures. It is described somewhat more abstractly to avoid a multitude of indices and technical details. It makes use of the so-called *greatest common divisor* (gcd) of matrix elements to simplify changed rows. The gcd of two numbers in $\mathbb{Z}$ can be calculated using Euclid's algorithm [2], and the gcd of the full row can be found by recursively applying this algorithm.

Algorithm 6 Gaussian elimination with Markowitz pivoting

Input: A sparse matrix $\mathbf{A} \epsilon \mathbb{Z}^{p \times n}$, a matrix $\mathbf{B} \epsilon \mathbb{Z}^{r \times n}$ containing the elimination rows, $\vec{o} \epsilon \mathbb{Z}_2^n$ containing a flag of elimination for all columns
Output: A fully eliminated matrix $\mathbf{B}$

While $\mathbf{A}$ is not empty
    Find the nonzero element a_{ij} of $\mathbf{A}$ with the lowest $(r_i - 1)(c_j - 1)$
    $o_j \leftarrow 1$
    $B \leftarrow B \cup \{A_i\}$
    FOR ALL p with $a_{pj} \neq 0$
        Eliminate a_{pj} from A_p using row A_i
        Reduce A_p using the gcd of its elements

This algorithm always terminates once the matrix is fully eliminated. It is important to maintain nonzero elements that are as low as possible throughout the algorithm. If a large number enters the matrix early on, it can 'spread' to other elements during the elimination procedure. This is the reason for the step of reduction during the elimination.

After running algorithm 6, the vector of flags $\vec{o}$ can be used to find all elements of the nullspace. Every variable that has not been eliminated corresponds with a dimension of the nullspace. Basis vectors for this space can be found by simply setting one of the variables that were not eliminated to one, and the rest to zero.

Backward solving with $\mathbf{B}$ then produces the values of the rest of the variables.

Algorithms 5 and 6 are sufficient to solve the systems encountered in PS. In case of FSP, some adjustments have to be made. Recall that the relevant matrices look like:

$$\mathbf{A} + b\mathbf{B} \tag{6.15}$$

Algorithm 5 needs to search for rows that are singlets of one matrix, and are completely empty for the other. In case of a singlet in $\mathbf{A}$, the procedure is identical. A singlet in $\mathbf{B}$ means that either the variable corresponding with the column is zero, or that b is zero. The case of b is zero is not particularly interesting, since then the problem just reduces to PS. We can therefore assume $b \neq 0$ moving forward, and algorithm 5 remains unchanged other than having to find singlet rows in either $\mathbf{A}$ or $\mathbf{B}$ where the same row in the other matrix is empty.

Algorithm 6 also requires some modification. One might question if the Markowitz count is sufficient for the system $\mathbf{A} + b\mathbf{B}$. Since the elements of this matrix are polynomials in b, it is preferable to incorporate the degree of these polynomials in the Markowitz count. The following criterion is used:

$$\left(r_i^{(k)} - 1\right)\left(c_j^{(k)} - 1\right)(d_{ij} + 1) \tag{6.16}$$

Where d_{ij} is the degree of the element at row i and column j. The $+1$ is there to make sure the count is not zero for elements that have no powers of b. Again, there is no particular mathematical foundation for the use of this criterion other than the fact that it works very well.

Fraction-free Gaussian elimination and gcd computations are both quite trivial in case of matrix elements in $\mathbb{Z}$. Computation of the gcd can just be done through Euclid's algorithm [11]. For FPS, the matrix elements are polynomials in $\mathbb{Z}[b]$. It is again quite important to keep the coefficients in $\mathbb{Z}$, so the computation of a polynomial gcd has to produce polynomials that share this property.

We first introduce two functions on a general univariate polynomial $f \epsilon \mathbb{Z}[x]$:

- $cont(f)$ is the *content* of f. It is the gcd of the coefficients of f.
- $pp(f)$ is the *primitive part* of f. It is defined by $f = cont(f) \cdot pp(f)$.

We also define the *pseudodivision* of polynomials: let c be the leading coefficient of g. Let $d = deg(f) - deg(g) + 1$. Then g pseudodivides f if there exist q, $r \epsilon \mathbb{Z}[x]$ such that $c^d f = gq + r$. The polynomial r is known as the *pseudoremainder* and is denoted as $prem(f, g)$.

The following algorithm, known as the *primitive polynomial remainder sequence algorithm* produces a gcd in $\mathbb{Z}[x]$ [30]:

Algorithm 7 Primitive polynomial remainder sequence algorithm

Input: Polynomials f, $g \epsilon \mathbb{Z}[x]$
Output: $gcd(f,g) \epsilon \mathbb{Z}[x]$.

$f_1 \leftarrow f$
$f_2 \leftarrow g$
$k \leftarrow 2$
WHILE $f_k \neq 0$
    $f_{k+1} \leftarrow pp\,(prem(f_{k-1}, f_k))$
    $k \leftarrow k+1$
$c \leftarrow gcd(cont(f), cont(g)) \epsilon \mathbb{Z}$
$gcd(f,g) \leftarrow c \cdot pp(f_{k-1})$

There is another complication. We cannot simply divide all elements of a row by the gcd of the polynomial elements of that row, since we might be dividing by zero. Consider eliminating M_{ij} of a general matrix $\mathbf{M}$ using row M_k. A method that avoids possible division by zero is:

$$\forall l: \; M_{il} = \frac{M_{kj}}{gcd(M_{kj}, M_{ij})} M_{il} - \frac{M_{ij}}{gcd(M_{kj}, M_{ij})} M_{kl} = \frac{M_{il}M_{kj} - M_{kl}M_{ij}}{gcd(M_{kj}, M_{ij})} \tag{6.17}$$

where algorithm 7 is used to find the gcd of the polynomial elements. Eq. (6.19) can be seen as multiplying the rows by the smallest factor that ensures the elements in the elimination column are equal, and then subtracting them.

7 Results and Conclusion

In this chapter we discuss the application of the implemented methods to some sets of β-functions of well-known theories. We list the invariants the program is able to identify and discuss its performance.

7.1 The SM

Although there are no direct indications of any application for invariants of the SM, other than perhaps the confirmation that the gauge couplings do not unify, it is still interesting to see if any invariants exist. The β-functions were taken from [33]. By diagonalizing the Yukawa matrices and introducing the CKM matrix, a smaller set of β-functions can be derived [31, 32]. The β-functions for the CKM matrix elements cannot easily be assigned a dimensionality, complicating the previously discussed methods. For this reason, the β-functions of the undiagonalized SM were used.

PS and FPS was performed for all combinations of mass dimension and coupling dimension between -6 and 6. The maximum allowed number of nonzero powers was set to 2, which is implicitly increased to 3 by FPS. $\mathcal{M}_p(\vec{d})$ was limited by setting:

$$\mathcal{M}_p(\vec{d}) = \{\vec{a}\epsilon\mathbb{Z}^n | \forall i\epsilon\,[1,...,r] :\ dim_i\,(M\,(\vec{a})) = d_i,\ \forall j\epsilon[1,...,n] :\ |a_j| \leq |\delta_j| + p\} \tag{7.1}$$

where we use the same indices as in chapter 5 and set $p = 2$. Furthermore, δ_j is the element of $\vec{d}$ that corresponds with the nonzero dimensionality of x_j, so it would for instance be the coupling dimensionality element of $\vec{d}$ for all gauge and Yukawa couplings. Table 7.1 lists the invariants of the one-loop β-functions that were found.

Name	Invariant
I_1	$95\frac{1}{g_1^2} + 123\frac{1}{g_2^2}$
I_2	$70\frac{1}{g_1^2} + 41\frac{1}{g_3^2}$

Table 7.1: Invariants of the undiagonalized SM at first loop order

I_1 and I_2 are invariants that are almost trivial given the simple β-functions of the gauge couplings. Invariants of similar form will appear in the MSSM as well. They can be used to construct a sum rule to show that the gauge couplings do not unify in the SM[31]. There are no additional invariants in the full 54-parameter Yukawa sector.

[31] In this case, the sum rule is $111I_1 - 218I_2 = 0$.

Since the parameter space of the SM is still relatively small compared with the full MSSM, the runtime of the program is not significant while searching for invariants. Since the resulting number of invariants is low, the filtering algorithm also takes negligible time.

7.2 The pMSSM

The one-loop β-functions of the MSSM are well known, but higher order results are harder to come by. Derivation of the two-loop β-functions can be found in [34, 35, 36], which are in agreement. In addition, the authors of [36] maintain a website [37] containing most MSSM β-functions up to 3-loop level[32]. The results of these sources were thoroughly compared and no discrepancies were found. The β-functions of the pMSSM can be found by applying the simplifications mentioned in section 3.4 to the β-functions of the MSSM. The same searching parameters as in section 7.1 were used. Table 7.2 lists the results for the first loop order. Since the sfermion mass matrices are assumed diagonal and degenerate in the first two components, we denote them by $m^2_{X1,2}$ and m^2_{X3}. The Yukawa matrices only have a single nonzero element that we will refer to as y_X.

[32]This source has the added benefit of being in a form that is easy to import into the program. With the complexity of the two-loop β-functions, typos are a real problem.

Name	Invariant
F_1	$g_1^{73} g_2^{-297} g_3^{-2816} y_u^{891} y_d^{693} y_e^{330} \mu^{-2013}$
F_2	$\frac{M_a}{g_a^2} \quad a = 1,2,3$
F_3	$5\frac{1}{g_1^2} - 33\frac{1}{g_2^2}$
F_4	$5\frac{1}{g_1^2} + 11\frac{1}{g_3^2}$
F_5	$20M_1^2 - 88M_3^2 + 99m_{u1,2}^2 + 66m_{e1,2}^2$
F_6	$4M_1^2 - 88M_3^2 + 33m_{u1,2}^2 + 66m_{d1,2}^2$
F_7	$2M_1^2 - 198M_2^2 - 88M_3^2 + 99m_{u1,2}^2 - 132m_{L1,2}^2$
F_8	$10M_1^2 + 594M_2^2 - 440M_3^2 + 99m_{u1,2}^2 + 396m_{Q1,2}^2$
F_9	$14M_1^2 + 198M_2^2 - 88M_3^2 + 132m_{H_u}^2 + 297m_{u1,2}^2 - 198m_{u3}^2$
F_{10}	$M_1^2 - 99M_2^2 + 88M_3^2 + 33m_{H_u}^2 + 33m_{H_d}^2 - 99m_{Q3}^2 - 33m_{L3}^2$
F_{11}	$16M_1^2 - 396M_2^2 + 88M_3^2 + 132m_{H_u}^2 + 99m_{u1,2}^2 + 198m_{d3}^2 - 396m_{Q3}^2$
F_{12}	$-3M_1^2 - 33M_2^2 + 88M_3^2 + 22m_{H_u}^2 + 22m_{H_d}^2 - 33m_{u1,2}^2 - 66m_{Q3}^2 - 11m_{e3}^2$
F_{13}	$\frac{1}{M_1}\left[\mathrm{Tr}\left(\mathbf{m}_Q^2 - \mathbf{m}_L^2 - 2\mathbf{m}_u^2 + \mathbf{m}_d^2 + \mathbf{m}_e^2\right) + m_{H_u}^2 - m_{H_d}^2\right]$
F_{14}	$73M_1 - 297M_2 - 2816M_3 - 891A_u - 693A_d - 330A_e + 2013\frac{b}{\mu}$

Table 7.2: Invariants of the pMSSM at first loop order

These invariants are all listed in [28, 39, 40], but often in different linear combinations. The results for the second loop order are listed in Table 7.3.

Name	Invariant
S_1	$3465\frac{M_1}{g_1^2} + \frac{1}{16\pi^2}\left(-2869M_1 - 1485M_2 + 13640M_3 - 3762A_u - 3498A_e + 1518\frac{b}{\mu}\right)$
S_2	$11\frac{M_2}{g_2^2} + \frac{1}{16\pi^2}\left(-M_1 - 209M_2 + 88M_3 - 22\frac{b}{\mu}\right)$
S_3	$693\frac{M_3}{g_3^2} + \frac{1}{16\pi^2}\left(-227M_1 - 3861M_2 + 3586M_3 + 198A_u + 330A_e - 1320\frac{b}{\mu}\right)$

Table 7.3: Invariants of the pMSSM at second loop order

The invariants S_1, S_2 and S_3 represent the two-loop continuation of F_2. As described in section 5.8, the program produces mixed invariants that also include $\frac{1}{16\pi^2}F_{13}$ and $\frac{1}{16\pi^2}F_{14}$ of Table 7.2. These two-loop continuations of F_2 are new, and are the first known invariants of β-functions up to second loop order.

Surprisingly, the filtering algorithm bottlenecks the program when searching through the β-functions of the pMSSM. Because of the large number of invariants, both PS and FPS can find a large number of products of these invariants. The filtering algorithm has to compute all viable products of established unique invariants to find out if new invariants are unique, adding to the computation times.

7.3 The MSSM

Since the pMSSM is a simplified version of the full MSSM, we expect to find less invariants for the β-functions of the full MSSM. In fact, all invariants that are found here must be included in the set of invariants that were found for the pMSSM, including the two-loop continuation of those invariants. Again, the same searching parameters as in section 7.1 were used. Table 7.4 lists the results for the first loop order.

Name	Invariant
G_1	$\frac{M_a}{g_a^2} \quad a = 1,2,3$
G_2	$5\frac{1}{g_1^2} - 33\frac{1}{g_2^2}$
G_3	$5\frac{1}{g_1^2} + 11\frac{1}{g_3^2}$
G_4	$-7M_1^2 + 627M_2^2 - 176M_3^2 - 44m_{H_u}^2 - 11\mathrm{Tr}\mathbf{m}_u^2 - 77\mathrm{Tr}\mathbf{m}_d^2 + 154\mathrm{Tr}\mathbf{m}_Q^2$
G_5	$7M_1^2 + 129M_2^2 - 160M_3^2 - 26m_{H_u}^2 - 14m_{H_d}^2 + 18\mathrm{Tr}\mathbf{m}_u^2 + 42\mathrm{Tr}\mathbf{m}_Q^2 + 7\mathrm{Tr}\mathbf{m}_e^2$
G_6	$M_1^2 + 165M_2^2 - 88M_3^2 - 11m_{H_u}^2 - 11m_{H_d}^2 + 33\mathrm{Tr}\mathbf{m}_Q^2 + 11\mathrm{Tr}\mathbf{m}_L^2$
G_7	$\frac{1}{M_1}\left[\mathrm{Tr}\left(\mathbf{m}_Q^2 - \mathbf{m}_L^2 - 2\mathbf{m}_u^2 + \mathbf{m}_d^2 + \mathbf{m}_e^2\right) + m_{H_u}^2 - m_{H_d}^2\right]$

Table 7.4: Invariants of the full MSSM at first loop order

The invariants G_1, G_2, G_3 and G_7 were already known and can be derived manually with ease [28]. However, G_4, G_5 and G_6 are new invariants that are much more difficult to derive manually because of the size of the β-functions. They were later confirmed in [38] using the symmetries of the underlying theory. Table 7.5 lists the results for the second loop order.

Name	Invariant
T_1	$11\frac{M_2}{g_2^2} + \frac{1}{16\pi^2}\left(-M_1 - 209M_2 + 88M_3 - 22\frac{b}{\mu}\right)$

Table 7.5: Invariants of the full MSSM at second loop order

As it turns out, only S_2 from Table 7.3 survives in the full MSSM. This can be

understood by considering the fact that the parameters $A_x = \frac{a_x}{Y_x}$ are found in S_1 and S_3. These parameters appear, and are defined as such, because they have simpler β-functions than both a_x and Y_x. These combinations of parameters with simpler β-functions cannot be formed in the full MSSM because the Yukawa and trilinear couplings are 3×3 complex matrices instead of single, real numbers.

The application of FPS for both the one-loop and two-loop β-functions is a major bottleneck for the runtime of the program. Computation times reach several hours. The filtering algorithm is again negligible in computation time because there is a low number of invariants.

7.4 Conclusion and Outlook

In this thesis, we have discussed the notion of renormalization group invariants and their application. We have built a computer algebraic method of finding these invariants that can be split up into three separate methods: Monomial Searching, Polynomial Searching and Factorized Polynomial Searching. The application of these methods to the β-functions of the MSSM has led to the discovery of three new invariants at one-loop level. In addition, two-loop continuations of the known invariants $\frac{M_a}{g_a^2}$ for $a = (1, 2, 3)$ at the one-loop level of the pMSSM were found. For the full MSSM, it was found that only the two-loop continuation of $\frac{M_2}{g_2^2}$ survives. One might ask if the existence of these invariants is a coincidence, or if they are implied by the theory. In [38], it was shown that the presence of the new invariants G_4, G_5 and G_6 can be linked to symmetries of the MSSM that manifest themselves in the one-loop β-functions. However, one would expect invariants to exist up to all loop orders if they are truely a property of the theory. No evidence of a two-loop continuation of the invariants G_4, G_5 and G_6 was found. In fact, in the full MSSM, only a continuation of the invariant $\frac{M_2}{g_2^2}$ was found. Continuations for $\frac{M_1}{g_1^2}$ and $\frac{M_3}{g_3^2}$ do exist in the pMSSM, but they seem to be only a consequence of the simplifications that are applied to the MSSM parameter space.

The current implementation of FPS only allows for the factorization of a single parameter at a time. Improving this could potentially lead to a large increase in the ability to reach more complex invariants. However, even if an efficient method for the factorization of multiple variables in a reasonable amount of time could be found, this would lead to other problems. For instance, the number of combinations of factorisable variables increases quickly, and since they all have to be tried for all dimensionality vectors, it will lead to long computation times. Additionally, many duplicates of invariants would be found, potentially in linear combinations, leading to a significant increase in the computation times for the filtering algorithm.

The methods described in this thesis can be applied to any set of β-functions. In particular, it might be interesting to consider the three-loop contributions of the β-functions of the MSSM. The procedure of section 5.8 can easily be extended to three loop, and it would most likely not strain the performace significantly. However, the availability of the three-loop β-functions is a problem. The website [37] includes most of them, but is missing those of the parameters μ and b. In [34, 35, 36], results are presented only up to two-loop order, making it very difficult to perform any kind of check on any three-loop contributions.

The new invariants found in this thesis can be used to find new sum rules that can be used in the determination of a SUSY breaking mechanism, or for other models that incorporate the unification of parameters that appear in these invariants. This was previously done in [28] for the one-loop pMSSM invariants.

A Spinors

The fields of the SM and MSSM are spinor fields. This appendix discusses the notation that is used for these spinors. Both the SM and MSSM violate parity, treating left-handed and right-handed fields differently. The distinction between these fields is expressed differently in the SM and the MSSM. The SM is usually expressed in the language of *Dirac spinors,* while the MSSM is expressed in *Weyl spinors*. We briefly discuss these in the following sections.

A.1 Dirac Spinors

Fermions can be described using Dirac spinors. They are four-component fields that transform with the four-dimensional representation of the Lorentz group. A Dirac spinor ψ is defined as the solutions to the free Dirac equation:

$$(i\gamma^\mu \partial_\mu - m)\,\psi = 0 \tag{A.1}$$

where γ^μ are the *Dirac matrices* (or gamma matrices) that are defined by their anticommutation relation:

$$\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu}\mathbf{I} \tag{A.2}$$

The gamma matrices can be used to construct:

$$\gamma^5 \equiv i\gamma^0\gamma^1\gamma^2\gamma^3 \tag{A.3}$$

Using this definition, left- and right-handed projection operators can be defined:

$$P_L \equiv \frac{1}{2}\left(\mathbf{I} - \gamma^5\right) \quad \text{and} \quad P_R \equiv \frac{1}{2}\left(\mathbf{I} + \gamma^5\right) \tag{A.4}$$

The left- and right-handed components of a Dirac spinor are then:

$$\psi_L \equiv P_L\psi \quad \text{and} \quad \psi_R \equiv P_R\psi \tag{A.5}$$

To build the SM, we also need the *adjoint spinor*:

$$\bar{\psi} = \psi^\dagger\gamma^0 \tag{A.6}$$

which can be used to build invariant quantities $\bar{\psi}\psi$ that can appear in the SM Lagrangian.

A.2 Weyl Spinors

Weyl spinors are two-components fields that are used in the *Weyl representation*. In this representation, Dirac spinors are rewritten as:

$$\psi = \begin{pmatrix} \chi_L \\ \chi_R \end{pmatrix} \tag{A.7}$$

Where χ_L and χ_R are Weyl spinors. They are labeled by L and R because they correspond exactly to the nonzero components of the left- and right-handed components of the Dirac spinor:

$$P_L\psi = \begin{pmatrix} \chi_L \\ 0 \end{pmatrix} \quad \text{and} \quad P_R\psi = \begin{pmatrix} 0 \\ \chi_R \end{pmatrix} \tag{A.8}$$

The MSSM uses a notation that is strictly left-handed. It is possible to transform a right-handed type spinor into a left-handed type spinor using:

$$\chi_R^c \equiv i\sigma^2\chi_R^* \tag{A.9}$$

Now that all fields can be expressed by left-handed spinors, we need to know how to make a Lorentz-invariant quantity out of two of these spinors. This is done using the two-dimensional antisymmetric tensor $\epsilon_{\alpha\beta}$, with components $\epsilon_{12} = -\epsilon_{21} = -1$ and $\epsilon_{11} = \epsilon_{22} = 0$. The invariant quantity for two Weyl spinors χ_1 and χ_2 is then:

$$\chi_1 \cdot \chi_2 \equiv \chi_1^\alpha \chi_{2\alpha} = \chi_1^\alpha \epsilon_{\alpha\beta} \chi_2^\beta \tag{A.10}$$

References

[1] Klaus Jänich, *Linear Algebra,* Springer-Verlag New York (2005).

[2] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C++,* Cambridge University Press (2002)

[3] Ronald Kleiss, *Monte Carlo Integration* `http://www.hef.ru.nl/~kleiss`

[4] Carlos Cid, Gaëtan Leurent: *An Analysis of the XSL Algorithm*, ASIACRYPT 2005. LNCS, vol. 3788, 333–352. Springer, Heidelberg (2005)

[5] Iyad A. Ajwa, Zhojun Liu, Paul S. Wang, *Gröbner Bases Algorithm,* Technical report, ICM Technical Reports Series (ICM-199502-00) (1995)

[6] David Cox, John Little, Donal O'Shea, *Ideals, Varieties and Algorithms,* Springer Science+Business Media, LLC (2007)

[7] Yousef Saad, *Numerical Methods for Large Eigenvalue Problems,* Manchester University Press (1992)

[8] `https://github.com/rbvh/RGEsearch`

[9] Sergio Pissanetzky, *Sparse Matrix Technology,* New York Academic Press (1984)

[10] Ole Østerby, Zahari Zlatev, *Direct Methods for Sparse Matrices,* Springer Verlag (1983)

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to algorithms,* MIT Press (2009)

[12] Nicolai M. Josuttis, *The C++ Standard Library - A Tutorial and Reference,* Pearson Education Inc. (2012)

[13] Laurence A. Wolsey, George L. Nemhauser, *Integer and Combinatorial Optimization,* Wiley-Interscience (1999)

[14] Meri Lisi, *Some Remarks on the Cantor Pairing Function,* Le Matematiche, 62(1) (2007) 55-65

[15] Mihir Bellare, Tadayoshi Kohno, *Hash Function Balance and Its Impact on Birthday Attacks*, Advances in Cryptology - EUROCRYPT '04 (2004) 401–418

[16] Yousef Saad, *Iterative Methods for Sparse Linear Systems,* Society of Industrial and Applied Mathematics (2003)

[17] I. S. Duff, A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices,* Clarendon Press (1986)

[18] Volker Strassen, *Gaussian Elimination is not Optimal*, Numerische Mathematik 13 (1969) 354-356.

[19] Jean-Yves L'Excellent, *Multifrontal Methods: Parallelism, Memory Usage and Numerical Aspects. Modeling and Simulation,* Habilitation à diriger des recherches, École normale supérieure de Lyon (2012)

[20] Roman Pearce, *Solving Sparse Linear Systems in Maple,* (2007) `http://www.mapleprimes.com/posts/41191-Solving-Sparse-Linear-Systems-In-Maple`

[21] W. Bosma, J. J. Cannon, C. Fieker, A. Steel (eds.), *Handbook of Magma functions*, Edition 2.16 (2010)

[22] Henri Cohen, *A Course in Computational Algebraic Number Theory,* Springer-Verlag Berlin-Heidelberg (1993)

[23] M. Peskin, D. Schroeder, *An Introduction to Quantum Field Theory*, Westview Press Colorado (1995)

[24] W. Buchmüller, C. Lüdeling, *Field Theory and SM,* DESY-06-151 (2006) [arXiv:hep-ph/0609174]

[25] S. P. Martin, *A Supersymmetry Primer,* (2011) [arXiv:hep-ph/9709356]

[26] A. N. Schellekens, *Beyond the SM,* (2013), `http://www.nikhef.nl/~t58/BSM2013.pdf`

[27] Betrand Delamotte, *A Hint of Renormalization,* Am.J.Phys. 72 (2004) 170-184 [arXiv:hep-th/0212049]

[28] J. Hetzel, W. Beenakker, *Renormalization Group Invariant and Sum Rules: Fast Diagnostic Tools for Probing High-scale Physics,* JHEP 1210 (2012) 176 [arXiv:hep-th/1204.4336]

[29] K. Ahnert, M. Mulansky, *Odeint - Solving Ordinary Differential Equations in C++*, AIP Conf. Proc. 1389, 1586-1589 (2011)

[30] Ruiyuan Chen, *The Subresultant Polynomial Remainder Sequence Algorithm,* (2013) `www.math.ubc.ca/~reichst/423-project-subresultant.pdf`

[31] C.R. Das, M.K. Parida, *New Formulas and Predictions for Running Fermion Masses at Higher Scales in SM, 2HDM, and MSSM,* Eur.Phys.J.C20:121-137 (2001) [arXiv:hep-ph/0010004]

[32] Stephen G. Naculich, *Third Generation Effects on Fermion Mass Predictions in Supersymmetric Grand Unified Theories,* Phys.Rev. D48 (1993) 5293-5304 [arXiv:hep-ph/9301258]

[33] Mingxing Luo, Yong Xiao, *Two-loop Renormalization Group Equations in the Standard Model*, Phys.Rev.Lett. 90 (2003) 011601 [arXiv:hep-ph/0207271]

[34] Stephen P. Martin, Michael T. Vaughn, *Two-Loop Renormalization Group Equations for Soft Supersymmetry-Breaking Couplings,* Phys.Rev.D50:2282,1994; Erratum-ibid.D78:039903 (2008) [arXiv:hep-ph/9311340]

[35] Youichi Yamada, *Two-loop Renormalization Group Equations for Soft SUSY Breaking Scalar Interactions: Supergraph Method,* Phys.Rev. D50 (1994) 3537-3545 [arXiv:hep-ph/9401241]

[36] I. Jack, D. R. T. Jones, *Soft Supersymmetry Breaking and Finiteness,* Phys.Lett. B333 (1994) 372-379 [hep-ph/9405233]

[37] `http://www.liv.ac.uk/~dij/betas/`

[38] Tom van Daal, *Renormalization Group Invariants in the Minimal Supersymmetric Standard Model,* (2014) `www.ru.nl/publish/pages/517373/thesis_tom_van_daal.pdf`

[39] D. A. Demir, *Renormalization group invariants in the MSSM and its extensions*, JHEP0511:003 (2005) [arXiv:hep-ph/0408043]

[40] M. Carena, P. Draper, N. R. Shah, and C. E. Wagner, *Determining the Structure of Supersymmetry-Breaking with Renormalization Group Invariants*, Phys.Rev. D82 (2010) 075005 [arXiv:1006.4363 [hep-ph]]