Radboud Universiteit

Bachelor thesis Physics and Astronomy

Institute for Mathematics, Astrophysics and Particle Physics (IMAPP)

Zooming in on the computational complexity of the functional renormalization group

Bachelor thesis Physics and Astronomy

Stijn Hennissen
Author

dr. Frank Saueressig
Supervisor

Agustín Silva
Co-Supervisor

December 28, 2025

Abstract

The extraction of physics predictions from an asymptotically safe QFT requires the construction of the UV-critical hypersurface of the underlying renormalization group fixed point. This thesis investigates the computational complexity of the boundary value method as a scalable alternative to the traditional shooting method. The boundary value method reformulates this construction of the UV-critical surface S_{UV} as an optimization problem for generating functions F_μ. Here we solve this problem using a genetic algorithm that has been optimized using modern principles from computer science. To provide a clear example of the functioning of this new method, it has been applied to a scalar field theory in two Euclidean dimensions.

A complexity analysis demonstrates the genetic algorithm's superiority for high-dimensional surfaces. While successfully predicting the $k = 0$ endpoint, the genetic algorithm has trouble producing connected S_{UV} trajectories. The work introduces systematic meta-optimization through testing suites and Pareto analysis.

These algorithmic treatment of the very general mathematical problem offers a broad applicability to asymptotic safety, dynamical systems, fluid dynamics, and more. We also outline future research directions, including adaptive basis functions, neural network based implementations, and applications to theories that offer ties to real world observables, e.g. in the context of asymptotically safe quantum gravity.

Acknowledgments

I would like to sincerely thank Frank Saueressig and Agustín Silva for their help as supervisors, teachers, and discussion partners throughout every stage of this research. I would also like to thank Remco Spelthan for the very fruitful discussion concerning the pilot experiment. Lastly, I would like to thank Ruben Bartelet for the many interesting discussions concerning various parts of this research.

Terminology and notation

Since this thesis considers both a physics perspective as well as a computer science perspective it is important to be clear about specific terminology.

In computer science the term "tensor" is used more freely than in physics. It is used to describe objects including a vector, an array, a list, a matrix, or a tensor in the physics sense. In this text specifically we will use the term tensor to denote a multi-dimensional matrix of real numbers, where we keep careful track of upper and lower indices as to enable the use of Einstein's sum convention. At the same time this careful bookkeeping will enable the use of numeric tensor algebra when implementing the algorithm. This allows for fast computation. Importantly, our use of the term tensor implies no connotation of covariance with respect to coordinate transformations or a change of basis, as one would expect from the use of this terminology in physics.

One could also ask whether it matters so much to have a notation that carries over easily to code, to the extend that it justifies the slightly confusing conventions surrounding non-physical tensors. In principle this is a matter of opinion. However, given the interest of this thesis in advancing the quality of algorithmic research in non-perturbative quantum field theory, we deem it important to have a notation and terminology that is both natural to adopt into code and is consistent with the wider computer science literature, as to encourage collaboration between the fields.

There are some occasions where "tensor" is used in the mathematical sense, for example when referring to the scalar-tensor theory considered in [1]. In these cases it is clear from the context which tensor is meant, since these are not tensors used in the context of code or algorithms.

There will be a lot of indices at play, so we have listed those that have consistent use cases in table 1. Integers N and M are the dimension of the truncated theory space and S_{UV} respectively.

Index	Usage	Range	Examples
n, l	Indices for coordinates in theory space	$n, l = 1, \dots, N$	$\beta^n, g^n, \partial_n F, \partial_n \psi$
m	Index for coordinates on S_{UV}	$m = 1, \dots, M$	β^m, u^m
μ, ν	Indices for the embedding of S_{UV} in $\mathbb{R}^N$	$\mu, \nu = 1, \dots, N - M$	$\beta^\mu, v^\mu, F_\mu, M_\mu, p_\mu$
i, j, k	Indices for collocation points	$i, j, k = 1, \dots, N_p$	$\psi^i, p_i, g^{in}_{col}, g^{jn}_{col}, g^{kn}_{\sim}$

Table 1 Reference table for index conventions used in the thesis.

Additionally, table 2 contains a reference for abbreviations used in the thesis. All abbreviations used are standard in the literature and are frequently used in their respective fields.

Abbreviation	Meaning	First occurrence
QFT	Quantum Field Theory	Start of chapter 1
SM	Standard Model of particle physics	Start of chapter 1
UV	Ultraviolet (high energy)	Start of chapter 1
IR	Infrared (low energy)	Start of chapter 1
RG	Renormalization Group	Start of chapter 1
QED	Quantum Electrodynamics	Start of chapter 1
QCD	Quantum Chromodynamics	Start of chapter 1
QG	Quantum Gravity	Start of chapter 1
FRG	Functional Renormalization Group	Start of chapter 1
EFT	Effective Field Theory	Start of chapter 1
EAA	Effective Average Action	Section 1.1
GFP	Gaussian Fixed Point	Section 1.1
NGFP	Non-Gaussian Fixed Point	Section 1.1
GA	Genetic Algorithm	Section 1.3
MCD	Multivariate Cauchy Distribution	Section 3.2
CVP	Circuit Valve Problem	Section 4.1.3
NN	Neural Network	Section 6.2.3
PINN	Physics Informed Neural Network	Section 6.2.3

Table 2 Reference table for abbreviations used in the thesis.

Contents

Chapter 1

Introduction

Quantum field theory (QFT) is arguably the most complete theoretical model of nature to date. It combines field theory, special relativity, and quantum mechanics into a single framework. QFT finds its uses in particle physics, where it unites three out of the four fundamental forces into the Standard Model of particle physics; as well as in condensed matter physics, where it provides the most complete framework to treat quasiparticles. In the search for new physics, QFT is almost always the natural starting point. A thorough introductory treatment of this topic can be found in [2].

In the treatment of quantum mechanical systems using QFT, one can describe interactions in an elegant manner using Feynman diagrams. The simplest diagrams are those at tree-level. These are distinct from loop diagrams by the fact that these higher order diagrams contain contributions of virtual particles, while tree-level diagrams do not. Internal loops can manifest as a cycle on any vertex of the diagram. Since these loops are made entirely out of virtual particles, they are allowed to be off-shell and as such may take on any energy and momentum. This tipically leads to divergences that are problematic for massive theories are those associated with high energy virtual particles and are termed "ultraviolet" (UV) divergences. For massless theories low energy particle divergences, termed "infrared" (IR) are also problematic. The usual tool to treat these divergences is regularization and subsequently renormalization.

The regularization procedure introduces a *regulator* into the loop-integrand. This term functions as a cutoff with respect to the momentum of the virtual particles inside the loop. The regulator allows the contribution of virtual particle loops up to some energy Λ_{UV}. Above this this cutoff scale the contributions of virtual particle loops are discarded. This can be viewed as averaging out all quantum fluctuations above Λ_{UV}. This leads to the idea of effective field theory, where we can treat macroscopic physics without considering microscopic quantum fluctuations. Regularization effectively assumes there will be new physics at higher and higher energy scales, but that the effects of these new physics can be decoupled on lower energy scales.

Renormalization is a different technique to deal with infinities produced by virtual particle loops. The idea is to split the infinite diverging part of the integrand from the finite converging part. Each part is then associated with their own coupling. The infinite part becomes a *bare* or unobservable coupling, while the finite part becomes the observable coupling. There is some freedom in this procedure, which is fixed by matching the observable couplings to experimental data. This fixes the observable coupling at the energy level of the experiment. The scale at which a coupling is matched

to experiment is called the renormalization point. Changing the renormalization point will change up to which energy level virtual particle loops contribute. This means that observable couplings have different values at different energy scales. This energy dependence is treated formally within the general theory of scale-dependence, called the *renormalization group* (RG). In doing so, one chooses a specific regulator to mediate the cutoff of loop diagrams beyond the renormalization point. The choice of regulator should not affect observable physics, since it is merely a tool, considered to be fundamentally nonphysical, but it will be affected by the choice of renormalization point, as new physics is accessed by raising Λ_{UV}.

While originally used to treat divergences in perturbative approaches to quantum electrodynamics, renormalization turned out to be a self-consistent mechanism to treat the effect of fluctuations in several fields, including condensed matter physics, particle physics, and statistical physics. Following the success of renormalization in quantum electrodynamics (QED) and quantum chromodynamics (QCD), namely the standard model of particle physics, the natural step was to find out if it was also capable of treating gravity as a quantum theory. While the standard model of particle physics contains only renormalizable operators in its Lagrangian, the direct treatment of general relativity as a quantum theory, with the metric in the Einstein-Hilbert action as a perturbation around Minkowski-spacetime, introduces non-renormalizable operators for the fundamental interaction. Non-renormalizable operators are terms in the Lagrangian that lead to exactly those kind of divergences that are problematic. These are divergences whose elimination requires couplings which are not part of the initial bare action. For gravity, non-renormalizable operators come in the form of higher curvature and higher derivative terms. Interactions involving non-renormalizable operators in the Lagrangian can not be treated perturbatively at high energy scales. Since this is the case for gravity, one may suspect that perturbation theory is not the correct treatment for quantum gravity (QG).

The underlying problem ties into the concepts of *asymptotic freedom* and *asymptotic safety*. For some interactions, like QCD, as the energy scale goes to infinity, the physical coupling constant goes to zero. At high energies QCD acts essentially as a free theory. A generalized notion is that of asymptotic safety, for which the physical coupling constants do not need to go to zero in the UV, but must approach some finite fixed value. This behavior is linked to an RG *fixed point*, associated with an interacting QFT, that controls the high energy behavior of the theory. For an asymptotically free or safe theory the UV divergences can be reconciled and physical predictions can be made. This condition for renormalization based on a fixed point was formulated for gravity by Weinberg in 1976 [3], while the initial idea of renormalization using a fixed point of the RG was proposed earlier by Wilson and Parisi for scalar field theories [4, 5]. The quantization of the Einstein-Hilbert action is neither asymptotically free, nor safe, and as such has the status of an effective field theory that breaks down at the Planck scale. This is because the non-finite behavior in the UV introduces divergence that can not absolved by a finite amount of counter-terms, which is possible for the other three fundamental interactions.

Since general relativity is perturbatively non-renormalizable, there has been an effort to construct a non-perturbative, asymptotically safe, quantum theory of gravity using non-perturbative quantum field theory techniques. This builds on the conjecture that gravity possesses a suitable UV fixed point. While initially calculations beyond perturbation theory proved difficult, large step could be

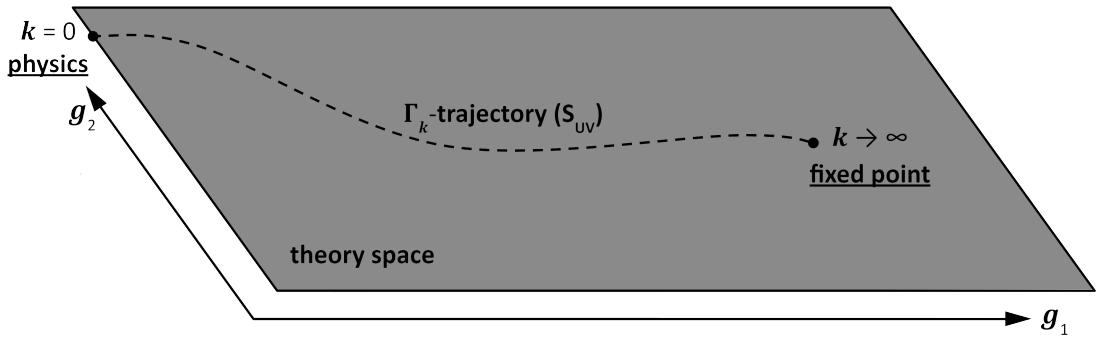


Figure 1.1 An illustrative depiction of theory space and other key elements. The Γ_k-trajectory follows the RG flow of the beta functions through theory space. At the $k = 0$ limit lies the physical endpoint. At the $k \to \infty$ limit lies the UV fixed point. All Γ_k-trajectories that flow into the fixed point in the high energy limit span the UV-critical surface S_{UV}.

made after the introduction of functional renormalization group (FRG) techniques, introduced in 1993 by Wetterich and Morris for scalar theories [6, 7], and by Reuter and Wetterich for general gauge theories on flat Euclidean spacetime [8]. An essential breakthrough came in 1996 when Reuter adapted this research line to gravity [9]. A critical problem was how to determine the existence of a fixed point for gravity for more than two dimensions. This was solved in 2002 by Reuter and Saueressig when they showed that the fixed point in $2 + \varepsilon$ dimensions extends up to four spacetime dimensions [10].

These developments originated from a new perspective on renormalization, proposed by Wilson [11], called *Wilsonian renormalization*. The focus of this new approach lies on the idea of a continuum of scale dependent effective field theories (EFTs), extending from a UV fixed point all the way to the observable physics that are recovered in the low energy, "infrared" (IR), limit. Each theory is encoded in a one-parameter family of *effective average actions* (EAAs) that are valid at some renormalization point, now newly termed the *coarse graining scale*, k, to hammer down the focus on averaging out contributions above this energy scale. Wilson proceeds to place an IR-regulator as cutoff at this variable energy scale k. He then integrates out quantum fluctuations in a narrow shell from k down to $k' = k - \delta k$. He is then left with a new EAA, now defined at coarse-graining scale k'. This procedure is then repeated iteratively to flow from the UV, down to the $k = 0$ limit, where observable physics are recovered. This flow of the EAA along coarse-graining scales is called the RG-trajectory. At any point along the energy scale ladder there is an EFT defined by an EAA that describes effective physics at the coarse-graining scale. The iterative integrating out of fluctuations can be thought of as integrating out contributions of virtual particles shell-by-shell. Wetterich then derived the equation, the *Wetterich equation*, which made this idea tractable to explicit calculations. This equation describes how an EAA changes when fluctuations are integrated out [6]. Utilizing the Wetterich equation has proven to be a very effective way to do calculations following the Wilsonian approach to renormalization.

1.1 Main ideas in asymptotic safety

In treating a QFT in the framework of asymptotic safety, one starts by describing the theory in terms of the quantum fields and their symmetries. The fields contain the degrees of freedom. From these fields there are multiple ways in which one can construct actions. The space consisting of all possible action functionals that can be constructed from the given field content and symmetries is called *theory space*. The strengths of different field combinations contributing to an action is determined by *couplings*, usually denoted $\{g_\alpha\}$. Since these couplings fully specify the content of a theory, they can be viewed as coordinates on theory space. There are an infinite number of couplings, since in principle a theory could contain an infinite number of independent interaction terms. Figure 1.1 illustrates the concept of theory space, including key elements that will be introduced in the rest of this section.

Solving the theory would be accomplished by knowing the effective action Γ, which gives a description of the quantum system taking all quantum fluctuations into account. The Wilsonian RG generalizes this effective action Γ by giving a family of effective actions $\{\Gamma_k\}$, depending on the momentum scale parameter k. The effective average action (EAA) Γ_k includes only fluctuations with momentum $p^2 > k^2$ in its effective vertices. The Wetterich equation, encodes the change of Γ_k when the coarse-graining scale k is lowered by integrating out additional fluctuations. By continued lowering of the coarse-graining scale k, the EAA flows through theory space towards different values of k. Practically, this means specifying the scale dependence of the couplings $\{g_\alpha(k)\}$. This gives rise to a trajectory for the EAA to follow through theory space by varying k. This trajectory is called the *RG trajectory* following the *RG flow*. Observable physics appears at the $k = 0$ endpoint of the RG trajectory, recovering the effective action $\Gamma = \lim_{k\to 0} \Gamma_k$. This trajectory with its endpoints is what we are interested in constructing in this thesis.

A consistent QFT corresponds to an EAA that is well-behaved for all values of k. In particular, the limits $k \to 0$ and $k \to \infty$ should exist. As described before, asymptotic safety sets a condition for the latter limit: dimensionless couplings stay fixed in this limit. This requires the existence of a UV fixed point $\{g^*_\alpha\}$. This is defined to be the point where the beta functions $\beta_n(g_\alpha)$, defining the flow field in theory space, are zero: $\beta_n(g^*_\alpha) = 0; \forall n$. Around the fixed point, lowering k may align the vector field generated by the beta functions to point to lower values of k, away from the fixed point; or the vector field may align to point to higher values of k, into the fixed point. The directions that point into the fixed point are called *UV-attractive* or *relevant*, while those that point away from the fixed point are called *UV-repulsive* or *irrelevant*. Relevant directions drag the flow into the fixed point as $k \to \infty$. There is also the possibility that a fixed point has both relevant and irrelevant directions. In this case we call it a saddle point. For a fixed point to be considered physical, it must have a finite number of relevant directions, which means that the fixed point controls the high-energy behavior, ensuring that couplings remain finite and well-defined in the UV. In addition, the relevant directions must respect fundamental symmetries of the theory (e.g., diffeomorphism invariance in quantum gravity).

Following a flow emanating from a fixed point to lower k-values gives the set of all points that are pulled into the fixed point as $k \to \infty$. This set is called the *UV-critical surface*, S_{UV}. All theories on S_{UV} are safe from divergences. The dimensionality of S_{UV} is equal to the number of relevant

couplings. This means that asymptotically safe theory is more predictive when the dimensionality of S_{UV} is smaller.

Two distinct types of fixed points can be identified. A fixed point for which all interactions vanish always exists. This corresponds to a free theory, and is called a *Gaussian* fixed point (GFP). A fixed point for which at least one relevant coupling is non-zero is called a *non-Gaussian* fixed point (NGFP). Standard perturbation theory is valid around the GFP only. In this sense GFPs are associated with theories that are perturbatively renormalizable and asymptotically free. It should come as no surprise that gravity does not go to a GFP in the UV. The asymptotic safety program then stipulates that is goes to a NGFP, such that the UV behavior may not be treated perturbatively [12].

1.2 Investigating asymptotic safety

For a new quantum field theory that includes gravity one would usually proceed as follows. Starting from a given field content and a set of symmetries one constructs the EAA Γ_k, which depends on the scale parameter k. The field content consist of fluctuating fields Φ and background fields $\bar{\Phi}$. The EAA satisfies the Wetterich equation

$$k\partial_k\Gamma_k[\Phi,\bar{\Phi}] = \frac{1}{2}\mathrm{STr}\Big[(\Gamma_k^{(2)}[\Phi,\bar{\Phi}] + \mathcal{R}_k[\bar{\Phi}])^{-1}k\partial_k\mathcal{R}_k[\bar{\Phi}]\Big], \tag{1.1}$$

where $\Gamma_k^{(2)}$ is the second functional derivative of Γ_k with respect to Φ, $\mathcal{R}_k$ is the regulator, and STr is the supertrace, the analogue of the trace in the theory of super linear algebra. The exact workings of the supertrace are unimportant for the current review, but can be found in detail in [13]. The regulator provides a k-dependent mass term for fluctuations with momenta $p^2 \ll k^2$ and vanishes for momenta $p^2 \gg k^2$. The Wetterich equation is an exact equation. The solutions Γ_k of the Wetterich equation give the bare action for $k \to \infty$ and the effective action for $k \to 0$. In case of an asymptotically safe theory the bare action is found as the fixed functional Γ_*.

Next, it is possible to choose a set of basis functionals $\{\mathcal{O}_n[\Phi,\bar{\Phi}]\}$ that span theory space. These are all the operators that can be constructed from the field content. The EAA then admits an expansion of the form

$$\Gamma_k[\Phi,\bar{\Phi}] = \sum_n \bar{g}_n(k)\mathcal{O}_n[\Phi,\bar{\Phi}]. \tag{1.2}$$

The couplings $\{\bar{g}_n(k)\}$ in equation (1.2) are dimensionful. These are traded for dimensionless couplings

$$g_n(k) \equiv \bar{g}_n(k)k^{-d_n}, \tag{1.3}$$

where d_n denotes the mass-dimension of $\bar{g}_n(k)$. The expansion (1.2) is then inserted into the Wetterich equation. The supertrace is then expanded to give the beta functions

$$k\partial_k g_n(k) = \beta_n(g_1, g_2, \dots). \tag{1.4}$$

Fixed points are found as sets of coupling constants $\{g_n^*\}$ for which $\beta_n(g_1^*, g_2^*, \dots) = 0$, for all $n = 1, 2, \dots$. Together with physical initial conditions the beta functions determine the entirety of Γ_k for all k.

From the beta functions we can directly infer some information. We can find the attractive and repulsive directions at the fixed point by linearizing the beta functions around the fixed point. The linearized beta functions are

$$k\partial_k g_n(k) \simeq (B^T)^l_n (g_l(k) - g^*_l), \quad B^n_l \equiv \left.\frac{\partial \beta^n}{\partial g^l}\right|_{g=g_*}, \tag{1.5}$$

where B^T denotes the matrix transpose of B. Equation (1.5) is solved in terms of stability coefficients $\theta_{(n)}$ and right-eigenvectors $V^{(n)}_n$, satisfying the eigenvalue equation

$$(B^T)^l_n V^{(n)}_l = -\theta_{(n)} V^{(n)}_n, \tag{1.6}$$

where (n) enumerates the eigenvalues. We then separate the stability coefficients into two groups $\{\theta_{(n)}\} \mapsto \{\theta_{(m)}, \tilde{\theta}_{(\mu)}\}$, such that $\mathrm{Re}(\theta_{(m)}) > 0$ and $\mathrm{Re}(\tilde{\theta}_{(\mu)}) < 0$. The eigendirections $V^n_{(m)}$ ($\tilde{V}^n_{(\mu)}$) are attractive (repulsive) as $k \to \infty$.

Since there are in principle an infinite number of couplings, the Wetterich equation gives rise to a system of infinitely many coupled differential equations. This system is very hard to solve exactly. In almost all cases one has to resort to restricting the theory space, and thus the number of couplings and differential equations, to a finite subset. This *truncation of the theory space* approximates all but a finite number of coupling constants by zero. This reduces the exact expression in equation (1.2) to the ansatz

$$\Gamma_k[\Phi, \bar{\Phi}] \approx \sum_{n=1}^{N} \bar{g}_n(k) \mathcal{O}_n[\Phi, \bar{\Phi}]. \tag{1.7}$$

How to choose a good truncation is an active area of research and should take into account multiple factors to have the ansatz capture the relevant physics. The final system can now be solved analytically or numerically.

1.3 Algorithmic treatments

The solving of the system of differential equations is a boundary value problem. The boundary conditions specified are the NGFP, the linear behavior of S_{UV} around the NGFP, and the experimental values for various physical quantities at $k = 0$.

Historically, this boundary value problem is solved by transforming it to an initial value problem. The initial conditions are provided by the NGFP and the linear behavior of the S_{UV} surface around the NGFP. One can then apply the *shooting method* to scan the space of effective actions appearing at $k = 0$ and compare these to physics predictions. The shooting method chooses a random direction at which to leave the NGFP and integrates the RG flow in the direction of decreasing k. The guessed direction then has to be refined until the trajectory ends up in a physically interesting region. A detailed account of this process for four-dimensional quantum Einstein gravity can be found in [14].

While historically the shooting method has been the tool of choice, this does not mean that this is the only approach. One could take a step back and instead solve the boundary value problem directly. The difference here is that solving the boundary value problem will give a parametrization of S_{UV}, as opposed to a set of integral curves within S_{UV}. This approach of solving the boundary

value problem directly is proposed in [15] and has been applied in [1] to a scalar-tensor theory in cosmology. Solving the boundary value problem is cumbersome and numerically expensive, especially as the number of couplings retained in equation (1.7) becomes large. This is the prime motivation for developing new computational methods that treat this problem. This will be done in this thesis.

With two possible methods to tackle the problem of finding the $k = 0$ surface embedded in S_{UV}, the natural question to ask is: which one should be used. In a low-dimensional approximation of theory space of a scalar field theory in $d = 3$ Euclidean dimensions, it has been shown that the boundary value approach produces the same numerical accuracy as the shooting method [15]. The problem with the shooting method surfaces when the dimension of S_{UV} and/or the dimension of the truncation grows larger. A direct example would be a theory of asymptotically safe quantum gravity supplemented by the matter degrees of freedom of the standard model, where the dimension of S_{UV} is expected to be of order twenty [16]. This surface should then be embedded in an even larger space also comprising couplings associated with beyond the standard model physics to be able to make predictions. In this case, using the shooting method will be near impossible, as will be shown in the complexity analysis in section 5.3. The new approach, developed in this thesis, should perform much better in these high-dimensional cases. This expectation is based on similar high-dimensional boundary value problems that have been solved in [17, 18].

The purpose of this thesis is the implementation of the boundary value approach proposed in [15] by using one of the most basic, but elegant, algorithms: the *genetic algorithm (GA)*. This algorithm is known to be a good first shot at solving any optimization problem, which the described problem will turn out to be. It is chosen in part since it is the minimal improvement over random search. However, the important reason is that the GA is suitable for a detailed analysis of the problem being solved, by considering how the algorithm performs when changing *control parameters*, i.e. the parameters used to tweak the functioning of the algorithm itself.

The rest of the thesis is organized as follows. The setup required to turn the boundary value problem into an optimization problem will be presented in chapter 2. The GA in full will be presented in chapter 3. A guiding set of good practices, followed within the computer science community, is reviewed in chapter 4. The results of the research are presented and discussed in chapter 5. The final conclusions as well as a broad outlook are presented in chapter 6.

Chapter 2

Methodology

In this chapter we will explain the methodology used to treat the boundary value problem mathematically. We introduce it in its general form in section 2.1. Then in section 2.2 we use it to describe the example model that will be used throughout the thesis.

2.1 The boundary value method

We shall continue where we left off at the end of section 1.2. At this point we used the Wetterich equation to compute the beta functions, found a NGFP, and computed its the relevant and irrelevant eigendirections by linearizing the beta functions. Figure 2.1 gives an illustrative depiction of the methodology used to construct S_{UV} that will be explained in this section.

We now treat the couplings in much the same way as the eigendirections, splitting them into two groups. This time the determining criteria is the mass dimension of the coupling. The mass dimension of each coupling can be determined from the EAA Γ_k. The couplings are split as

$$g^n(k) \mapsto \{u^m(k), v^\mu(k)\}, \tag{2.1}$$

where $\{u^m(k)\}$ are those with non-zero mass dimension, and $\{v^\mu(k)\}$ are those with zero mass dimension. Couplings $\{u^m(k)\}$ are called relevant, while $\{v^\mu(k)\}$ are irrelevant. We define all couplings $\{g^m(k)\}$ with positive mass dimension into ones with negative mass dimension by defining new couplings as $u^m(k) = 1/g^m(k)$. The reason we want couplings with negative mass dimension, is because this allows us to associate the $k = 0$ endpoint with the condition that all relevant couplings are simultaneously zero. The redefinition of couplings with positive mass dimension then means that relevant couplings are those that are negative while irrelevant couplings are those that are zero. Indeed, since we may write the dimensionful coupling $\bar{u}^m(k)$ as $\bar{u}^m(k) = u^m(k)k^d$ with $d < 0$, and since $\bar{u}^m(k)$ must be finite to correspond to an endpoint in the classical regime, we see that for $k \to 0$ also $u^m(k) \to 0$. The physical predictions for the couplings $\{v^\mu(k)\}$ are then read off at $\{u_0^m\} \equiv \{u^m(k = 0)\}$.

The boundary value problem is then treated as follows. We are given a truncation of theory space, that can be identified with $\mathbb{R}^N$. On this space we construct a vector field, the beta functions, that determine the flow of a point through this space. We are given a single point in theory space,

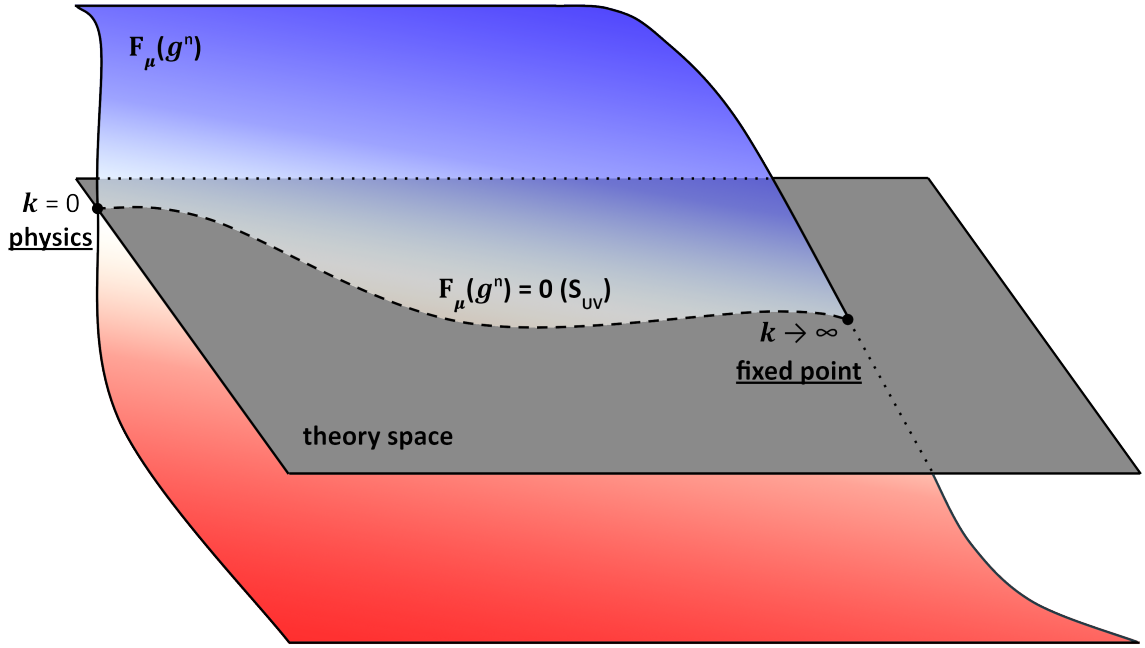


Figure 2.1 An illustrative depiction of theory space, how $F_\mu(g^n) = 0$ gives S_{UV}, and other key elements. The surface S_{UV} follows the RG flow of the beta functions through the truncated theory space. The generating function F_μ is invariant under this flow. At the $k = 0$ limit lies the physical endpoint. At the $k \to \infty$ lies the NGFP. The surface S_{UV} contains all points that flow into the fixed point in the high energy limit, and is given by the subspace $F_\mu(g^n) = 0$.

the fixed point, and are asked to construct the surface in $\mathbb{R}^N$ that is tangent to the vector field and includes the given point. By enforcing the relevant eigendirections as boundary condition around the fixed point, we recover S_{UV}. The idea is to use the u^m as coordinates on S_{UV}. The embedding of S_{UV} into $\mathbb{R}^N$ is then given implicitly by a set of $N - M$ generating functions

$$F_\mu(u^m, v^\mu) = 0. \tag{2.2}$$

The generating functions are combinations of couplings whose values are conserved along the RG flow. In particular, this means that the equation

$$k\partial_k F_\mu(g^n) = 0, \tag{2.3}$$

is satisfied. This condition ensures that the surface defined by $F_\mu(u^m, v^\mu) = 0$ is invariant under the RG flow, which is a necessary property for the UV-critical surface.

In the vicinity of the NGFP, the generating functions F_μ can be found to linear order. The eigenvectors $\{V^n_{(m)}, \tilde{V}^n_{(\mu)}\}$ span the tangent space at the fixed point. The vectors $V^n_{(m)}$ are tangent to S_{UV}, but, in general, the vectors $\tilde{V}^n_{(\mu)}$ are not normal to the surface. Following the Gram-Schmidt process we project the $V^n_{(m)}$ into an orthogonal basis $t^l_{(m)}$. We can then construct a set of normal vectors as

$$n^l_{(\mu)} = \tilde{V}^l_{(\mu)} - \sum_{(m)} t^l_{(m)} \frac{\left\langle t_{(m)} \middle| \tilde{V}_{(\mu)} \right\rangle}{\left\langle t_{(m)} \middle| t_{(m)} \right\rangle}, \tag{2.4}$$

where $\langle x|y\rangle$ is the standard inner product in $\mathbb{R}^N$. At the linearized level around the fixed point, the condition $F_\mu(g^n) = 0$ is given by

$$F^*_\mu(g^l) = \sum_n n^l_{(\mu)}(g^l - g^l_*). \tag{2.5}$$

The final step that allows us to construct F_μ, and thus S_{UV}, comes in the form of the *master equation*. The master equation is derived as follows. We start with the condition that F_μ is conserved under RG flow, equation (2.3). The chain rule then gives

$$0 \stackrel{!}{=} k\frac{\partial F_\mu}{\partial k} = \sum_n \frac{\partial F_\mu}{\partial g^n} k\partial_k g^n. \tag{2.6}$$

Using the definition of the beta functions, equation (1.4), reduces the expression further to

$$\sum_n \frac{\partial F_\mu}{\partial g^n}\beta^n \stackrel{!}{=} 0. \tag{2.7}$$

Splitting $g^n \mapsto \{u^m, v^\mu\}$ finally gives the master equation as

$$\sum_m \frac{\partial F_\mu}{\partial u^m}\beta^m + \sum_\nu \frac{\partial F_\mu}{\partial v^\nu}\beta^\nu = 0. \tag{2.8}$$

The assumption is made that $\det\left(\frac{\partial F_\mu}{\partial v^\nu}\right) \neq 0$, so that the implicit function theorem guarantees that the system $F_\mu(g^n) = 0$ has a local solution in the form

$$v^\mu\big|_{S_{UV}} = v^\mu(u^m), \tag{2.9}$$

that is, a relation for the couplings v^μ in terms of the coordinates u^m parameterizing the surface S_{UV}.

2.2 The example model

The example model is given by a scalar field theory in two-dimensional Euclidean space: an Ising model. The RG flow admits a NGFP. Following the derivation [6], the EAA is given by the local potential approximation

$$\Gamma_k[\phi] \simeq \int d^2x \left[\frac{1}{2}(\partial\phi)^2 + U_k[\phi]\right]. \tag{2.10}$$

Here $\phi(x)$ is a real scalar field with mass-dimension zero and $U_k[\phi]$ is the interaction potential, which is even in ϕ by assumption. The potential is truncated at 4th order in the field

$$U_k[\phi] = \frac{1}{24\bar{u}_k}\phi^4 + \frac{\bar{v}_k}{2\bar{u}_k}\phi^2. \tag{2.11}$$

This gives a two-dimensional truncation of theory space. Here, the couplings follow the convention introduced in equation (2.1). The couplings are then made dimensionless. The coupling u_k will be the free parameter and v_k the predicted one.

In particular, this parametrization of S_{UV} in terms of u_k and v_k ensures that $u_0 = 0$ allows one to

extract v_0 from S_{UV}. The value of v_0 can be related to the full effective action of the theory, since $u = 0$ ensures $k = 0$. This is because the mass dimension of u_k is negative. To see this, we analyze equation (2.10). We know that the action must have mass dimension zero, which is also true for the effective average action Γ_k. We know that $d^d x$ has mass dimension $-d$, so in our case -2. To balance this, both $(\partial_k \phi)^2$ and $U_k[\phi]$ must have mass dimension 2. We then analyze equation (2.11). The field ϕ has mass dimension zero, so u_k and v_k must have mass dimension -2 and zero respectively. Since $\bar{u}_k$ has negative mass dimension, we can write the dimensionful $\bar{u}_k$ in terms of the dimensionless u_k as $\bar{u}_k = u/k^2$. Since $\bar{u}_k$ must remain finite as $k \to 0$ it is required that $u \to 0$. Thus $u = 0$ gives us the $k = 0$ result.

The beta functions are derived in [19]. They are

$$\beta_u = 2u - \frac{3u^3}{2\pi(v+u)^3}, \quad \beta_v = -\frac{u^2(7v+u)}{4\pi(v+u)^3}. \tag{2.12}$$

The NGFP is located at

$$(u_*, v_*) = \left(\frac{343}{288\pi}, -\frac{49}{288\pi}\right), \tag{2.13}$$

the stability coefficients are

$$\theta_{(1)} = \frac{1}{3}(-1+\sqrt{43}), \quad \tilde{\theta}_{(1)} = -\frac{1}{3}(1+\sqrt{43}), \tag{2.14}$$

and the right-eigenvectors are

$$V^n_{(1)} = \left(-8+\sqrt{43}, 1\right)^T, \quad \tilde{V}^n_{(1)} = \left(-8-\sqrt{43}, 1\right)^T. \tag{2.15}$$

Thus S_{UV} is one-dimensional. We are dealing with a saddle point with one UV-attractive and one UV-repulsive eigendirection. This is why one of the couplings will be predicted and the other is free. This deviated from the standard approach used in condensed matter and the study of critical phenomena, where the fixed point is used as IR-completion instead of the UV-completion that we consider.

Using the Gram-Schmidt process we construct the normals to S_{UV} at the fixed point and obtain the generating function $F(u, v)$ to linear order around the fixed point, in accordance to equations (2.4) and (2.5).

$$F^*(u,v) = -\frac{(27+4\sqrt{43})\sqrt{27-4\sqrt{43}}}{82}u - \frac{(44+5\sqrt{43})\sqrt{27-4\sqrt{43}}}{82}v + \frac{49}{288\pi}\sqrt{\frac{422+61\sqrt{43}}{82}}. \tag{2.16}$$

We omit the index μ since S_{UV} is one-dimensional. Normalization is chosen such that $||\nabla F^*|| = 1$.

It is the generating function that is the final output of the algorithm. By setting $F(u_k, v_k) = 0$ one obtains an implicit expression for S_{UV}. Then from $F(u_0, v_0) = F(0, v_0) = 0$, one extracts the prediction for v_0.

Chapter 3

The algorithm

In this chapter we will detail the algorithm used to find the surface $F_\mu(u, v) = 0$. In addition to the general form, we will explain the algorithm as applied to the example introduced in chapter 2. We stress that the algorithm is a general way to construct generating functions $F_\mu(g_k^n)$ that satisfy the master equation (2.8), subject to the linear behavior around the fixed point. We work through the setup phase of the algorithm in sections 3.1, 3.2, 3.3, and 3.4. Then, we discuss the genetic algorithm itself in section 3.5.

3.1 Initialization and input

We start off by specifying the initialization of the algorithm. The algorithm will take as input:

- The dimension N of the theory space and the dimension M of S_{UV}.
- The beta functions β^n.
- The fixed point $g_*^n \mapsto \{u_*^m, v_*^\mu\}$.
- The generating function F_μ^* to linear order around the fixed point.

In addition, we specify the control parameters for later parts of the algorithm. These will be elaborated on in section 3.5. For the scalar field in two dimensions we will have $N = 2$ and $M = 1$. The rest of the input can be found in the section 2.2.

3.2 Functions

There are few functions that are used in the algorithm. We have already introduced the beta functions in equation (2.12). Both beta functions for u and v will for the purpose of efficient computation be packed into a single functional N-dimensional (0,1)-tensor β^n.

We expand the generating function into a set of N_p basis functions $\{\psi^i\}$ as

$$F_\mu(g^n) = \sum_{i=1}^{N_p} p_{i\mu}\psi^i(g^n), \tag{3.1}$$

where $p_{i\mu}$ are the $N_p \times (N - M)$ expansion parameters that the algorithm seeks to find. The basis functions may depend on the specific problem that is treated, but in general care should be take to make sure that the basis functions are able to convey the desired amount of detail. For the example model we decided to choose Multivariate Cauchy Distributions (MCDs) as basis functions. These take the form

$$\psi^i : \mathbb{R}^N \mapsto \mathbb{R}, \quad g^n \mapsto \left(1 + \sum_n^N \frac{(g^n - g^{ni}_{col})^2}{\sigma^2}\right)^{-1}, \tag{3.2}$$

where σ is the scale parameter, akin to the median absolute deviation of the Univariate Cauchy Distribution, and $\{g^{ni}_{col}\}$ are collocation points that are discussed in section 3.3. The basis functions admit partial derivatives of the form

$$\partial_l \psi^i : \mathbb{R}^N \mapsto \mathbb{R}, \quad g^n \mapsto -\frac{2}{\sigma^2}(g^l - g^{li}_{col})\psi^i(g^n)^2. \tag{3.3}$$

The algorithm stores the basis functions as a functional N_p-dimensional (0,1)-tensor ψ^i. The partial derivatives are stored as a functional $(N \times N_p)$-dimensional (1,1)-tensor $(\partial\psi)^i_n$.

To allow efficient calculation later on in the algorithm we pre-calculate the values of β^n, F^*_μ, $\partial_l F^*_\mu$, ψ^i, and $\partial_l \psi^i$ for each collocation point g^{nj}_{col}. This results in the tensors $\beta^{nj} \equiv \beta^n(g^{nj}_{col})$, $(F^*)^j_\mu \equiv F^*_\mu(g^{nj}_{col})$, $(\partial F^*)^j_{\mu l} \equiv \partial_l F^*_\mu(g^{nj}_{col})$, $\psi^{ij} \equiv \psi^i(g^{nj}_{col})$, and $(\partial\psi)^{ij}_l \equiv \partial_l\psi^i(g^{nj}_{col})$. These tensors are now not functional anymore and do not have to be recomputed. This results in a major increase in computational efficiency.

Finally, we define the master equation as a functional N_p-dimensional (1,0)-tensor of the parameters $p_{i\mu}$, based on equation (2.8). In terms of the pre-calculated tensors, it is defined as

$$\mathcal{M}^j_\mu(p_{i\mu}) = p_{i\mu}(\partial\psi)^{ij}_n \beta^{nj} \equiv (p\partial\psi\beta)^j_\mu \equiv \mathcal{M}^j_\mu, \tag{3.4}$$

where the latter two forms are used when it is implied that the master equation has been evaluated for a specific choice of $p_{i\mu}$. These forms allow us to conveniently write that an optimal solution $p^\star_{i\mu}$ of the problem is one for which

$$\overline{\mathcal{M}} \equiv \sum_{j=1}^{N_p} \left| \sum_{\mu=1}^{N-M} \mathcal{M}^j_\mu \right| \stackrel{!}{=} 0. \tag{3.5}$$

The absolute value is introduced to turn the problem from finding the zeroes into a minimization problem. Given that most quantities in equation (3.4) and (3.5) are pre-calculated and related by tensor algebra, it is possible to efficiently compute $\overline{\mathcal{M}}$ for any given set $p_{i\mu}$. This is of course deliberate as $\overline{\mathcal{M}}(p_{i\mu})$ will be a natural choice for a fitness function. This will become apparent in section 3.5.

3.3 Collocation points grid

To obtain F, we start by constructing a grid of N_p collocation points $\{g^{ni}_{col}\}$, one per basis function. There is some freedom in how to construct the grid. In general one would want to make sure the dense parts of the grid are placed over parts of the truncated theory space where detail is desired, such as the $k = 0$ subspace or the fixed point. In addition, the grid is constructed such that one collocation point is exactly at g^n_* and a line of collocation points falls close to but not over the

subspace $u_k = 0$, corresponding to $k = 0$. We avoid points with $u = 0$ or $u = -v$ since these result in zeros and singularities in the beta functions, at least, for the example theory we treat.

We choose to divide the grid into four quadrants, each of which can be given a different resolution and thus density. The grid used for the example is shown in figure 3.1. We approximate F_μ with basis functions ψ^i. One basis function is placed centered at each grid point. One may chose to use more basis functions, as proposed in [15], which makes the final system of equations overdetermined. One should never choose less than N_p basis functions as this makes the system underdetermined. The collocation points are stored as an $(N_p \times N)$-dimensional (0,2)-tensor g^{ni}_{col}.

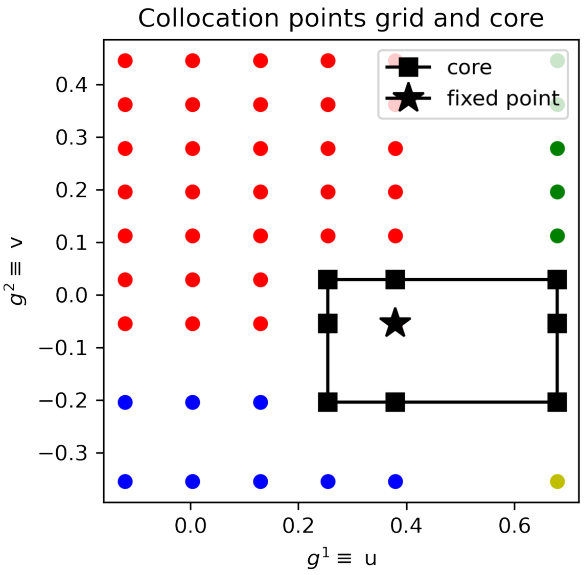


Figure 3.1 The grid used for the example model. The four colors show the quadrants whose resolution can be individually adjusted. The black border shows the core around the NGFP.

3.4 Fixing parameters

Since we know the linearization of the generating function around the fixed point we are in principle able to fix the parameters for K out of the N_p basis functions to force them to reproduce the linear behavior of F_μ close to the fixed point. This is the way in which we enforce the boundary conditions. A natural choice for the K basis functions are those associated with the collocation points in a direct shell around the fixed point, since further away from the fixed point the linearized generating function will become increasingly inaccurate. We call this subset of K collocation points the "core", see figure 3.1, and denote a point in the core as $g^{nj}_{\sim}$.

In order to fix parameters for all K points in the core we use that

$$F_\mu(g^{nj}_{\sim}) = p_{i\mu}\psi^i(g^{nj}_{\sim}) \simeq F^*_\mu(g^{nj}_{\sim}). \tag{3.6}$$

The parameters for any point $g^{nj}_{\sim}$ in the core are then taken as

$$p_{j\mu} \simeq \frac{1}{\psi^j(g^{nj}_{\sim})}\left[F^*_\mu(g^{nj}_{\sim}) - \sum_{i\neq j} p_{i\mu}\psi^i(g^{nj}_{\sim})\right]. \tag{3.7}$$

This fixes a total of $K \times (N - M)$ parameters.

This procedure eliminates the trivial solution $p_{i\mu} = 0$. This solution is undesirable, since it corresponds to $F_\mu(g^n) = 0$ for all g^n. This solution is uninteresting, since it encodes no information about S_{UV}. Aside form this, starting by fixing parameters gives a good initial guess towards the optimal solution.

3.5 The genetic algorithm

Given the steps taken in the previous sections, only one final step remains: optimizing the expansion parameters. In principle, any optimization algorithm should do the trick and there are many to choose from. Different optimization algorithms serve different purposes and are tailored to different problems. We choose to implement one of the classic optimization algorithms that is relatively simple to apply and can tackle a wide variety of problems: the *genetic algorithm (GA)*. In the outlook, section 6.2, we will present other options for optimization algorithms.

The core idea behind the GA is to leverage techniques from the oldest optimization that we know of in nature: evolution. Natural evolution is in the essence an optimization method meant to optimize a species survivability [20]. This functions through a structure of genes and chromosomes that encode heritable characteristics of a species which change with successive generations [21]. This change comes about through several mechanisms, the most concrete being: natural selection, genetic mutation, and genetic crossover. Originally proposed by Alan Turing [22], the idea of an algorithm to mimic the mechanisms of natural evolution has been around since 1950. Since then, the idea has developed into an entire class of algorithms termed *evolutionary algorithms*. The GA has become arguably the most famous of all, as it gained traction through the work of John Holland in the early 1970s [23]. In particular, Holland introduced a formal framework to predict the quality of subsequent generations of the "species" modeled by the algorithm, known as *Holland's Schema Theorem* [24]. The strength of the GA lies in the relative simplicity of implementation combined with a wide diversity in applications. As such it is often one of the first approaches used to tackle an optimization problem.

A traditional GA consists of five fundamental parts:

- **Gene:** The fundamental quanta that can be changed in order to optimize. This can be a single bit, or a real number, or any other dial that can be turned and that can not be decomposed into smaller components. In our case this is a single parameter $p \in \mathbb{R}$ that exist in the parameter set $p_{j\mu}$.
- **Individual/Chromosome:** A set of genes that constitutes a solution to the optimization problem. The end goal for the algorithm is to find the optimal individual, that is, the optimal solution to the optimization problem. An individual or chromosome is often directly called a solution, even though the exact language would be that an individual or chromosome gives a solution when substituted back into the problem. In our case an individual or chromosome is a specific choice for all parameters $p_{i\mu}$. The associated solution is the generating function, but we refer to both $p_{i\mu}$ and $F_\mu(g^n)$ with "the solution" interchangeably.
- **Population/Generation:** A collection of different individuals/chromosomes. At the start of the algorithm the population has a high degree of randomness as to cover a wide spread of different solutions. As the algorithm optimizes, the randomness in the population shrinks as it converges to the optimal solution. Analogous to this view, the dynamics of an algorithm is sometimes described in terms of change in entropy, where producing a rapid decrease in entropy is a condition for algorithmic convergence [25]. This type of entropy is called *Shannon entropy*, after Claude Shannon [26]. We will not use this terminology, but one might find it

helpful to keep this picture in mind.

- **Fitness/Scoring function:** A way to score each individual in the population, rating how close to optimal the individual is. In our case this is the measure $\overline{\mathcal{M}}$ derived from the master equation. Our score thus has to be as close to zero as possible. This means the problem is a minimization problem and lower scores are better.
- **Optimization cycle:** The core of the algorithm. Iteratively, the algorithm will change and adjust the population to make it converge to the optimal solution. The workings of this cycle will be treated in more detail in section 3.5.2.

The fundamental idea behind Holland's Schema Theorem, later generalized as the *Price equation* [27], is that we can divide a chromosome into blocks of genes. Different blocks of genes are thought to have different traits or alleles, when thinking of the connection to natural evolution. These alleles in principle have their own fitness, representing different aspects of the optimization problem being solved. When thinking of alleles as blocks of genes in this way, we are essentially considering patterns of genes. Specific patterns of genes correspond to different alleles being present in a chromosome and each allele present adds a certain trait to the solution represented by this chromosome. These traits can be rather abstract or very concrete. For example, in the case of constructing a surface, an allele could represent a zero or pole at a specific position. If the optimal solution and those solutions that are near-optimal always have a zero or pole in these positions, we expect that this allele should always show up, it is a necessary pattern. These patterns of genes are formally called schema. Through considering these schema as cylindrical sets and thus as a topological space, it is possible to show that the frequency of alleles with good fitness scores appearing in a population will increase exponentially during training [24].

3.5.1 Initialization

The GA starts off with an initialization step. In this step we construct the initial population. In the pure implementation of the GA this initial population would be entirely random, but most modern GA's use information about the problem to construct a more educated initial population that is expected to be closer to the optimal solution. However, one should always be careful in steering the initial solution too much in a certain direction, as to avoid the algorithm under-exploring other parts of the solution space that might contain unexpected optima.

In our case, we decide to have the initial population adhere to the boundary conditions at the fixed point. We do this by fixing the parameters as described in section 3.4. This choice is very safe, as any optimal solution should also respect these boundary conditions. The control parameter that can be adjusted to optimize the algorithm's performance here is of course the method of initialization. In our case this control parameter has two options: completely random or random with boundary conditions.

After creating the initial population, the fitness scores are calculated for this initial population. After this, we initialize lists in which data can be collected while the algorithm is running. These comprise metrics such as runtime, scores, and/or intermediate optimal solutions. We are then ready to start the iterative optimization cycle.

3.5.2 The optimization cycle

The GA will iterate the following steps:

1. **Selection:** Given a population we decide which individuals make it to the next generation and which do not. This can be done in many different ways. In almost all cases the result is that we keep some part of individuals, remove another part and we might have multiple copies of the same individual based on their fitness score. A detailed overview of the most common selection methods can be found in [28].

   The method we employ is the following. We have as control parameter a *keep percentage* x. We sort the population and only keep the top x percent. From this top x percent we select individuals at random weighted by their score such that those with better scores get selected more frequently. We keep selecting individuals until we have filled the new population. In the literature this selection method is called *roulette selection*.

2. **Mutation:** Now that we have sampled a new population we want to introduce some new variety to explore the solution space further. This is done by changing the value of some genes for each individual in the population. There is a vast collection of methods for this process we call mutation, an overview of the most common can be found in [29]. The method that is used will depend on what the genes represent and the specific problem being tackled. For example, if the gene is a bit, a mutation would be flipping the bit, but if the gene is a real number, a mutation would be adding or subtracting some random amount. The latter is the case we are dealing with, so this is what we will consider for the rest of the text.

   The method that we use was derived from [30]. There are two control parameters: *mutation percentage* x and *mutation amount* y. We consider all genes among all individuals in the population and randomly let x percent get mutated. For those x percent of genes we add a (possibly negative) amount according to a Gaussian distribution with as standard deviation the fitness score of the individual that the gene belongs to, multiplied by y, and with mean zero. This mutation method is called *Gaussian mutation*.

3. **Shuffle:** We now mix the population to redistribute information evenly for the crossover process.

4. **Crossover:** Where mutation changes individual genes, we will now consider the mixing of blocks of genes. This is motivated by the theory of alleles underlying Holland's Schema Theorem. Crossover seeks to mix entire alleles between individuals in the populations. Once again, there is a wide variety of methods to perform crossover. A broad overview of crossover methods can be found in [31].

   The method that we used is *one-point crossover*. The control parameter is the *crossover percentage* x. We divide the population into pairs of two individuals. For x percent of these pairs we pick a random point along the individual's genes. We then switch the genes before this point between the individuals in the pair. This way we effectively swapped some alleles between the individuals, promoting genetic diversity in the population.

5. **Fitness:** Finally, we reevaluate the fitness score for the population.

With any iterative approach there is the problem of when to halt the iterative cycle. There is no concise solution to this problem. Practically, one should always introduce a maximum number of generations, but additional stopping criteria may be used. These criteria are very problem specific. In our case one could consider setting a stopping criterion when the fitness score is within a set distance from zero.

Chapter 4

Best practices within computer science research

As theoretical physics research in general, and asymptotic safety in particular, gradually move towards utilizing supplementary algorithms in the research process, it is important to look critically at the procedure with which these algorithms are developed, expanded, and tested. A clear direction to follow in this regard are the practices that exist within algorithmic computer science research. The most relevant of these practices will be outlined in this chapter. First, in section 4.1, we will discuss a fundamental concept in theoretical computer science, *computational complexity*. Then, in section 4.2, we will consider how one might optimize an algorithm itself to solve a problem as effectively as possible given some criteria for "effectiveness". The pseudocode for our implementation of the procedures introduced in section 4.2 can be found in appendix A.

4.1 Computational complexity

In practice, the performance of any algorithm depends on the hardware it is executed on. This poses a considerable problem for comparing different algorithms developed by different research groups in an unbiased way. In particular, the properties that are most hardware dependent are computation time and memory space allocation. This splits the field of complexity theory into two major aspects: *time complexity* and *space complexity*.

4.1.1 Time complexity

In order to characterize the time complexity of an algorithm we start by defining *elementary operations* that are assumed to take constant time; computational quanta, if you will [32]. The choice of elementary operations usually depends on the *operating level* of the algorithm. An algorithm running directly on the bit-level will have a single bit-switch as its elementary operation, while many higher-level algorithms use addition, subtraction, multiplication, and division all as elementary operations.

The core idea is to determine the number of elementary operations performed by an algorithm as a function of the size of the input data. The total often depends on the content of the input, so for a detailed analysis one would calculate this total for the best case (Ω-notation), worst case

($\mathcal{O}$-notation), and average (Θ-notation) scenario. For each of these we express the complexity in *big-O notation*, also known as *Bachmann-Landau notation* or *asymptotic notation*. It describes the limiting behavior of a function when the argument tends to a certain limiting value, often infinity [33, 34]. In the case of complexity analysis the function gives the number of elementary operations performed by an algorithm, with the size of the input data as parameter. The size of the input data is often split up into multiple parameters, whose complexity can be analyzed separately or combined. An example can be found in section 5.3. In practice, we know the big-O time complexity for many sub-algorithms, such as loops and condition statements, which combine easily into the total time complexity. In the case of a less detailed analysis, one usually states only the worst case $\mathcal{O}$-complexity.

Working with these elementary operations is based on the concept of the *Turing machine*, where these elementary operations are commonly called *steps*. The Turing machine is a mathematical model developed by Alan Turing [35] for which it is proven that we can break any computable algorithm down into these steps. We analyze an algorithm as if it was performed by a Turing machine, since it is the most general mathematical model to do so. The power then comes in the form of the *Church-Turing thesis* [36]. This states that a function from $\mathbb{R}^m$ to $\mathbb{R}^n$ (an algorithm) can be calculated effectively if and only if it can be calculated effectively by a Turing machine. In particular, this can be understood as saying that for every algorithm calculated on any machine there exists an equivalence of the algorithm calculated on a Turing machine. This has the important consequence that if we analyze the complexity of an algorithm as if it were computed on a Turing machine, by considering elementary operations, then the complexity obtained is a property of the algorithm. Importantly, it is independent of the machine it is computed on.

Formally, the term “effectively”, as used in the Church-Turing thesis, means that the computation can be carried out in a finite number of steps. It is said that the computing machine *halts* on the problem. Determining whether a computing machine halts on a problem is an undecidable problem. The problem was first formulated by Turing as the *Halting Problem*, and the indeterminably was implicitly proved by him in 1936 [35]. An explicit proof was later given by Martin Davis in 1965 [37]. We will return to the Halting Problem and undecidable problems in section 4.1.3. Nowadays, the term “effectively” has a looser meaning. It is commonly used to say that a problem is computable without exceeding the resources (time, space) available.

There are two situations where calculating complexity is particularly difficult. The first situation occurs when an algorithm makes use of a great deal of randomness. Nowadays, we have found that randomness can play an important part in efficiently exploring a problem’s solution space. The second situation occurs when the machine being used offloads some of the work to another machine that uses a different set of elementary operations. In practice, this second case occurs when a high-level programming language uses functions that are pre-written in a low-level language, since lower-level languages generally perform elementary computations faster at the cost of being harder to work with. The mixing of elementary operations makes it nearly impossible to compute complexity exactly. In these difficult cases one usually assigns some predefined complexity to code blocks. These are often based on a large set of experimental results. One can then fit an appropriate big-O complexity function to the experimental results and find the approximate complexity. This works quite well since big-O notation is only concerned with the highest order complexity.

4.1.2 Space complexity

The space or memory complexity of an algorithm is a measure of how much information has to be stored in order to execute the algorithm as a function of the input size. The analysis of space complexity is often much more specific to the machine on which the algorithm is run; this makes it very difficult to find a well-defined general measure of space complexity [38]. This problem is often circumvented, as the time complexity is always an upper bound for space complexity. The reason that this is true is because we require at least one elementary operation to change a single memory space. Thus at most an algorithm takes up as many memory spaces as the amount of elementary operations performed. The reason why it is an upper bound and not an exact equivalence, is because memory can almost always be reused. This has the practical consequence that space complexity is almost never explicitly calculated or even addressed, except in the extreme cases where a lot of data is used, the available memory is extremely limited, or a specific effort is being made to be considerate with memory space.

4.1.3 Greater importance of complexity

The goal of any complexity analysis is to be able to compare algorithms on even grounds. In addition, a well-performed complexity analysis gives insight into another layer of structure within the algorithm. In this regard, it is important to go beyond just giving the final complexity and show which part of the algorithm contributes which term to the complexity. The focus of big-O notation on the fastest growing term hides underlying structural nuances.

There is however a more fundamental goal in doing a complexity analysis, which lies in the generalization of algorithms to a wide set of problems. Complexity theory defines a set of fundamental *complexity classes*. These classes group problems into sets, defined by how efficient they can be solved or how efficient a solution to the problem can be verified [39]. The complexity analysis framework provides a system to sort computational problems in terms of their complexity. This system is hierarchical in nature. For example, any problem that can be solved in polynomial time (class P) can also be solved in exponential time (class EXP), since $P \subset EXP$.

Throughout the rest of the text we will discuss complexity associated with solving problems, but the exact same notions work for checking problems. The overlap of solving and checking problems is an active area of research in which the biggest unsolved problem is the $P = NP$ millennium problem [40, 41]. This problem asks whether each problem that can be solved in polynomial time (class P), can also have its solution verified in polynomial time (class NP). An example is the problem of deciding whether an incomplete Sudoku grid can be filled in completely. When presented with a "Yes" case, it is easy to verify whether the grid is indeed legally filled in; the problem belongs to the NP class [42]. There is however no known algorithm to solve a Sudoku in polynomial time, so the problem may or may not belong to the P class (look-up tables do not count as a valid solving algorithm) [43].

It is possible to prove that certain problems have a most *fundamental complexity class*. Namely, if a problem's most fundamental class is P, there can not exist an algorithm that solves the problem in less than polynomial time [44]. A problem stated in a form such that it belongs to its fundamental

complexity class is called *complete*. Keeping good track of the complexity of algorithms solving a problem is effectively the only way one has to find the fundamental class of a problem, short of direct proof. In return, finding the fundamental class of a problem often provides invaluable insights into the mathematical structure of the problem and its solution by comparing to other problems in the same class. An example of a P-complete problem is the *Circuit Valve Problem (CVP)* [45]. When given a Boolean (logic) circuit and its inputs we are asked to determine the output. There are many P-class problems that can be reduced to solving CVP. For example, deciding whether a linear system $Ax \leq b$ has a solution (Linear Programming Feasibility) is in P (Khachiyan's ellipsoid method) [46]. It can be reduced to CVP by encoding the constraints as a Boolean circuit [47]. Another example is the problem of deciding whether the maximum flow in a flow network exceeds some set value. This problem is in P (Ford-Fulkerson method) [48], but can be reduced to CVP by simulating the network as a circuit [49]. A final example is the problem that asks to decide whether a given string can be generated by a given *context-free grammar*, which is a mathematical structure behind language encoding. The dynamic parsing algorithm to solve this problem is in P [50], but can be encoded as a circuit, making it equivalent to CVP [51].

Next we introduce the concept of NP-hard problems. An NP-hard problem is one that is at least as hard to solve as the hardest problems to solve in the NP-class. Note that these need not be as hard to verify as the hardest problems to verify in the NP-class. As a matter of fact, an NP-hard problem need not be in NP at all. The importance in defining NP-hard problems as a complexity class, is that if you can solve an NP-hard problem in polynomial time, then you can solve any problem in NP in polynomial time. This final statement implies $NP \subset P$, covering one side of the equality in the $P = NP$ problem. In addition, NP-hard $\cap$ $NP = NP$-complete. Famous NP-hard problems are the *traveling salesman* problem [52], the *graph coloring* problem [47, 52], the *longest simple path* problem [47], the *knapsack* problem [53], and many more, often formulated in the language of graph theory [54].

Complexity classes provide a framework in which we can start to consider relations between problems. To this extend we first introduce the notion of *computational equivalence*. Two problems A and B, with complexity classes C_A and C_B, are computationally equivalent if they can be solved using the same algorithm [39]. That is, we can transform A into B and vice versa, without this transformation being more computationally complex than the algorithm solving A or B directly. As a corollary, this implies that if A and B are computationally equivalent, then they belong to the same complexity class. A more relaxed notion is that of *computational reducibility*. We say A can be reduced to B if the transformation from A to B is surjective, but not bijective, and less computationally complex than the algorithm solving B [38]. This means that $C_B \subset C_A$. This gives a better view of the notion of a complete problem. A *complete problem* is any problem A for which there exists no problem B for which A reduces to B [55]. If A is a complete problem then C_A is its *fundamental complexity class*. A broad overview of complexity classes and their interrelations can be found in [47].

In section 4.1.1 we introduced the *halting problem*. Given a machine with some input, it asks to determine whether the machine halts on the input within a finite number of steps [35]. The halting problem is the prime example of an *undecidable* problem [37]. A problem is undecidable if there is no general algorithm that solves the problem for all possible inputs. The fact that the halting problem

is undecidable is of particular interest, since it provides a gateway into proving that other problems are undecidable. In particular, if we have some problem A and we can reduce the halting problem to A or reduce A to the halting problem, then A too is undecidable. As an example consider the problem of determining whether a given statement about natural numbers is true or false. Certainly, if we check the statement for all natural numbers and find no contradiction the statement is true. Otherwise, if we do find a contradiction, we halt the process since we know the statement must be false. Thus, if the process of checking the statement halts then the statement is false, and if it does not halt the statement is true. We have reduced the problem to the halting problem, so it must also be undecidable. The halting problem can be directly related to *Gödel's incompleteness theorems* [56].

The concept of reducing problems to others and mapping between problems is, of course, not new. One could say that it is at the very core of mathematics. When solving a mathematical problem we are constantly mapping between problems to reduce the initial problem to one that we are able to solve, through a series of transformations. This is one of the driving forces in modern mathematics. It is then no surprise that the transformations of problems are most often clever mathematical methods. If there is anything that modern mathematics has shown us, it is the unbelievable power of these equivalences we know as isomorphisms in group theory, isometries for metric spaces, homeomorphisms in topology, diffeomorphisms for manifolds, and computationally equivalent transformations in complexity theory [39, 57–60]. These mappings are of such a usefulness that they were even endowed their own research field, that being category theory [61]. A modern treatment of complexity classes in the context of category theory is given in [62].

Computer science has ever since its birth been a field very close to application [63–65]. Since Turing's time, efforts have grown to treat it in a more theoretical way, but this is not yet comparable to the theoretical fields within mathematics. For a large part this is because it has not yet been necessary to treat the theory of algorithms formally. However, over time there has been an increasing interest in formal methods for developing algorithms. The best example of this is the refinement of matrix multiplication algorithms. For a long time the state of the art for matrix multiplication was Strassen's algorithm [66]. Over time, there has been a long string of papers and research dedicated to producing an algorithm that is even better [67]. Over time, the problem has been pushed to better and better complexity classes. The reason that people involved in this effort were able to compare results is complexity analysis.

4.2 The testing suite

In most cases, an algorithm will have a set of control parameters that are used to refine the algorithm's performance with respect to a specific problem. These parameters usually have a significant impact on the performance of the algorithm. This applies in particular if the algorithm contains a stochastic element, as is the case with a GA. The choice of control parameters has been the subject of many research projects and is still frequently debated [68–70]. What is generally agreed on is that this choice usually depends on the specifics of the problem the algorithm is applied to.

A strategy that is often most straightforward, albeit tedious, is to search the control parameter space directly [71, 72]. This is commonly referred to as a *testing suite*. One starts by considering

probable ranges for each control parameter. This comprises a range for numerical parameters or a set of possible fitness functions one wants to compare. Secondly, one has to identify suitable ways to score the algorithm's performance. Finally, one runs tests for desired combinations of control parameters, recording scores for each. In the end one compares scores and, ideally, picks the combination that gives the highest efficiency. It is often the case that there is not a single best combination. Usually there are at least some control parameters that can either be fixed or restricted to a specific range [73]. There are more advanced implementations of testing suites that leverage stochastic elements, but these are not considered in favor of clarity in this introductory treatment [74].

An important point that we glossed over is how one would go about making the choices described above, such as the mutation amount or crossover chance. Some would say it is intuition, but that seems like an unsatisfactory answer. Intuition is often a good starting point, but what is most important in the end is domain knowledge concerning the problem the algorithm is trying to tackle. For example, domain knowledge would allow the designer to take information about the geometry of the search space. If the designer knows that the system is very sensitive to changes in input, they may expect the search space to have strong oscillatory geometry. In this case they would choose a lower learning/exploration/mutation rate, since setting these rates too high will prevent the algorithm from exploring the search space in a meaningful way, often leading to quick divergence. As another example, if the designer has information about a symmetry of the problem, they may incorporate this to further restrict the search space. If a function is approximated by the algorithm and we know the function to be symmetric, only half of the function needs to be found, restricting the search space. The next section will consider each control parameter and discuss the choices we have made with respect to the algorithm we have implemented.

Before we do this, however, there is a final point to consider. What we implement is essentially a meta-optimization problem. We attempt to find optimal control parameters in order to find optimal parameters for the basis functions. In this regard it is a valid question to ask whether we are artificially increasing the complexity of the whole situation. While brute-force methods have been employed very effectively for simpler problems [32], even simple meta-optimization quickly leads to algorithms that outperform brute-force methods, increasing the incentive to reuse the algorithm [74, 75]. However, this answer merely diverts the question: when is it worth it to do rigorous meta-optimization? The answer is that it really depends on the specific problem at hand. In general, if we expect to treat high-dimensional or otherwise computationally intensive problems, or if we expect to treat the same or similar problems a lot of times, it is almost always worth it to do meta-optimization [76]. In science, we are primarily concerned with finding methods to treat problems in a standardized, reproducible, and rigorous way. Because of this, it is important that the algorithms we use are optimized, consistent, and potentially also scalable. Investing in automated parameter tuning has been shown to improve reproducibility and scalability in a wide variety of problems within physics and in more general settings [73, 77–80].

4.2.1 Control parameters

Control parameters are the degrees of freedom that we have in tuning an algorithm. Our algorithm has the following control parameters.

- **The size and shape of the collocation point grid:** This has a direct effect on the maximum amount of detail that can be resolved by the algorithm. We expect that the most detail is needed around the area were we expect the $k = 0$ endpoint. We place a higher density of points in that region. The number of collocation points was chosen so that we use an order of magnitude less points compared to previous methods [15] in order to really test whether our method is more efficient. We did not vary the size or shape when testing.
- **The choice of basis functions:** The basis functions are chosen to be in line with previous work [15], as to be able to focus on varying parameters related to the GA. There will be a discussion on different choices of basis functions in section 6.2.2. We did not vary the basis functions when testing.
- **Whether and when to fix parameters:** Fixing parameters forces the generating function to abide to the linear behavior around the fixed point, but since the grid has finite resolution, this introduces and error. One could decide to fix parameters only once for the initial population (before the iteration cycle), at each iteration, or never at all. We chose the first option since for the theory and grid we use, it was the only one that gave sensible results.
- **The size of the population:** Population size has a major impact on performance, so it is a good idea, if possible, to keep this low for initial tests and only increase it when other control parameters have been further constrained. It should be noted that this is not always possible. For some problems the GA will simply fail to converge for population sizes taken too small. We chose a small population size of four, eight, and sixteen.
- **The form of the fitness function:** There is some freedom in how fitness is computed. It is preferred to stay as close to the formulation in the original problem as possible, as to not create unforeseen biases [81]. The most common modification is to choose a maximum value. This prevents that scores run away to infinity. It is advisable to really spend time implementing the fitness function in an efficient manner, as it will be evaluated in every iteration for each individual in the population. We chose a maximum value of 10^{30} when testing.
- **The selection method:** There is a large number of possible selection methods. Our choices are explained in section 3.5.
- **The parameters associated with the selection method:** The selection method will have parameters related to how much of the previous generation is kept. It may also have parameters related to how much we favor individuals with good scores from the previous generation. These parameters in general have an impact on the diversity of a generation. Keeping more of the previous generation or favoring individuals with good scores hampers a higher degree of diversity. High diversity helps to explore the solution space. Too little diversity may lead to the algorithm getting stuck in local minima. In addition, it is important to keep more results for a

higher degree of trait retention, since some relevant traits may be present in individuals that otherwise score badly. If too many of those individuals are thrown out, one risks losing the trait that may only show itself as important when introduced into an already more successful individual through crossover. On the other hand, too much diversity may result in the algorithm failing to converge to an optimal solution. We tested keeping between 0% and 40% of the previous generation, while weighing based on fitness scores when selecting individuals from the previous generation. To be specific, keeping $x\%$ means that the best $x\%$ of the previous generation is copied directly to the new one. In this way we make sure the best individuals are not accidentally lost. The rest of the new generation is filled by picking from the entire previous generation at random, possibly with some weighing, as explained in section 3.5.

- **The mutation method:** There is a large number of possible mutation methods. Our choices are explained in section 3.5.

- **The parameters associated with the mutation method:** The mutation method comes with parameters dialing frequency and strength of the mutation. Both a high mutation rate and mutation strength strongly increase diversity. Mutation parameters are different to those used in selection, since those for mutation promote diversity by creating new traits, while selection retains traits that are already present. A risk associated with high mutation parameters is that the algorithm might diverge, as opposed to merely not converging. Diverging behavior is especially dangerous since it will doom the algorithm entirely, while not converging might be temporary and will only slow the algorithm down. Because of this, the mutation parameters are often the most tricky aspect and require the most testing. We tested mutation chances between 0.01% and 10% of genes mutating and mutation amounts between 0.01% and 10% of the previous gene value being added or subtracted.

- **The crossover method:** There is a large number of possible crossover methods. Our choices are explained in section 3.5.

- **The parameters associated with the crossover method:** The crossover method has parameters related to how often crossover occurs. High crossover parameters generally promote diversity. Crossover serves the purpose of combining traits in new ways, since new combinations may lead to synergetic improvements. Too much crossover may break apart too many existing trait blocks, resulting in a population which is too homogeneous. Too little crossover may cause the algorithm to have trouble converging, since some trait combinations will be hard to find by pure mutation. We tested crossover chances between 0% and 80%.

The first three control parameters are specific to the problem we are solving. We name those *meta parameters*. The rest are particular to the GA, we name those *control parameters*, in line with the usual use of the term. In addition, there are the parameters that specify the theory under consideration, those described in section 3.1, that are named *theory parameters*. In the pseudocode presented in appendix A the population size is grouped with the meta parameters instead of the control parameters. This is purely for computational efficiency and consistency, so that the same starting population is used for each test.

4.2.2 Scoring

In order to draw meaningful conclusions once the testing suite has explored the control parameter space one has to decide on ways to score the result of each test in a way that allows one to determine (a range of) the optimal control parameters. These scoring methods are essentially the meta-fitness functions. This choice is guided by the qualities we want to be bestowed upon the algorithm by the choice of control parameters. In essence, this is a multiple objective optimization problem. In most cases this does not have a single solution that is most favorable for all objectives.

We illustrate the concept of scoring based on the mutation rate. A high mutation rate ensures a quicker exploration of the search space; termed "high diversity". However, a high mutation rate also reduces the rate at which the algorithm converges to an optimum. We wish to have both high diversity as well as fast convergence. There is no way to choose a mutation rate such that there is both maximum diversity and maximum convergence.

In the field of game theory, this is resolved by finding a Pareto optimal solution. Given a solution that is scored on multiple criteria, it is called a Pareto optimal solution if for each individual criterion it obtains as high a score as possible, without diminishing the score of any other criterion in the process. A particular solution is the result of making a specific set of choices regarding the control parameters of the algorithm. A Pareto optimal solution is a specific set of choices, where changing one choice will always result in some criterion's score being lowered. In the example this would mean having chosen a mutation rate such that there is no way to change the mutation rate that improves diversity without reducing convergence; or the other way around.

There is no guarantee that a Pareto optimum is unique. In most cases where one exist there are multiple. The set of all Pareto optima is called the Pareto frontier. In general, finding the Pareto frontier is a computationally hard problem that belongs to the NP-hard complexity class [82]. However, we are in luck if we restrict ourselves to a finite set of options for each choice. In this case, the problem of finding the Pareto frontier reduces to the problem of finding the maxima of a point set problem from computational geometry. This problem has been solved for n options and d choices with time complexity $O(n(\ln n)^{d-3} \ln \ln n)$ and space complexity $O(n)$ [83]. This means that in our case finding the Pareto frontier is an insignificant problem compared to any other part of the algorithm or testing suite.

4.2.3 Error estimates

When employing an algorithm it is important to get a good idea about the error. The problem of estimating the error of an optimization algorithm is notoriously difficult.

One reason is that these algorithms are universal approximators [84, 85], meaning they can optimize any function given sufficient capacity. However, this flexibility makes it hard to derive tight bounds. The optimization procedure is often very non-linear and requires specific control parameters for the optimization algorithm to converge. The search space is non-convex, meaning multiple minima exist. This means convergence guarantees are weak, which makes a general error bound or law very elusive [86].

Another reason that is particularly relevant for GAs, but also many other modern optimization

algorithms, is the stochastic nature of the optimization method. This leads to chaotic population dynamics that are very sensitive to the initial population used [81].

When making use of any sort of sampling, as is the case for the basis functions and collocation point grid, there is another error introduced, related to over- or under-fitting. This is called the generalization error. There exist theoretical bounds for this error based on model complexity, but these are often too loose to be useful in practice [87, 88]. This is because they depend heavily on the optimization problem itself and the sampling method [89].

The general problem that covers everything we have just explained is the lack of a closed form solution. The processes that lead to optimization are fundamentally emergent, meaning no simple equation governs their behavior [90].

In practice, error bounds are inferred from experimental validation [91]. In our case this is done by the testing suite. We can compare the results found by the testing suite with previous methods that produced more accurate results, albeit more slowly. In addition, since the fitness function gives a direct measure of the distance away from the optimal solution, we expect that the error is proportional to the fitness function. The Pareto frontier could then be used to give an empirical scaling law for the error based on the other optimization criteria, given that enough tests are done to fit a function to the Pareto frontier that embodies this scaling law.

Chapter 5

Results and Discussion

In this chapter we present the results of our research. We start with the results from the GA in section 5.1, that is, the optimized GA produced by the meta optimization done through the testing suite. After that we will discuss the results found by the testing suite in its entirety in section 5.2. Finally, we will present the complexity analysis for both the shooting method and the GA, and compare them, in section 5.3.

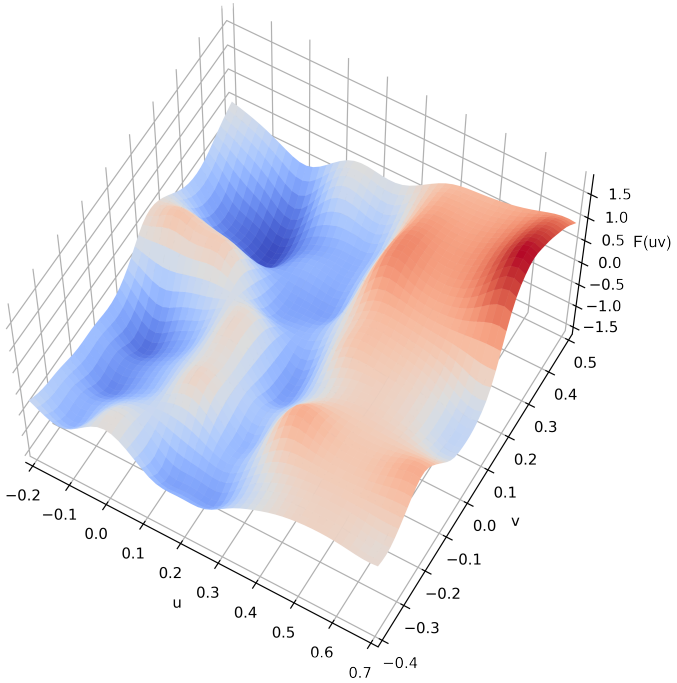


Figure 5.1 Embedding of the two-dimensional surface given by $F(u, v) = z$ in three-dimensional space. The UV-critical hypersurface corresponds to the $z = 0$ plane.

5.1 Genetic algorithm

The testing suite has produced a Pareto frontier of Pareto-optimal results. We will consider in more detail the Pareto-optima that had the best fitness score. This is not an absolute optima, in the sense that there are Pareto-optima that took less time to compute.

To quickly summarize what we expect the GA to provide in our case: The GA should provide the expansion coefficients p_i that, when combined with the basis functions $\psi^i(u, v)$, will give the generating function $F(u, v)$. The generating function can be visualized as a two-dimensional surface embedded into three-dimensional Euclidean space. This is shown in figure 5.1. Inside this three-dimensional Euclidean space, the (truncated) theory space is the uv-plane. The UV-critical surface corresponds to $F(u, v) = 0$. In our case, the UV-critical surface is a one-dimensional subset of theory space, and it contains the NGFP and the $k = 0$ endpoint, which lies on the $u = 0$ line. The v-coordinate of the intersection between $F(u, v)$ and the $u = 0$ line is the final result that can be related to the

couplings associated with observable physics.

Figure 5.2 shows all the important parts that were just described, as well as the $v = 0.15$ result we wish to match from previous methods [15]. The figure shows a section of the uv-plane, filled with the vector field generated by the beta functions. The GA provides the $F(u, v)$ surface whose height is shown by the color gradient. The level curve for $F(u, v) = 0$ is denoted by the dotted line, which should be the UV-critical surface. On the UV-critical surface lies the NGFP and the $k = 0$ endpoint. Figure 5.4 shows the $F(0, v) = z$ slice.

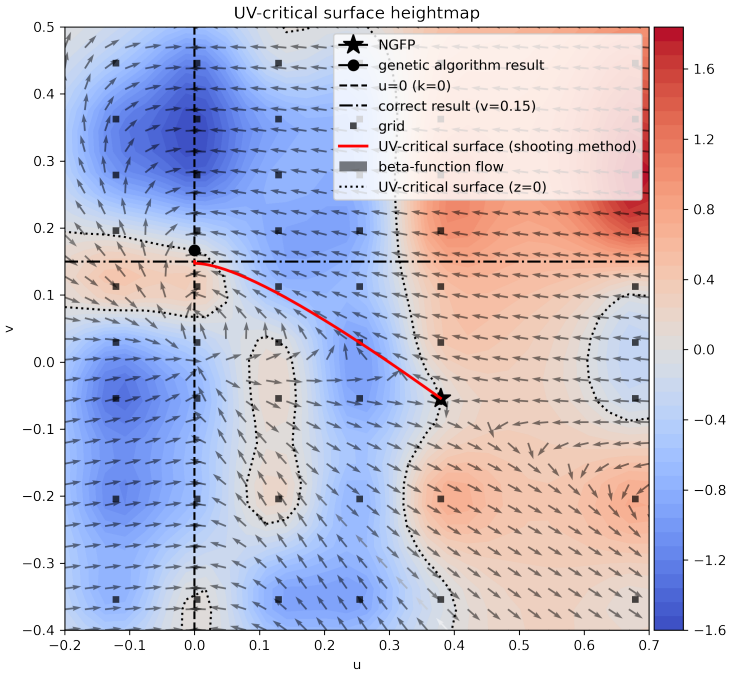


Figure 5.2 A section of theory space. The gradient denotes $F(u, v)$ and the vector field denotes the beta function flow. The dotted line represents the UV-critical surface. Marked are the NGFP (equation (2.13)), the $k = 0$ endpoint, the $u = 0$ slice, and the collocation point grid. The GA does not produce a UV critical trajectory connecting the NGFP and the $k = 0$ endpoint. The red curve shows the path of the shooting method, which may be regarded as the correct UV-critical surface. Note that the direction the beta function flow points into has been inverted, as to better illustrate the flow from $k = \infty$ to $k = 0$.

We observe that the UV-critical surface provided by the GA does not contain a connected $z = 0$ level surface from the NGFP to the $k = 0$ endpoint, at least not in the section of theory space we consider. From previous treatments of this problem we know that this trajectory does exist, and that it should follow approximately a straight line between the two points. The most probably cause for this discrepancy is the resolution of the collocation point grid. The about sixteen collocation points that cover the section of theory space in which the trajectory should form are just not able to encode the information necessary to form the correct trajectory. This becomes evident when we observe that in a lot of occasions the UV-critical surface seems to trace around and between collocation points. If the trajectory is actually supposed to run over a collocation point this will become a problem. A higher resolution may cause the UV-critical surface to lay closer to its true location, as was shown for 500 collocation points in [15]. We have not been able to reproduce this using the GA.

In order to trust the physical meaning of the predicted endpoint, it is critical that a connected trajectory can be constructed. As stated before, we expect that the basis functions are a significant reason why finding a connected trajectory is difficult. Because of this, we have done a pilot experiment using a neural network. The neural network learns $F(u, v)$ directly without the use of basis functions. This is further discussed in section 6.2.3.

Although we were unable to produce a connected trajectory, we can see how it would emerge for higher grid density. Figure 5.3 shows the result of using the optimal control parameters selected by the testing suite, but with a higher grid density. In particular, these are the same control parameters that produced figure 5.2. We observe that for this higher density a correct trajectory starts to emerge, and technically it is connected, but this hardly suffices. This result indicates that indeed the trajectory could emerge with more training or even higher grid density. However, the big problem is inconsistency. We see that the $F(u, v)$ produces a very uneven surface. The general trend shows where S_{UV} should emerge, but the algorithm has trouble locking in. This behavior gives the indication that there are too many parameters to optimize compared to the degrees of freedom required to describe the trajectory; the algorithm is producing noise in an attempt to capture local behavior. The trajectory seems to be too much of a global trend to optimize for with the very localized basis functions. While a more exact trajectory may form at some point, in light of the trail experiment discussed in section 6.2.3, it seems like using basis functions is not the optimal approach.

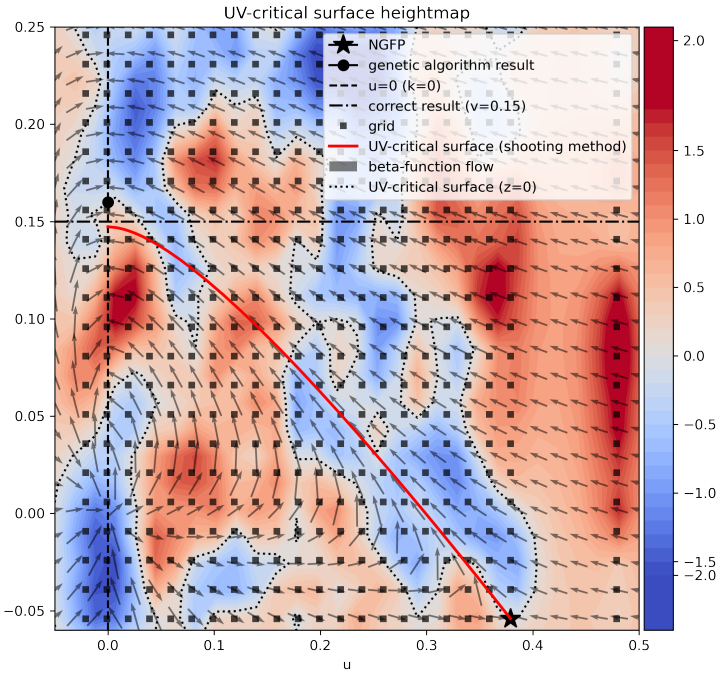


Figure 5.3 A section of theory space with a higher grid density. The gradient denotes $F(u, v)$ and the vector field denotes the beta function flow. The dotted line represents the UV-critical surface. Marked are the NGFP (equation (2.13)), the $k = 0$ endpoint, the $u = 0$ slice, and the collocation point grid. We observe the emerging of a UV-critical trajectory connecting the NGFP and the $k = 0$ endpoint. The red curve shows the path of the shooting method, which may be regarded as the correct UV-critical surface. Note that the direction the beta function flow points into has been inverted, as to better illustrate the flow from $k = \infty$ to $k = 0$.

Somewhat surprisingly, even if the correct trajectory is not present, the endpoint at $k = 0$ can still be located and is very close to the result from previous work [15]. Previous work found $v = 0.15 \pm 0.01$, which is taken to be the correct result for the purpose of training the GA. The GA was able to produce a the result $v = 0.16 \pm 0.01$, where the error is based on the standard deviation of the solutions in the Pareto frontier with population size eight; from figure 5.6 these form a distinct group with better scores. Given the simplicity of the algorithm and the severe reduction of collocation points needed, 54 as opposed to 500, this is a very good result. The GA obtained this result without being able to give the correct trajectory, which is unexpected and not how the algorithm should ultimately function. The trajectory is what is supposed to carry the information from the NGFP to the observable regime. The current approach however does not explicitly consider this trajectory. Instead the information is extracted from the entire beta function field. This raises the question where the information is really stored, a question we will not address directly in this text, but which might be interesting to

keep in mind in future work. In any case, if the algorithm were to be given more training time, a broader parameter set to optimize, or more collocation points, its is expected to do better.

Special care should be given to what figure 5.4 tells us. Notable, there are three, points within the section of the $u = 0$ slice that we consider, where $F(0, v) = 0$. As it stands, the algorithm has no way to distinguish these points in a way that allows it to choose the "correct" one. This has to be done by a human with additional contextual knowledge. For the scenario under consideration this is no problem, but in the future this may need to be addressed further. Contemporary methods that shoot backwards from the $k = 0$ endpoint to the NGFP could possibly be used to address this issue, by verifying whether a point where $F(0, v) = 0$ can indeed reach the NGFP. It also remains to be decided whether these multiple points should really be there or whether they are merely numerical artifact produced by the limited accuracy of the algorithm.

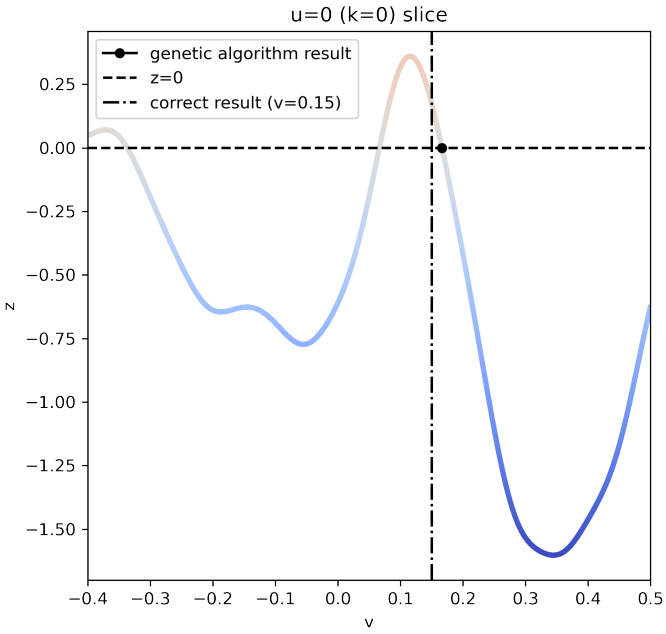


Figure 5.4 The $F(0, v) = z$ slice. The UV-critical hypersurface is at $z = 0$. Denoted are the result from the genetic algorithm and the correct result from previous work. The correct result is not unique and has to be chosen based on context from all $F(0, v) = 0$ points.

When looking at the other Pareto-optima, as well as the result from previous work [15], we observe some defining structures of $F(u, v)$. Focusing on the slice in figure 5.4, we see a clear peak. This peak is also present in almost all other Pareto-optima, as well as in the previous work. Even more defining seems to be the large valley on the positive side of the peak, which is even more apparent in previous work. These findings point at that there are indeed defining features of $F(u, v)$. The question is whether these features are regular between related theories and whether these features tell something about the underlying physics. In this regard a further survey of $F(u, v)$ itself would be in order.

5.2 Testing suite

The testing suite consists of two concrete parts. We start with the evaluation of the exploration of the GAs control parameter space. Secondly, we present the Pareto frontier and its implications.

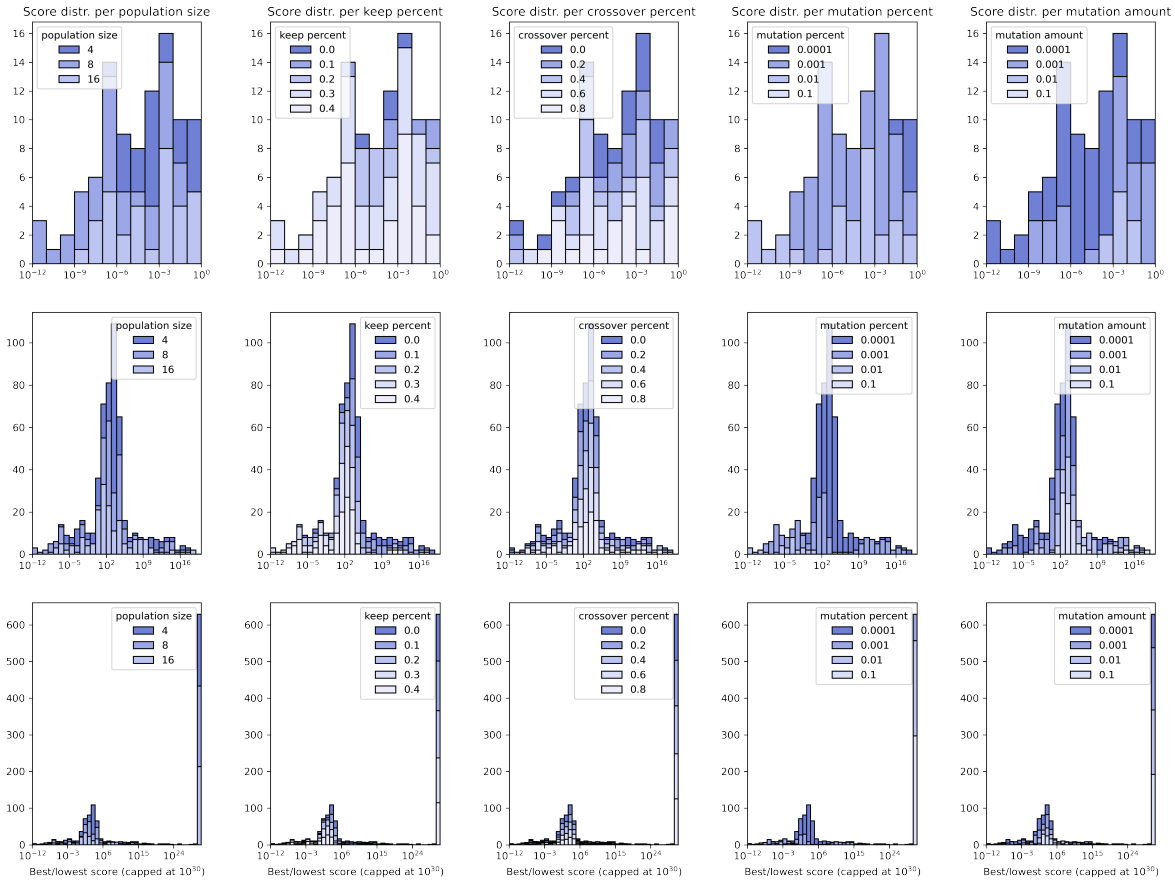
5.2.1 Exploring parameter space

The testing suite explores the control parameter space spanned by population size, keep percent, crossover percent, mutation percent, and mutation amount. Each control parameter takes between three and five values. We investigate how these values are distributed for specific scores produced by the testing suite. This also gives insight into the general score distribution. Figure 5.5a shows the score distribution sorted by population size. We observe that for lower scores, which are better, there are no solutions with population size 4 or 16. This means that while 8 may not be the absolute optimal, the optimal should be bounded between 4 and 16. This would warrant a more detailed testing suite run for population sizes in this interval. We also observe a peak for scores between 1 and 10^5, centered around 10^3. The majority of the solutions at 10^3 have population size 4. It seems that there might be a significant difficulty in the problem that has to be overcome to make the jump to scores below 1, as a lot of solutions get stuck here. The problem may be related to low population size, but this conclusion is not well grounded. Population size seems to have no effect on which solutions diverge to scores beyond 10^{30}.

Figure 5.5b shows the score distribution sorted by keep percentage. We see that for lower scores we require a higher keep percentage. As the upper bound we placed on the keep percentage is included in the lowest scores, we do not know whether a higher keep percentage would have produced even better scores. This should be investigated further. The low keep percentages seem to be almost entirely filtered by whatever complexity is creating the filtering peak at scores between 1 and 10^5. We can conclude with reasonable confidence that a high keep percentage helps deal with this complexity in the problem. Keep percentage seems to have no effect on which solutions diverge to scores beyond 10^{30}.

Figure 5.5c shows the score distribution sorted by crossover percentage. We have no indication that the choice of crossover percentage has any significant impact on the solutions. In further investigations we can safely turn the crossover off entirely. It is important to note that if in further investigations the algorithms seem to hit a wall, it might be helpful to reinvestigate the effect of crossover to determine whether at this point it could make a difference.

Figure 5.5d shows the score distribution sorted by mutation percentage. We observe that for lower scores there are no solutions for 0.0001 and 0.1 mutation probability. This means the optimum is bounded by these values. Further investigation should be done within this interval to determine a more exact optimum. The filtering peak filters out the lowest mutation percentage. We can conclude with reasonable confidence that a high enough mutation percentage is needed to deal with the complexity that causes the filtering peak. In addition, we observe that the divergence to scores $> 10^{30}$ is certainly related to too high of a mutation percentage. In further investigations one should see whether there is some mutation percentage that provides a turnover point where higher percentages diverge and lower percentages converge. If this point can not be found it is expected



(a) Score distribution sorted by population size. From bottom to top the diagrams are zoomed in further towards lower scores.

(b) Score distribution sorted by keep percentage. From bottom to top the diagrams are zoomed in further towards lower scores. Percentages are given as probabilities: between 0 and 1.

(c) Score distribution sorted by crossover percentage. From bottom to top the diagrams are zoomed in further towards lower scores. Percentages are given as probabilities: between 0 and 1.

(d) Score distribution sorted by mutation percentage. From bottom to top the diagrams are zoomed in further towards lower scores. Percentages are given as probabilities: between 0 and 1.

(e) Score distribution sorted by mutation amount. From bottom to top the diagrams are zoomed in further towards lower scores.

that the divergence behavior is intertwined with some other control parameter of the algorithm, probably mutation amount, as we shall see next.

Figure 5.5e shows the score distribution sorted by mutation amount. We observe that for lower scores the solutions almost entirely consist of the lower bound we set on mutation amount. This suggest that the optimum could still be an even lower mutation amount. This should be investigated further. Additionally we see that the filter peak entirely filters the highest mutation amount that was tested. If we look at the divergence behavior we see a slight dependence on mutation amount, where

higher mutation amounts are more likely to cause divergence. In the future it should be investigated if values for the two mutation parameters can be chosen so that convergence is ensured.

The results produced by the testing suite are mostly in line with what we expect from the control parameters. Each control parameter tunes a specific function of the algorithm, as described in section 4.2.1. In particular, the control parameters associated with mutation have a clear impact on divergence, while choosing a high enough keep percent promotes convergence, as expected. Most unexpected is the result that a higher population size does not produce better results automatically, which is what is generally expected. This could be explained by the algorithm getting overwhelmed by a large number of local minima. This means that for a larger population size there is no clear "best individual in the population", which hampers convergence. By forcing a lower population size, you artificially force the population to act less as a homogeneous group, meaning local minima are less likely remain as a sustained feature in the population. However, this reasoning is only a possible scenario and in practice it is very hard to determine how features, like high population size not always being better, emerge. The result that crossover is less important indicates that there may not be many major synergies between traits.

5.2.2 The Pareto frontier

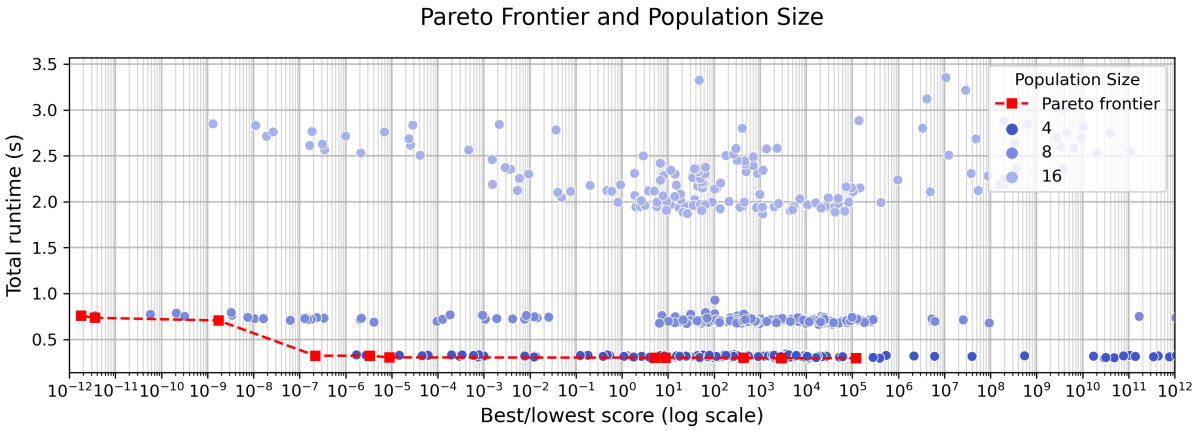


Figure 5.6 Pareto frontier produced by the testing suite. All blue solutions are dominated by solutions on the frontier. Scores are displayed up to 10^{12}, the full figure can be found in appendix B. No solutions with population size sixteen are present on the frontier.

From all the solutions provided by the testing suite we need to select all multi-objective optima, these form the Pareto frontier introduced in section 4.2.2. The Pareto frontier produced by the testing suite is shown in figure 5.6. Note that this figure includes scores up to 10^{12}, the entire figure can found in appendix B. The Pareto frontier splits the solution space in two. Those points outside the frontier are unreachable solutions, no solution to the problem can ever be this optimal. The points inside the frontier are all dominated by some solution on the frontier. If these dominated solutions were further optimized, then at most they could end up on the frontier.

The frontier is characteristic for the problem-algorithm combination. Another algorithm applied to the same problem or another problem solved by the same algorithm could have a different frontier. The frontier for our problem-algorithm combination actually has a very interesting feature.

None of the solutions with population size sixteen are present on the frontier. This means that for the control parameter set of our algorithm, there is a finite score-optima for population size, which lies between eight and sixteen. This optima could be local, that is, there could be some population size larger then sixteen that could be even better, but the existence of even a local optima allows one to fix one parameter, population size, in further testing. One could then optimize the other parameters, after which one may explore additional optima above population size sixteen.

These kind of insights gained from the Pareto frontier are what makes it such a powerful tool in making educated decisions on meta-optimization. Another kind of insight is that the shape of the Pareto frontier can give insight into the scaling law of each optimization objective with respect to some optimization parameter. In our case, we can gather that runtime always increases with population size, which is not too surprising, but further investigation of different population sizes might give an idea of the scaling law with which the runtime increases. Additionally, for our case, it seems that the minimum score has at least some second order dependence on population size, that is, it admits a bounded minimum.

These scaling laws are not the only ones that can be gathered from the Pareto frontier. Since the fitness score is a measure of the accuracy of the solution, it is expected that the error of the genetic algorithm is proportional to the fitness score. Given enough solutions produced by the testing suite, one could derive a scaling law for the error of the genetic algorithm. We do not have enough data to do this with enough accuracy.

5.3 Complexity analysis of the algorithms

5.3.1 Shooting method

The complexity of the shooting method can be defined in terms of two parameters. The first is the resolution ε. This parameter is related to the accuracy of the numerical method. In the case of the shooting method it goes as the inverse of the step size of the numerical integrator. The second parameter is M, which is the dimension of S_{UV}.

The complexity of the shooting method is found as the complexity of taking a single shot times the amount of shots being taken. The complexity of taking one shot is proportional to the steps involved in taking the shot. This is proportional to the diagonal of a hypercube with length ε and dimension M, so its scales as $\mathcal{O}(\varepsilon\sqrt{M})$. At each step M equations have to be evaluated. The number of shots to be taken scales with the volume of the aforementioned hypercube, since we shoot at some constant fraction of all points in this space. The total complexity becomes $\mathcal{O}(M \times \varepsilon\sqrt{M} \times \varepsilon^M) = \mathcal{O}(\varepsilon^{M+1}M^{3/2}) \approx \mathcal{O}(\varepsilon^M)$ for large M.

We see that the shooting method belongs to the polynomial complexity class for ε and the exponential complexity class for M. It is important that for large M the exponential term always dominates the scaling. For intermediate M, where the exponential and square root term are of same order of magnitude, both terms should be included and we say the complexity is $\mathcal{O}(\varepsilon^{M+1}M^{3/2})$. This happens when $\varepsilon \approx M^{\frac{3}{2M}}$, which is at a maximum for $M = e$ such that $\varepsilon \approx 1.7$. Practically, such a resolution is unrealistic, as it would mean doing numerical integration with only $\mathcal{O}(1)$ integration

steps. From this it is safe to conclude that the shooting method has exponential complexity.

5.3.2 Genetic algorithm

The complexity of the GA is also defined in terms of $\tilde{\varepsilon}$ and M, where $\tilde{\varepsilon}$ takes on a different value from ε. In the case of the GA we define $\tilde{\varepsilon}$ to be the side-length of the hypercube of collocation points such that $\tilde{\varepsilon} = M_{CP}^{1/M}$.

The complexity then scales with the calculation of the fitness functions, in our case the master equation, which is simple tensor contraction. The complexity of tensor contraction is given as $\mathcal{O}(\text{output dimensions} \times \text{contracted dimensions})$ which in our case leads to complexity of order $\mathcal{O}(\tilde{\varepsilon}^M)$ for large M. The GA then also falls in the exponential complexity class.

At first glance the question arises why the GA would be preferred over the shooting method. The secret lies in the difference between ε and $\tilde{\varepsilon}$. In the best case $\varepsilon = \mathcal{O}(100)$ [92], while realistically one finds that it is one to three orders of magnitude higher, up to $\mathcal{O}(10^5)$ for hard problems [93, 94]. In comparison, for the example given, we used $\tilde{\varepsilon} = 6$ and previous work used $\tilde{\varepsilon} = \mathcal{O}(20)$ [15]. This means that effectively the shooting method has complexity $\mathcal{O}([100\tilde{\varepsilon}]^M) - \mathcal{O}([10^5\tilde{\varepsilon}]^M)$. Since we expect that M will not exceed $\mathcal{O}(100)$ for any computationally tractable problems, the factor $100 - 10^5$ has a very significant impact.

Chapter 6

Conclusion and Outlook

6.1 Summary of results

Working in the framework of asymptotic safety, we can treat problematic divergences due to virtual particle loops by UV-completing a theory by an interacting renormalization group (RG) fixed point. Extracting physics predictions from this high-energy completion requires constructing the UV-critical hypersurface of the fixed point and, in particular, the admissible endpoints of the RG trajectories where all quantum corrections are taken into account. The latter form a subspace within S_{UV}.

In order to construct S_{UV} one has to solve a boundary value problem. The historical approach formulates this problem as an initial value problem and solves for individual trajectories in S_{UV} using the shooting method. This method works quite well for low-dimensional scenarios, but becomes intractable as the dimensionality of the problem increases. In this case one has to search for "physical" trajectories on a higher dimensional space, among a set of possibly infinite trajectories. Instead, the boundary value method proposed in [15] solves problem directly. This is done by formulating S_{UV} implicitly as the surface $F_\mu = 0$, where F_μ are generating functions that are invariant under the RG flow and solve the master equation (2.8). By solving the master equation, which is a system of partial differential equations (PDEs), one finds F_μ and thus S_{UV} and the $k = 0$ endpoints.

This thesis presents an algorithmic treatment of the boundary value method. The boundary value method is formulated as an optimization problem by expanding F_μ into a set of basis functions, whose expansion coefficients can be optimized such that F_μ solves the master equation. The optimization problem was solved by means of a genetic algorithm (GA). We then leveraged techniques from the field of computer science to optimize the GA itself, by means of meta-optimization through a testing suite. This gave insight into the values that should be chosen for each control parameter. We found that the parameters associated with the mutation operation had the greatest impact, since they directly affected whether the algorithm would converge.

The testing suite produced a large set of solutions to the boundary value problem, of varying quality. We constructed the Pareto frontier to give a optimal solution boundary for this multi-objective optimization problem, where runtime and accuracy where were both considered. This emphasized that for the boundary value problem, a larger population size may not be more beneficial, and even produced worse results in several cases. When more tests are included into the Pareto frontier, a scaling law between model error and runtime may be found.

We used complexity theory to show that the GA should outperform the shooting method as the dimensionality of the problem increases. This is because the GA uses a resolution of order $\mathcal{O}(6) - \mathcal{O}(20)$, while the shooting method uses a resolution of order $\mathcal{O}(100) - \mathcal{O}(10^5)$. The complexity of both algorithms were found to be exponential in the dimension of S_{UV}, but since this resolution is the base of the exponential, the GA scales much better.

The GA was able to recover the $k = 0$ endpoint to the same precision as previous methods, but was not able to construct a connected S_{UV}. It is expected that this failure is due to the number or choice of basis functions. A trail experiment seeking to improve this result is discussed in section 6.2.3. Otherwise, the GA used less collocation points and less time to obtain its results.

In addition to applications to the GA, the testing suite has been discussed as a general framework to be utilized in the further development of algorithms. The testing suite provides a very general way of optimizing algorithms in a rigorous and structured way. It provides insight into how different parts of the algorithm are responsible for solving different aspects of the problem. In doing so we learn about the underlying structure of the problem and get an idea about which methods to try next. This information is invaluable in the process of developing algorithms in a rigorous way. Applying the testing suite to the GA provides an insightful pedagogical example of meta-optimization, since the control parameters are relatively easy to interpret. This has the advantage that we can reason why some control parameters have the optimal choices they do. For most algorithms this will not be the case, meaning one has to put full trust in the testing suite.

We treated complexity theory as a general framework to classify problems and solutions. We argued why this classification is important by showing how it may be used to reduce unsolved problems to others that do have a solution. This solution can then be applied to the original problem. Additionally, complexity theory provides a way of comparing algorithms in an absolute way, that allows researchers to compare algorithms and systematically improve them. Without this, there really is no measure to compare different algorithms' performances.

6.2 Outlook

One of the goals of the research presented in this thesis is to provide a foundation of ideas and examples that can be built upon in further research. We do not claim that the GA is the most efficient or accurate treatment of the boundary value problem. The GA is, however, a very simple model that demonstrates the principles involved in algorithmic research. The control parameters have clear interpretations and the expansion coefficients are a sensible object to optimize. All in all, the GA is not without flaws. Some of these flaws will be addressed in this outlook, as well as some further ideas about how one might continue the research presented.

6.2.1 Applications of the boundary value approach

The boundary value approach treats a very general mathematical problem that can be applied in a variety of contexts. We are given a space endowed with a vector field as well as a point in this space and some linear information around this point. We are then able to construct the surface tangent to

the vector field that includes the given point. The GA is only one optimization algorithm and more advanced algorithms are expected to improve upon it. Solving the boundary value problem has widespread applications. For example, the problem comes up in the study of dynamical systems and chaos theory (e.g. the construction of invariant manifolds in phase space) [95], general relativity and black hole physics (e.g. in horizon problems) [96], fluid dynamics (e.g. the flow around objects) [97], control theory (e.g. finding reachability sets for dynamical systems) [98], and many other fields.

6.2.2 Basis functions

The algorithm as implemented in this research used only a single kind of basis function. For this initial study, this was done to somewhat limit the search space of the testing suite. However, in principle, there is no reason to stick to one kind of basis function. A problem with the MCDs we used is that they are localized at the collocation point they are assigned to. The MCDs are very good at encoding finer detail, but since they have a set smoothness, this detail is effectively restricted to one scale. This is a problem, as we have no reason to expect patterns would not arise at multiple scales. By restricting the basis functions to one scale we are imposing an artificial minimum error into the algorithm. This is of course undesirable.

If one were to move forward with the basis functions, a first step could be to let the smoothness of each basis function become an optimization parameter, alongside the expansion coefficients. The difficulty with this approach is that the basis functions can not be pre-calculated any more, and have to be recalculated at every iteration. This will slow the algorithm down significantly. Alternatively, one could introduce a few global basis functions to encode large scale behavior. A natural choice would be a few Fourier terms, or a single sinusoid with tunable amplitude, phase, and frequency. This approach of mixing different types of basis functions has already been shown to be successful in other problems [99].

6.2.3 Pilot Experiment: Physics Informed Neural Networks

The prominent failure of the GA is its inability to give a connected S_{UV}. The existence of a connected S_{UV} is an absolute requirement for the theory under consideration to have any physical grounding. We found that the reason why a connected S_{UV} fails to emerge is likely because of the choice and number of basis functions. A natural question then is to ask: Can we do without the basis functions altogether? The answer is, yes!

Without the basis functions we have to predict F_μ directly. The natural algorithm to do this is the *Neural Network (NN)*. A NN in its simplest from is a stand-in for a function of an n-dimensional input vector to an m-dimensional output vector. It has been found that a large enough NN can approximate any function on a domain where it is continuous [100]. Between the input and the output is a network of hidden layers. These form a graph, representing a network, whose edge-weights are learned to have the NN mimic the function as accurately as possible. The behavior of the function that the NN mimics is encoded by a *loss function*. The loss function gives a value, the loss, that should be at a global minimum when the NN exactly mimics the functions that it is a stand-in for. In this way, the loss function serves virtually the same role as the fitness function in the GA. We say that training the

NN is exploring the *loss landscape* to find the global minimum. If no global minimum exists, then the the function being mimicked is not unique and the problem is underdetermined [86].

In broad lines, the optimization cycle is as follows. During *forward propagation* some sample input data is fed through the network, layer by layer, where calculations are done based on the edge-weights combined with different procedures for combining all edges entering a vertex. The loss function gives a loss value for each output, which quantifies the error of this output. During *backpropagation* the gradient of the loss function with respect to each edge weight is calculated in a process that leverages the chain rule from calculus to efficiently carry the error signal back through the network layers. The edge weights are then updated according to the loss-gradient in the direction that reduces the loss the most. This cycle is repeated until the desired accuracy is achieved [86].

The type of NN that is tailored to our application is the *Physics Informed Neural Network (PINN)*. The "Physics Informed" part is somewhat of a misnomer, as these networks are in no way particular to physics. A PINN is merely a NN that treats PDEs together with their boundary conditions [101]. The inclusion "Physics Informed" stems from the original application, where the boundary condition is a dataset of physical data from experiments. The PINN involves multiple loss functions that are combined together, in the simplest case just by summation. The PINN then finds the global minimum of the combined loss landscape. There is one loss function per PDE and one per boundary condition. A prominent application of PINNs is in solving the Navier-Stokes equations to find the flow around particular geometries [102].

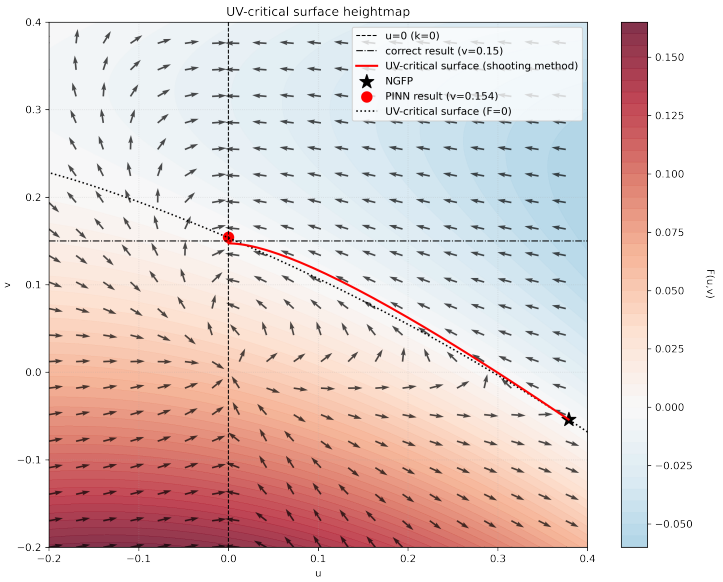


Figure 6.1 A result of the Physics Informed Neural Network trail experiment. The gradient denotes $F(u, v)$ and the vector field denotes the beta function flow. The dotted line represents the UV-critical surface. Marked are the NGFP, the $k = 0$ endpoint, the $u = 0$ slice, and the correct result from previous work. The PINN produces a UV-critical trajectory connecting the NGFP and the $k = 0$ endpoint. The red curve shows the path of the shooting method, which may be regarded as the correct UV-critical surface. Note that the direction the beta function flow points into has been inverted, as to better illustrate the flow from $k = \infty$ to $k = 0$.

In our case, the PDEs are of course given by the master equation, while the boundary condition is the linear behavior around the fixed point. The input would be a random set of points covering theory space. In this way we can do without a rigid grid or a set of basis functions. This presents the real first step away from spectral methods [103–105].

As a trail experiment we implemented a very basic PINN and had it predict the same $F(u, v)$ as the GA was asked to predict. Figure 6.1 shows a result of the trail exam, as well as the correct S_{UV} as obtained by the shooting method superimposed [15]. This result was obtained without any major

optimization. We see that the S_{UV} found by the PINN is connected and almost entirely tangent to the beta functions. For the trail experiment, 200 solutions were found, with varying training times. Half of these solutions used 100 input points and the other half used 10000. Those using 100000 used double the computation time compared to those using 100, suggesting a better time complexity then any algorithm seen applied to this problem before. Both the 100 and 10000 samples version were faster than any previous method by at least one order of magnitude. Among the 200 solutions none showed the kind of disconnected S_{UV} that the GA produced, and about 9% show an S_{UV} that never reaches the $u = 0$ line. All things considered, the pilot experiment shows that PINNs are indeed a very promising method to investigate moving forward.

6.2.4 Adaptive theory expansion

An important feature of both the GA and a PINN, that has not been explored yet in this research, is the fact that the training process can be stopped and picked back up again at any time. All one has to do is save the current generation in the case of the GA, or the current edge weights in the case of the PINN. This opens the possibility of changing the theory under consideration during the training process.

An immediate application one may think of is the adaptive expansion of the theory by increasing the order of the truncation. One could present the algorithm with a theory truncated at order n, and optimize for this order of truncation. One then pauses the algorithm, increases the order of the truncation to $n + 1$, and start the optimization process up again. Since the optimal solution for order n should be close to the optimum for order $n + 1$, the optimization should converge better than if one had started at order $n + 1$. This process of increasing task difficulty during training is called *curriculum learning* and can in principle be continued indefinitely [106].

We can take this idea even further by including a *scheduler* into the algorithm [107, 108]. A scheduler is a background process that makes changes while the algorithm is running. When these changes trigger is based on the schedule. Events on the schedule could trigger at a set number of iterations, or for example when a set fitness score or loss value is reached. The event could change some of the algorithms control parameters. For example, one may want the mutation strength to decrease as the algorithm gets closer to the minimum [109]. On the other hand, one could also use this scheduler to increase the order of truncation of the theory when the previous truncation has reached a low enough fitness score or loss value. Another application would be to place more focus on areas where the algorithm has difficulty predicting F_μ, e.g. by adaptively adding collocation points in this area.

In these and many more scenarios, schedulers provide a easy way to adaptively change the theory or algorithm during training. This makes the training process less rigid and linear. This may provide an advantage in more complex scenarios.

6.2.5 Physical theories

In this research, the algorithm has been applied to a toy model. This was desirable, as this model was very well studied. This allowed the research to focus primarily on the algorithmic treatment

of the problem. The toy model has, however, even with the shooting method, a quite easy to compute S_{UV}. Moving forward, we would like to focus on more complex theories. In particular, we would like to show that the GA or its probable predecessor, the PINN, can treat physical problems. A prime candidate is the model presented in [1]. This model shows the emergence of slow-roll cosmological inflation out of a scalar-tensor theory in four-dimensional Euclidean space. Importantly, the predictions made by solving the system of equations for this model can be compared to physical observations. The model is an absolute step up from the toy-model considered up until now and it will provide a more interesting physical connection.

6.3 Closing remarks

The research presented in this thesis tackles the general mathematical problem of finding the surface tangent to a vector field given a single known point and boundary conditions at this point. We have focused on the application within asymptotically safe gravity, but the methods used are absolutely general. The process of algorithm design and optimization has been treated, with as an example the GA. Once again, the ideas presented here apply to a very general context. The GA shows some flaws, but the trail experiment indicates that most of these flaws are circumvented when physics informed neural networks are employed. This indicates a clear route moving forward and presents the first major alternative to spectral methods. The GA serves as an insightful pedagogical example of how meta-optimization can be applied, since the control parameters are relatively easy to interpret. The generality of the problem treated also invites future research to apply the methods presented in a wider context. In this regard it would be most interesting to tackle theories that exhibit a connection to observable physics.

Bibliography

[1] A. Silva, *Emergence of inflaton potential from asymptotically safe gravity*, 2024. arXiv: 2406.10170 [hep-th].

[2] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*. Westview Press, 1995.

[3] S. Weinberg, "Critical Phenomena for Field Theorists," in *14th International School of Subnuclear Physics: Understanding the Fundamental Constitutents of Matter*, Aug. 1976. DOI: 10.1007/978-1-4684-0931-4_1.

[4] K. G. Wilson and J. Kogut, "The renormalization group and the ε expansion," *Physics Reports*, vol. 12, no. 2, pp. 75–199, Aug. 1974. DOI: 10.1016/0370-1573(74)90023-4.

[5] G. Parisi, "On non-renormalizable interactions," in *New Developments in Quantum Field Theory and Statistical Mechanics Cargèse 1976*. Springer US, 1977, pp. 281–305. DOI: 10.1007/978-1-4615-8918-1_12.

[6] C. Wetterich, "Exact evolution equation for the effective potential," *Physics Letters B*, vol. 301, no. 1, pp. 90–94, Feb. 1993. DOI: 10.1016/0370-2693(93)90726-X.

[7] T. R. Morris, "The exact renormalization group and approximate solutions," *International Journal of Modern Physics A*, vol. 09, no. 14, pp. 2411–2449, 1994. DOI: 10.1142/S0217751X94000972.

[8] M. Reuter and C. Wetterich, "Effective average action for gauge theories and exact evolution equations," *Nuclear Physics B*, vol. 417, no. 1, pp. 181–214, Apr. 1994. DOI: 10.1016/0550-3213(94)90543-6.

[9] M. Reuter, "Nonperturbative evolution equation for quantum gravity," *Physical Review D*, vol. 57, no. 2, pp. 971–985, Jan. 1998. DOI: 10.1103/PhysRevD.57.971.

[10] M. Reuter and F. Saueressig, "Renormalization group flow of quantum gravity in the einstein-hilbert truncation," *Physical Review D*, vol. 65, no. 6, Feb. 2002. DOI: 10.1103/physrevd.65.065016.

[11] K. G. Wilson, "Renormalization Group and Critical Phenomena. I. Renormalization Group and the Kadanoff Scaling Picture," *Phys. Rev. B*, vol. 4, pp. 3174–3183, 1971. DOI: 10.1103/PhysRevB.4.3174.

[12] F. Saueressig, *The functional renormalization group in quantum gravity*, 2023. arXiv: 2302.14152 [hep-th].

[13] Y. Lozano *et al.*, "Supertrace," in *Concise Encyclopedia of Supersymmetry: And noncommutative structures in mathematics and physics*. Springer Netherlands, 2004, pp. 476–476. DOI: 10.1007/1-4020-4522-0_637.

[14] M. Demmel, F. Saueressig, and O. Zanusso, "A proper fixed functional for four-dimensional Quantum Einstein Gravity," *JHEP*, vol. 08, p. 113, 2015. DOI: 10.1007/JHEP08(2015)113.

[15] F. Saueressig and A. Silva, *On harvesting physical predictions from asymptotically safe quantum field theories*, 2024. arXiv: 2403.08541 [hep-th].

[16] Á. Pastor-Gutiérrez, J. M. Pawlowski, and M. Reichert, "The asymptotically safe standard model: From quantum gravity to dynamical chiral symmetry breaking," *SciPost Physics*, vol. 15, no. 3, Sep. 2023. DOI: 10.21468/scipostphys.15.3.105.

[17] Y. Wang, Z. Lin, Y. Liao, H. Liu, and H. Xie, *Solving high dimensional partial differential equations using tensor neural network and a posteriori error estimators*, 2024. arXiv: 2311.02732 [math.NA].

[18] K. K. Sabelfeld and N. A. Simonov, *Stochastic methods for boundary value problems: numerics for high-dimensional PDEs and applications*. Walter de Gruyter GmbH & Co KG, 2016.

[19] M. Reuter and F. Saueressig, *Quantum Gravity and the Functional Renormalization Group: The Road towards Asymptotic Safety*. Cambridge University Press, 2019.

[20] S. M. W and H. B. K, *Strickberger's evolution*. Jones and Battlett learning, 2014.

[21] National Academies of Sciences, Engineering, and Medicine. "Evolution resources." (2016).

[22] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950. DOI: 10.1093/mind/LIX.236.433.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 2nd. MIT Press, 1992, ISBN: 978-0262581110.

[24] J. H. Holland, "Adaptation in natural and artificial systems," in *Proceedings of the International Symposium on Circuits and Systems*, IEEE, 1975, pp. 123–134.

[25] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, pp. 620–630, 4 May 1957. DOI: 10.1103/PhysRev.106.620.

[26] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[27] G. R. Price, "Selection and covariance," *Nature*, vol. 227, pp. 520–521, Aug. 1970. DOI: 10.1038/227520a0.

[28] K. Jebari, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, pp. 333–344, Dec. 2013.

[29] N. Soni and T. Kumar, "Study of various mutation operators in genetic algorithms," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4519–4521, 2014.

[30] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 1, 1995, pp. 384–. DOI: 10.1109/ICEC.1995.489178.

[31] A. J. Umbarkar and P. D. Sheth, "Crossover operators in genetic algorithms: A review," *ICTACT Journal on Soft Computing*, vol. 6, no. 1, pp. 1083–1092, Oct. 2015. DOI: 10.21917/jisc.2015.0150.

[32] D. E. Knuth, *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, 1997.

[33] P. Bachmann, *Analytische Zahlentheorie*. Teubner, 1894, vol. 2.

[34] E. Landau, *Handbuch der Lehre von der Verteilung der Primzahlen*. B. G. Teubner, 1909, vol. 1, p. 61.

[35] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 2, no. 42, pp. 230–265, 1936. [Online]. Available: http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf.

[36] S. C. Kleene, *Introduction to Metamathematics*. 1952.

[37] M. Davis, Ed., *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Raven Press, 1965.

[38] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[39] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[40] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71, ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[41] S. A. Cook, "The P versus NP problem," *Clay Mathematics Institute Millennium Problems*, 2000.

[42] T. Yato and T. Seta, "Complexity and completeness of finding another solution and its application to puzzles," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E86-A, no. 5, pp. 1052–1060, 2003.

[43] H. Simonis, "Sudoku as a constraint problem," in *Proceedings of the CP Workshop on Modeling and Reformulating Constraint Satisfaction Problems*, 2005, pp. 13–27.

[44] C. H. Papadimitriou, "Computational complexity," in *Encyclopedia of Computer Science*. 2003, pp. 260–265, ISBN: 0470864125.

[45] R. E. Ladner, "The circuit value problem is P-complete," *ACM SIGACT News*, vol. 7, no. 1, pp. 27–29, 1975. DOI: 10.1145/1008304.1008308.

[46] L. G. Khachiyan, "A polynomial algorithm in linear programming," *Doklady Akademii Nauk SSSR*, vol. 244, no. 5, pp. 1093–1096, 1979.

[47] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979, ch. 6 (P-Completeness), ISBN: 978-0716710455.

[48] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956. DOI: 10.4153/CJM-1956-045-5.

[49] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995, ch. 4 (Reductions), ISBN: 978-0195085914.

[50] D. H. Younger, "Recognition and parsing of context-free languages in time n^3," *Information and Control*, vol. 10, no. 2, pp. 189–208, 1967. DOI: 10.1016/S0019-9958(67)80007-X.

[51] N. D. Jones and W. T. Laaser, "Complete problems for deterministic polynomial time," *Theoretical Computer Science*, vol. 3, pp. 105–117, 1976. DOI: 10.1016/0304-3975(76)90068-2.

[52] R. M. Karp, "Reducibility among combinatorial problems," pp. 85–103, 1972. DOI: 10.1007/978-1-4684-2001-2_9.

[53] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994, ch. 9.4 (Knapsack), ISBN: 978-0201530827.

[54] R. Diestel, *Graph Theory*, 5th. Springer, 2017, ch. 10 (Hamiltonian Cycles), ISBN: 978-3662536216.

[55] C. Moore and S. Mertens, *The Nature of Computation*. Oxford University Press, 2011.

[56] S. Aaronson, *Rosser's theorem via turing machines*, 21 July 2011.

[57] P. Aluffi, *Algebra: Chapter 0*. American Mathematical Society, 2009, ISBN: 0821847813.

[58] M. Ó Searcóid, *Metric Spaces*. Springer, 2006, ISBN: 1846283698.

[59] J. R. Munkres, *Topology*. Prentice Hall, 2000, ISBN: 0131816292.

[60] J. M. Lee, *Introduction to Smooth Manifolds*. Springer, 2012, ISBN: 1441999817.

[61] F. W. Lawvere and S. H. Schanuel, *Conceptual Mathematics: A First Introduction to Categories*. Cambridge University Press, 2009, ISBN: 0521719162.

[62] J. M. E. Hyland and A. J. Power, "Categories for complexity," *Mathematical Structures in Computer Science*, vol. 17, no. 6, pp. 1173–1211, 2007.

[63] A. B. Tucker, *The Computer Science Handbook*, 2nd. CRC Press, 2014, ch. 1.

[64] W. Isaacson, *The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution*. Simon & Schuster, 2014.

[65] P. J. Denning, "The science of computing," *Communications of the ACM*, vol. 28, no. 4, pp. 350–353, 1985.

[66] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969. DOI: 10.1007/BF02165411.

[67] M. Bläser, "Fast matrix multiplication," *Theory of Computing Library, Graduate Surveys*, vol. 5, pp. 1–60, 2013. DOI: 10.4086/toc.gs.2013.005.

[68] K. L. Mills, J. J. Filliben, and A. L. Haines, "Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters," *Evolutionary Computation*, vol. 23, no. 2, pp. 309–342, Jun. 2015. DOI: 10.1162/EVCO_a_00137.

[69] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986. DOI: 10.1109/TSMC.1986.289288.

[70] B. Freisleben and M. Härtfelder, "Optimization of genetic algorithms by genetic algorithms," in *Artificial Neural Nets and Genetic Algorithms*, 1993, pp. 392–399, ISBN: 978-3-7091-7533-0.

[71] H. Almulla and G. Gay, "Learning how to search: Generating effective test cases through adaptive fitness function selection," *Empirical Software Engineering*, vol. 27, no. 2, p. 38, Jan. 2022. DOI: 10.1007/s10664-021-10048-8.

[72] A. Arcuri and G. Fraser, "On parameter tuning in search based software engineering," in *Search Based Software Engineering*, Springer Berlin Heidelberg, 2011, pp. 33–47, ISBN: 978-3-642-23716-4.

[73] A. E. Eiben and S. K. Smit, "Parameter selection in evolutionary algorithms," *Genetic Programming and Evolvable Machines*, vol. 12, pp. 1–25, 2011.

[74] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[75] G. Wilson *et al.*, "Best practices for scientific computing," *PLoS Biology*, vol. 12, no. 1, 2014.

[76] L. Yang and A. Shami, "Hyperparameter optimization: A survey of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[77] H. Ranocha, M. Sayyari, L. Dalcin, M. Parsani, and D. I. Ketcheson, "Automatic tuning of time integrators for partial differential equations," *Journal of Computational Physics*, vol. 457, 2022.

[78] R. Legin, M. Isi, K. W. K. Wong, Y. Hezaveh, and L. Perreault-Levasseur, *Gravitational-wave parameter estimation in non-gaussian noise using score-based likelihood characterization*, 2024. arXiv: 2410.19956 [astro-ph.IM].

[79] A. Radovic *et al.*, "Machine learning at the energy and intensity frontiers of particle physics," *Journal of Instrumentation*, vol. 13, no. 06, 2018.

[80] K. Lejaeghere *et al.*, "Reproducibility in density functional theory calculations of solids," *Science*, vol. 351, no. 6280, 2016.

[81] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[82] H. E. Romeijn, D. R. Morales, and W. van den Heuvel, "Computational complexity of finding pareto efficient outcomes for biobjective lot-sizing models," *Naval Research Logistics: an international journal*, vol. 61, no. 5, pp. 386–402, Jan. 2014. DOI: 10.1002/nav.21590.

[83] H. N. Gabow, J. L. Bentley, and R. E. Tarjan, "Scaling and related techniques for geometry problems," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery, 1984, pp. 135–143. DOI: 10.1145/800057.808675.

[84] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[85] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[87] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[88] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[89] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[90] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2015.

[91] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[92] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 2nd. Wiley, 2008, ch. 3 (Runge-Kutta Methods), ISBN: 978-0470723357.

[93] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd. Springer, 1993, ch. II.4 (Error Control), ISBN: 978-3540566700.

[94] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998, ch. 4 (Stiffness), ISBN: 978-0898714128.

[95] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 1983.

[96] R. M. Wald, *General Relativity*. University of Chicago Press, 1984.

[97] G. B. Whitham, *Linear and Nonlinear Waves*. Wiley, 1974.

[98] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. Springer, 1999.

[99] M. Ahmed and K. DeJong, "Function approximator design using genetic algorithms," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, 1997, pp. 519–524. DOI: 10.1109/ICEC.1997.592365.

[100] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[101] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[102] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.

[103] J. A. Dietz and T. R. Morris, "Spectral methods in asymptotically safe quantum gravity," *JHEP*, vol. 05, p. 109, 2013. DOI: 10.1007/JHEP05(2013)109.

[104] N. Christiansen, K. Falls, J. M. Pawlowski, and M. Reichert, "Fixed-functionals in asymptotically safe gravity," *Phys. Rev. D*, vol. 92, 2015. DOI: 10.1103/PhysRevD.92.124020.

[105] B. Knorr and F. Saueressig, "Spectral flows in asymptotically safe quantum gravity," *Phys. Rev. Lett.*, vol. 121, 2018. DOI: 10.1103/PhysRevLett.121.161304.

[106] Y. B. et al., "Curriculum learning," *ICML*, 2009.

[107] A. A. R. et al., "Progressive neural networks," *arXiv:1606.04671*, 2016.

[108] L. N. Smith, "Cyclical learning rates for training neural networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. DOI: 10.1109/WACV.2017.58.

[109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

Appendix A

Pseudocode for the testing suite

Algorithm 1: Testing Suite (control parameters for genetic algorithm)

Data: *Theory parameters:* an array P_T containing:
the beta functions β^n,
the fixed point g_*^n,
the true result $v^\star$,
the linear generating function F_μ^* and its derivative $\partial_n F_\mu^*$.
Meta parameters: an array P_M containing for each of the following an array of choices:
the number of collocation points in each direction $n_p = (n_{pu-}, n_{pu+}, n_{pv-}, n_{pv+})$,
the bounds of the collocation point grid $B_p = (B_{pu-}, B_{pu+}, B_{pv-}, B_{pv+})$,
the basis functions ψ^i and their derivatives $\partial_n \psi^i$,
a Boolean b_f whether to fix parameters or not,
the population size coefficient c_P.
Control parameters: an array P_C containing for each of the following an array of choices:
the fitness function f,
the selection operator $\hat{S}$,
the mutation operator $\hat{M}$,
the crossover operator $\hat{C}$,
the number of generations I.
Result: Pareto frontier of parameter combinations
forall *combinations of* P_M **do**
    $CP \leftarrow$ generate collocation point grid **with** n_p, B_p;
    $N_p \leftarrow$ length of CP;
    $\sigma \leftarrow$ calculate smoothness factor **with** n_p, B_p;
    $\mathtt{F} \leftarrow$ precalculate F_μ^* **with** CP, g_*^n;
    $\mathtt{dF} \leftarrow$ precalculate $\partial_n F_\mu^*$ **with** CP, g_*^n;
    $\psi \leftarrow$ precalculate ψ^i **with** CP, σ;
    $\mathtt{d}\psi \leftarrow$ precalculate $d\psi^i$ **with** CP, σ;
    $\beta \leftarrow$ precalculate β^n **with** CP;
    $P \leftarrow (c_P N_p) \times N_p$ tensor of unique random $r \in [-1, 1] \subset \mathbb{R}$;
    if b_f **then**
        $CP_{\text{core}} \subset CP$;
        foreach $p \in P$ **do**
            $p[CP_{\text{core}}] \leftarrow$ fix parameters **with** $p, CP_{core}, \mathtt{F}, \psi$;
        end
    end
    forall *combinations of* P_C **do**
        run algorithm **with** $P, P_C, \psi, \mathtt{d}\psi, \beta$;
        run performance metrics **with** $v^\star$;
        export metrics to file;
    end
end
find Pareto frontier **with** *performance metrics files*;

Appendix B

Full Pareto frontier

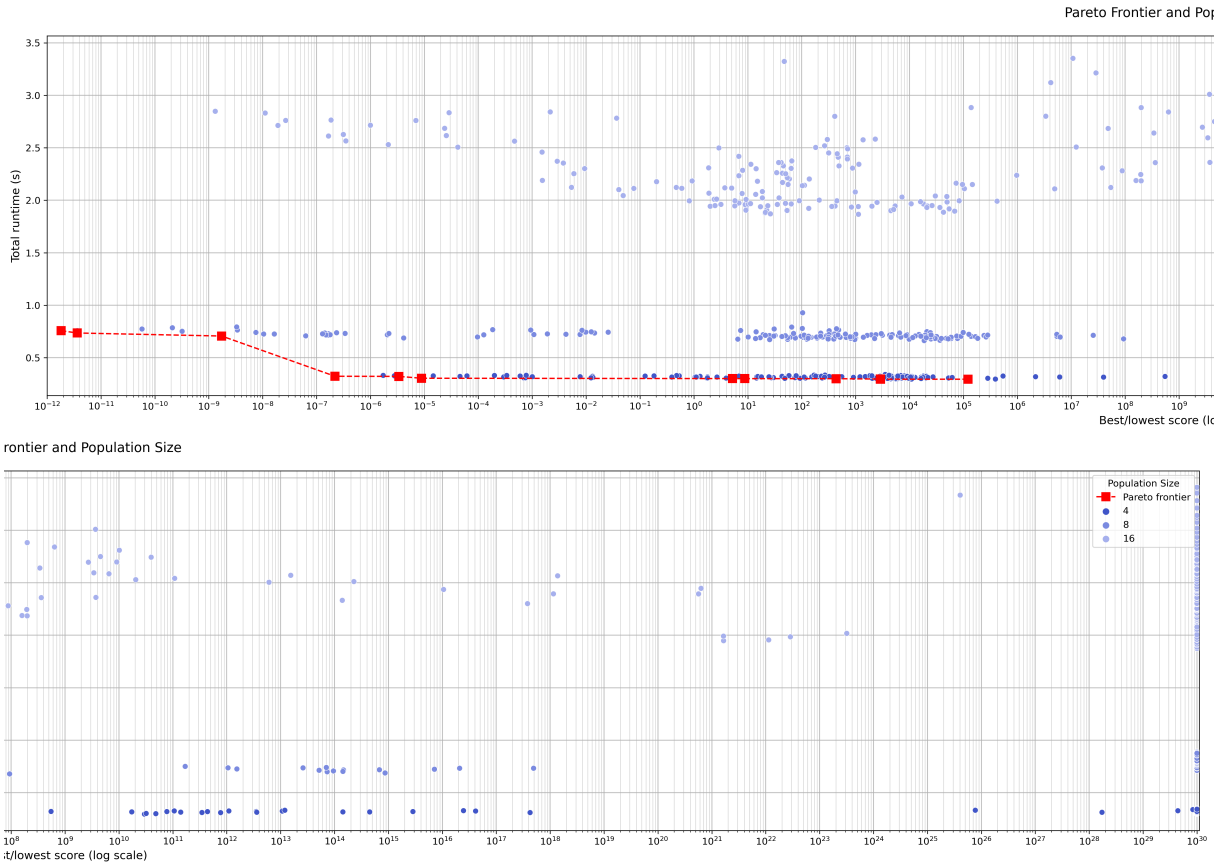


Figure B.1 Full Pareto frontier produced by the testing suite, split into two parts. All blue solutions are dominated by solutions on the frontier. No solutions with population size sixteen hold a presence on the frontier. Scores of solutions diverging above 10^{30} have been capped at 10^{30}.