University of Southampton

Faculty of Engineering and Physical Sciences
School of Physics and Astronomy

Machine Learning Methods for Exploring Beyond Standard Model Parameters

by

Mauricio Javier Ardiles Diaz
Masters in Theoretical Physics
ORCiD: 0009-0006-6640-9273

A thesis for the degree of
Doctor of Philosophy

2nd June 2025

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Physics and Astronomy

Doctor of Philosophy

Machine Learning Methods for Exploring Beyond Standard Model Parameters

by Mauricio Javier Ardiles Diaz

Physics aims to describe natural phenomena through the construction of theoretical models that capture essential system behaviours, with predictions tested against observations. In particle physics, the Standard Model (SM) has been a monumental success but remains incomplete, leaving unresolved challenges such as explaining neutrino masses, dark matter, the hierarchy problem, and gravity. Moreover, recent experimental anomalies, including results from scalar searches, hint at new physics and motivate the exploration of Beyond the SM (BSM) scenarios.

However, performing phenomenological studies in BSM models poses two major challenges. First, the number of possible models is immense. Second, within a single model, the parameter space is characterised by high dimensionality, sparsity of feasible configurations, and the computational cost of numerical evaluations, necessitating advanced parameter scan algorithms.

This thesis introduces a new formulation for parameter scan algorithms based on an active search methodology. This approach leverages Machine Learning (ML) modelling and sequential decision-making techniques, borrowing concepts from Bayesian Optimisation. A new, sample-efficient parameter scan algorithm, called b-CASTOR, is proposed. Additionally, a Python library named `hep-aid` is presented, designed for the easy use, integration, and development of parameter scan algorithms in phenomenological studies. Finally, a Reinforcement Learning formulation for parameter space scans is reviewed. While this approach yielded negative results, the insights and limitations derived from the project are discussed.

This thesis aims to advance the development of ML-based parameter scan algorithms, addressing computational challenges and laying the foundations for a systematic exploration of BSM models.

Contents

List of Figures

List of Tables

Acknowledgements

First, I want to thank my supervisors, Stefano Moretti and Srinandan Dasmahapatra. They let me explore topics I was interested in and trusted me throughout this journey. They guided me when I felt lost and supported me when I needed it most. For that, I am truly grateful.

I also want to thank my family. Living far away from them during this PhD has been one of the hardest parts of this journey. I could only visit them once, but a big part of my heart has always been in Chile with them. They have always believed in me and supported me unconditionally. I will always be grateful to them for helping me pursue my dreams.

To my friends, both old and new. To Michel, Stefy, and Seba back in Chile. And to the amazing colleagues and friends I've made during this PhD—Giorgio, Jacan, Raj, Ahmed, Alessandro, Vlad, and Pratyusha—for the long breaks and no work. I wish you all the best in everything ahead. And my friends from 9A! the first year of this PhD was the best because of you, Toru, Yichi, Dan, Tim, Clara, Pong and Adrian to a bit.

Lastly, to Kiku. I was so shy to leave my room the first day you arrived at City Gateway. Little did I know, I was about to meet the love of my life. Over these past four years, you have supported me and brought so much happiness to every moment.

Part I

Introduction, Background, and Problem Definition

Chapter 1

Introduction

Physics seeks to describe the natural world through simplified, abstract representations –*models*– that capture the essential behaviour of systems. The task of the physicist is to construct such models and test their predictions against experimental observations. If a prediction does not align with observation, the model must be ruled out, modified, or replaced with a new one, and the process begins again. In particle physics, these models represent fundamental particles and their interactions, serving as a bridge between theoretical frameworks and experimental observations.

The most successful outcome of such a model-building process is the Standard Model (SM) of particle physics. This historic achievement, alongside decades of advancements in the field, culminated in the discovery of the Higgs boson with a mass of 125 GeV at the Large Hadron Collider (LHC) by the ATLAS [3] and CMS [8] experiments in 2012. Despite its success, the SM is widely regarded as incomplete. It faces several observational challenges, such as the explanation of neutrino masses, the unresolved baryon-antibaryon asymmetry of the universe and the nature of dark matter and dark energy. Theoretical challenges also exist, including the hierarchy problem, explanation of the SM's family structure and the absence of a quantum description of gravity.

Moreover, hints of New Physics (NP), known as anomalies, have been slowly emerging [9, 10]. These anomalies span a broad energy range and arise from both precision measurements and direct experimental searches. They include flavour observables, the anomalous magnetic moment of the muon, the $W^\pm$ boson mass, and the possible existence of additional neutral spin-0 particles, the latter being the primary focus of this work. Notably, anomalous experimental signals around $\approx 95,$GeV have been reported in searches for new Higgs bosons. Various experimental analyses support this anomaly, including a $\gamma\gamma$ (di-photon) excess observed at CMS [11], a $\tau^+\tau^-$ (di-tau) excess also reported by CMS [12], and a $b\bar{b}$ excess detected by LEP [13].

Such anomalies can be addressed by Beyond the Standard Model (BSM) scenarios, motivated by the need to explain these phenomena either individually or collectively.

Specifically, focusing on theoretical constructions that could account for the aforementioned 95 GeV anomalies, several BSM scenarios have been proposed [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. Among these frameworks, we focus on the $(B-L)$ Supersymmetric Standard Model ($(B-L)$SSM), a non-minimal realisation of Supersymmetry. This model not only addresses many of the SM's unresolved issues [36] but also provides a potential explanation for the $\gamma\gamma$ anomaly at 95 GeV [37]. From a physics perspective, our goal is to determine whether this BSM scenario can also account for the $\tau^+\tau^-$ and $b\bar{b}$ excesses. Within BSM phenomenology such an endeavour is known the Parameter Scan (PS) problem [38].

The Parameter Scan Problem (PSP) involves a systematic exploration of the multi-dimensional parameter space of a new physics scenario. This process includes calculating numerical values for model predictions across various points in its parameter space, applying experimental and theoretical constraints, and identifying satisfactory regions that can explain multiple phenomena. The satisfactory regions in the parameter space are found by checking whether a theoretical prediction matches within some error margin measured features of anomalous data or respects exclusion limits if no BSM observations have been made. However, parameter scan (PS) methods must address several computational challenges, including:

- **Sparsity and Disconnected Regions**: The regions in the parameter space of a BSM scenario that can accommodate a combination of experimental results are often sparse and possibly disconnected.
- **High Dimensionality**: The parameter space of these models is typically high-dimensional, covering a wide range of in each dimension.
- **Computational Cost**: Numerically evaluating a single configuration of a BSM model using standard High Energy Physics (HEP) software tools is computationally expensive.

Therefore, selecting a PS method suitable for a specific phenomenological study is not trivial. Expert knowledge of the BSM model and its computational demands must be evaluated to perform a successful phenomenological study.

The PSP is commonly framed as a sampling problem [39], often addressed using Bayesian inference techniques [40]. One widely used approach is Markov-Chain Monte Carlo (MCMC) [41] methods, which are employed to estimate probability density functions across the parameter space. Extensions to MCMC, such as the Nested Sampling method [42], are also highly popular in the High Energy Physics (HEP) community. While MCMC methods and their variations are effective for probabilistic inference, model fitting, and have been widely adopted in the physical sciences [39], they still face the challenges of sparsity, high dimensionality, and computational cost in the context of BSM parameter space exploration.

Recent advances have explored the integration of Machine Learning (ML) [43, 44] methods into PS techniques offering a promising approach to addressing challenges related to efficiency and scalability. Artificial Neural Network (ANNs) based methods [5, 45] have been proposed, adopting different formulations, such as *regression* and *classification*. For regression, the physical observables are learned in an incremental manner, while for classification the viability of a parameter space configuration is treated as a label. Further, in both cases the learned model is incorporated into a policy to perform an informed sampling strategy. An alternative approach was developed in [46, 47], using Active Learning (AL) practices to train a Neural Network (NN) discriminator. The primary aim of this approach is to incrementally learn the decision boundary in regions of the parameter space where the model is allowed. Although NN based methods offer diversity in the search space configurations leading to an informative characterisation of the satisfactory regions, they require large datasets to achieve high accuracy, which can be challenging when dealing with computationally expensive HEP toolbox. This sample efficiency problem was noted in [48], where an alternative strategy was employed. The PSP is re-framed as a single-objective Black-Box Optimisation (BBO) problem, employing various optimisation methods, including Evolutionary Algorithms (EA) [49, 50] and Bayesian Optimisation (BO) [51]. BO is a model based approach for optimising black-box functions that are computationally expensive to evaluate. It builds a surrogate model, typically a Gaussian process (GP) [52], to approximate the objective function and uses an acquisition function to decide where to sample next. By balancing exploration and exploitation, BO aims to find the global optimum with the least possible function evaluations.

Practically, PS methods either from the sampling approach or the ML based approach need to be implemented in a computational framework to be used by the community. Several PS and sampling libraries have been developed for phenomenology, including `BSM Toolbox` [53], `xBit` [54], `EasyScan_HEP` [55], and `BSMart` [47]. Each of these libraries addresses the PS problem with unique software designs and specific usage goals. They share common features such as integration with a set of HEP packages, implementation of various PS algorithms, and the use of configuration files to simplify setup. These tools provide the community with a range of resources tailored to different applications.

The research projects in this thesis aim to: first, advance the development of ML-based approaches in the existing literature by proposing a PS method that improves sample efficiency, particularly in cases where numerical evaluations of BSM models (with the aforementioned $(B-L)$SSM as our benchmark example) are computationally expensive; second, foster innovation in PS algorithm development through the introduction of a novel computational framework; and finally, provide insights and lessons learned from addressing the PSP using different ML approaches such as formulating the PSP as a Reinforcement Learning (RL) problem.

This chapter on this thesis are structured as follows:

1. **The Standard Model and Beyond**: The theoretical foundations of particle physics are introduced, starting with an overview of the SM. This includes an description of the SM Lagrangian and its fields. A brief discussion follows on the theoretical and experimental aspects of the Higgs boson. The challenges faced by the SM, along with anomalies hinting at new physics, are then reviewed. Subsequently, various BSM scenarios proposed to address these challenges are examined, to give a sense of the model space, finishing with the introduction of Supersymmetry and the $(B - L)$SSM.

2. **Automated BSM Phenomenology**: It begins with an overview of the HEP software ecosystem commonly used in phenomenological studies. The PSP is then defined, along with the computational challenges inherent in conducting parameter scans, including issues such as high computational cost, multiple constraints, and high dimensionality. This chapter, highlights the primary research question that this thesis seeks to address, setting the stage for the developments described in the later chapters.

3. **Data Modelling**: An in-depth exploration of ML techniques for regression is introduced, specifically focusing on how to perform regression given a dataset. This includes the use of GPs, which learn a distribution over a family of functions that can describe the dataset, and ANNs, which utilise deterministic mappings. The discussion builds from linear regression to MLPs.

4. **Sampling and Decision Making**: A review of a small set of Monte Carlo sampling algorithms that are well-established in particle physics research and utilised in this thesis, specifically Rejection Sampling and the MCMC Metropolis-Hastings (MH). Following this, decision-making frameworks are introduced, including BO, which plays a crucial role in the new PS methods proposed in this work, and Reinforcement Learning (RL), a behavioural framework for optimisation and exploration.

5. **Space Exploration with Reinforcement Learning**: This chapter explores the application of RL for the PSP. Although the results were unsatisfactory due to RL's sample-intensive nature and the difficulty in creating an effective reward function, this exploration highlights key limitations and challenges of this formulation. The insights gained significantly influenced the thesis's direction, guiding the development of its primary contributions.

6. **Bayesian Active Search on Parameter Space**: The PSP is formulated as an Active Search (AS) Problem which borrows the key elements from BO by shifting the task to search of a class of points than optimisation. In this approach, multiple phenomenological signatures of a particular BSM model are set as the multiple objectives, constrained by experimental measurements. These can refer to particle masses, Branching Ratios (BRs), production cross-sections or any model prediction information. We introduce a novel ML approach, named b-CASTOR, to efficiently scan the parameter space. This method uses GP surrogates to approximate multiple objectives constrained by experimental measurements and employs a volume-based acquisition function to ensure a comprehensive characterisation of the satisfactory region in the parameter space. Our method outperforms traditional competing algorithms, such as those based on MCMC methods, offering a more effective strategy for exploring parameter spaces in computationally expensive BSM scenarios.

7. **`hep-aid`: A new computational framework**: This chapter presents `hep-aid`, a modular Python library designed for utilising, implementing, and developing PS algorithms. Originally conceived for sample-efficient, multi-objective AS approaches, it has proven to be versatile for implementing a variety of PS methodologies. The library currently integrates three ML-based approaches: Constraint Active Search (CAS) algorithm, a point-wise multi-objective AS method, the proposed b-CASTOR algorithm and an NN-based PS algorithm known as MLScan, which leverages Neural Networks. The library's modules and functionalities are designed to be easily extensible and compatible with other external software used in phenomenology, fostering adaptability and innovation in PS exploration.

Finally, the search for a definitive BSM model involves two primary search spaces: the *model space*–an ever-growing and vast number of BSM scenarios proposed by researchers–and the *parameter space* of each model. Each specific parameter configuration within a single BSM scenario constitutes a distinct model, offering rich and potentially unique phenomenology. This can result in vastly different predictions even within the same BSM scenario. The primary results of this thesis advance the methodologies for exploring the second type of space, the *parameter space*. However, these methods have been developed with a long-term vision of creating methodologies for the automated exploration of the first type of space, the *model space*.

Chapter 2

The Standard Model and Beyond

2.1 The Standard Model of Particle Physics

The ATLAS [56] and CMS [57] Collaborations at CERN announced the observation of a Higgs boson with a mass of approximately 125 GeV in 2012. This discovery was a pivotal milestone for the now strongly established SM [58] of particle physics, a quantum field theory (QFT) that provides a comprehensive framework for understanding the universe at its most fundamental level by describing elementary particles and their interactions, illustrated in Figure 2.1.

The SM is formed by the combination of electroweak theory [59, 60, 58] and Quantum Chromodynamics (QCD) [61, 62, 63]. Constructed using the principles of QFT, the SM describes particles as excitations of underlying fields that exist and interact according to a Lagrangian that is invariant under the gauge symmetry group,

$$G_{\mathrm{SM}} = \mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \tag{2.1}$$

The electroweak theory describes the electromagnetic and weak interactions between quarks and leptons and is based on the gauge symmetry group $\mathrm{SU}(2)_{\mathrm{L}} \times \mathrm{U}(1)_{\mathrm{Y}}$, which corresponds to weak left-handed isospin and hypercharge, respectively. QCD, on the other hand, is the theory of the strong interaction between coloured quarks and is based on the gauge symmetry group $\mathrm{SU}(3)_{\mathrm{C}}$.

A key component of the SM is the Electroweak Symmetry Breaking (EWSB) mechanism [64, 65, 66]. This mechanism provides mass to the $W^{\pm}$ and Z gauge bosons while keeping the photon massless, preserving the unbroken $\mathrm{U}(1)_{\mathrm{em}}$ symmetry. Additionally, through their Yukawa interactions with the Higgs field, EWSB enables the SM to generate masses for quarks and charged leptons, though neutrinos remain massless within the SM.

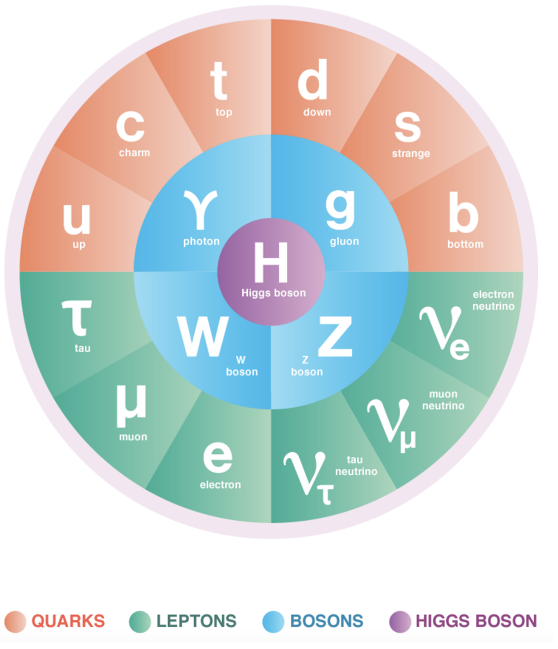


FIGURE 2.1: Diagram of the constituents of the Standard Model of particle physics taken from [1], illustrating the fundamental particles that form the atoms, which in turn make up the chemical elements, the universe, and ourselves. The quarks (red) and leptons (green) make up matter, while the bosons (blue) mediate the fundamental forces, with the Higgs boson (purple) responsible for mass generation through the Higgs mechanism.

2.1.1 The SM Lagrangian

The Lagrangian of the SM can be written as a sum of separate Lagrangian terms. Each term encapsulates the dynamics and interactions of a specific field sector, namely the gauge, fermion, Higgs, and Yukawa sectors. It is expressed as,

$$\mathcal{L}_{SM} = \mathcal{L}_G + \mathcal{L}_f + \mathcal{L}_{\text{Higgs}} + \mathcal{L}_Y \tag{2.2}$$

The first term describes the dynamics of the gauge fields, the force mediators,

$$\mathcal{L}_G = -\frac{1}{4} G^a_{\mu\nu} G^{\mu\nu}_a - \frac{1}{4} W^a_{\mu\nu} W^{\mu\nu}_a - \frac{1}{4} B_{\mu\nu} B^{\mu\nu} \tag{2.3}$$

where $G^a_{\mu\nu}$, $W^a_{\mu\nu}$, and $B_{\mu\nu}$ are the field strength tensors for the $\mathrm{SU}(3)_C$, $\mathrm{SU}(2)_L$, and $\mathrm{U}(1)_Y$ gauge groups, respectively. The index a runs over the generators of the respective gauge groups, while μ and ν are spacetime indices. The second term in equation (2.2) contains the kinetic terms and gauge interactions of the fermions, the fields describing matter, which is given by

$$\mathcal{L}_f = i \sum_f \bar{f}^i \not{D} f^i \tag{2.4}$$

where f represents the set of fermions $f = \{q, l, u_R, d_R, e_R\}$ with q and l being the left-handed quark and lepton doublets, and u_R, d_R, e_R being the right-handed singlets. The indices i and j label the three generations of fermions, known as flavour indices. The Higgs field is a scalar field responsible for giving mass to particles through spontaneous symmetry breaking, with its dynamics and potential described the third term in equation (2.2) and is given by,

$$\mathcal{L}_{\text{Higgs}} = (D_\mu \Phi)^\dagger (D^\mu \Phi) - V(\Phi), \tag{2.5}$$

The fermions acquire mass through their interactions with the Higgs field, described by the Yukawa terms corresponding to the last Lagrangian term in equation (2.2),

$$\mathcal{L}_Y = -Y^e_{ij} \bar{L}^i e^j_R \Phi - Y^u_{ij} \bar{Q}^i u^j_R \widetilde{\Phi} - Y^d_{ij} \bar{Q}^i d^j_R \Phi + \text{ h.c.} \tag{2.6}$$

where Y^e_{ij}, Y^u_{ij}, Y^d_{ij} are the Yukawa coupling matrices, and $\widetilde{\Phi} = i\tau_2 \Phi^*$ is the conjugate Higgs doublet.

From the Lagrangian, the interaction terms between particles determine the Feynman rules of the model. These rules determine the interaction vertices, propagators, and external states, allowing the systematic calculation of scattering amplitudes. These amplitudes describe the probability for specific interaction processes, such as particle production, decay, or scattering events. This quantity forms the foundation for computing physical observables that can be directly compared with experimental measurements. Key observables include:

- **Cross-sections**: Quantify the likelihood of specific particle production processes.
- **Decay widths**: Measure the rates of particle decays.
- **Branching ratios**: Indicate the fraction of decays proceeding through specific channels.
- **Kinematic observables**: These include invariant mass and angular distributions, which are reconstructed from the momenta of decay products.

2.1.2 Matter Fields

To describe matter, the SM includes three generations of quarks and leptons, each consisting of left-handed and right-handed chiral components. The left-handed fermions transform as weak isospin doublets under $\mathrm{SU}(2)_L$, while the right-handed fermions are weak isospin singlets,

$$\begin{aligned}
L_1 &= \begin{pmatrix} \nu_e \\ e^- \end{pmatrix}_L, e_{R_1} = e_R^-, Q_1 = \begin{pmatrix} u \\ d \end{pmatrix}_L, u_{R_1} = u_R, d_{R_1} = d_R \\
L_2 &= \begin{pmatrix} \nu_\mu \\ \mu^- \end{pmatrix}_L, e_{R_2} = \mu_R^-, Q_2 = \begin{pmatrix} c \\ s \end{pmatrix}_L, u_{R_2} = c_R, d_{R_2} = s_R \\
L_3 &= \begin{pmatrix} \nu_\tau \\ \tau^- \end{pmatrix}_L, e_{R_3} = \tau_R^-, Q_3 = \begin{pmatrix} t \\ b \end{pmatrix}_L, u_{R_3} = t_R, d_{R_3} = b_R
\end{aligned} \tag{2.7}$$

The third component of the weak isospin, I_f^3, takes values $I_f^3 = \pm\frac{1}{2}$ for the upper and lower components of the left-handed doublets, respectively, and $I_f^3 = 0$ for the right-handed singlets. The hypercharge Y_f is then defined for each fermion by the relation,

$$Y_f = 2Q_f - 2I_f^3, \tag{2.8}$$

where Q_f is the electric charge. This assignment ensures that the total hypercharge and electric charge of all fermions satisfy,

$$\sum_f Y_f = \sum_f Q_f = 0 \tag{2.9}$$

which guarantees the cancellation of gauge anomalies [67]. Anomaly cancellation is crucial for the mathematical consistency and renormalisability of the theory [68].

2.1.3 Gauge Fields

The interactions between matter fields in the SM are mediated by gauge fields, which correspond to spin-one bosons. The field B_μ , associated with the generator Y of the $\mathrm{U(1)_Y}$ group, couples to the hypercharge of matter fields, contributing to both electromagnetic and weak interactions. Similarly, the fields $W_\mu^{1,2,3}$, associated with the generators $T^a (a = 1, 2, 3)$ of the $\mathrm{SU(2)_L}$ group, couple to the weak isospin of matter fields, mediating weak interactions. In the strong interaction sector, there is an octet of gluon fields $G_\mu^{1,\dots,8}$, which correspond to the eight generators of the $\mathrm{SU(3)_C}$ group. This correspondence, a manifestation of the gauge principle, is a cornerstone of gauge theories [69], where the symmetry structure dictates the interactions between matter and gauge fields.

The generators of the $\mathrm{SU(2)_L}$ group are half of the Pauli matrices, satisfying the commutation relations:

$$\left[T^a, T^b\right] = i\epsilon^{abc}T^c, \tag{2.10}$$

where ϵ^{abc} is the antisymmetric tensor, and the indices $a, b, c = 1, 2, 3$ run over the three generators of $\mathrm{SU(2)_L}$. Similarly, the $\mathrm{SU(3)_C}$ group has eight generators, which correspond to half of the eight traceless, Hermitian Gell-Mann matrices that satisfy the algebra,

$$\left[T^a, T^b\right] = i f^{abc} T^c, \tag{2.11}$$

where f^{abc} are the structure constants of $\mathrm{SU}(3)_\mathrm{C}$. The generators are normalised such that:

$$\mathrm{Tr}\left[T^a T^b\right] = \frac{1}{2}\delta^{ab}. \tag{2.12}$$

Here T^a is used for both $\mathrm{SU}(2)_\mathrm{L}$ and $\mathrm{SU}(3)_\mathrm{C}$ generators, as the context determines the relevant group. The dynamics of the gauge fields and their interactions are described by the field strength tensors, which appear in the gauge kinetic terms of the SM Lagrangian,

$$\begin{aligned} G^a_{\mu\nu} &= \partial_\mu G^a_\nu - \partial_\nu G^a_\mu + g_s f^{abc} G^b_\mu G^c_\nu \\ W^a_{\mu\nu} &= \partial_\mu W^a_\nu - \partial_\nu W^a_\mu + g_2 \epsilon^{abc} W^b_\mu W^c_\nu \\ B_{\mu\nu} &= \partial_\mu B_\nu - \partial_\nu B_\mu \end{aligned} \tag{2.13}$$

where g_s, g_2 and g_1 are, respectively, the coupling constants of $\mathrm{SU}(3)_\mathrm{C}, \mathrm{SU}(2)_\mathrm{L}$ and $\mathrm{U}(1)_\mathrm{Y}$. The non-abelian field strength tensors $G^a_{\mu\nu}$ and $W^a_{\mu\nu}$ include terms proportional to their structure constants, which lead to triple and quartic gauge boson couplings, corresponding to the self-interactions of the gauge bosons.

Finally, the interaction between gauge bosons and matter fields is defined by the covariant derivative,

$$D_\mu \psi = \left(\partial_\mu - i g_s T_a G^a_\mu - i g_2 T_a W^a_\mu - i g_1 \frac{Y_q}{2} B_\mu\right)\psi. \tag{2.14}$$

For quarks, this is the full covariant derivative, as quarks are charged under all three gauge groups. For leptons, the term for $\mathrm{SU}(3)_C$ is absent, as leptons are not colour-charged.

2.1.4 The Higgs sector

The EWSB mechanism allows the SM to generate masses for the $W^\pm$ and Z gauge bosons, quarks, and charged leptons, while preserving the described theoretical framework for fermions and gauge fields. To achieve this, a self-interacting complex scalar field Φ whith hypercharge $Y_\phi = +1$, transforming as a doublet under $\mathrm{SU}(2)_\mathrm{L}$, is introduced,

$$\Phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} \tag{2.15}$$

Then, a scalar potential is defined as,

$$V(\Phi) = \mu^2 \Phi^\dagger \Phi + \lambda \left(\Phi^\dagger \Phi\right)^2 \tag{2.16}$$

When $\mu^2 < 0$, the neutral component of Φ develops a vacuum expectation value (vev), while the charged component remains zero to preserve $\mathrm{U}(1)_{\mathrm{em}}$,

$$\langle\Phi\rangle_0 = \begin{pmatrix} 0 \\ \frac{v}{\sqrt{2}} \end{pmatrix} \text{ with } v = \left(-\frac{\mu^2}{\lambda}\right)^{1/2} \tag{2.17}$$

This induces the spontaneous breaking of SM gauge symmmetry

$$\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \to \mathrm{SU}(3)_C \times \mathrm{U}(1)_{\mathrm{em}} \tag{2.18}$$

The global minimum of the potential defines the ground state of the theory. Spontaneous symmetry breaking implies that the vacuum does not respect the full symmetry of the Lagrangian.

The Higgs field couples to the W_μ and B_μ gauge fields via the covariant derivative in the Higgs Lagrangian, as defined in equation (2.5). The covariant derivative, specified in equation (2.14), excludes the $\mathrm{SU}(3)_C$ term, as the Higgs field is colour-neutral. After symmetry breaking, the neutral and charged Goldstone bosons, corresponding to the broken generators, mix with the gauge fields and provide the longitudinal components of the $W^\pm$ and Z bosons in the unitarity gauge. These bosons acquire masses,

$$m_W^2 = \frac{g_2^2 v^2}{4}, \quad m_Z^2 = \frac{\left(g_1^2 + g_2^2\right) v^2}{4}. \tag{2.19}$$

A generator of the gauge symmetry remains unbroken, corresponding to the $\mathrm{U}(1)_{\mathrm{em}}$ gauge symmetry. Its associated gauge field, the photon, remains massless. Similarly, the eight colour gauge bosons, the gluons, remain massless, as they correspond to the conserved $\mathrm{SU}(3)_C$ gauge symmetry with its eight unbroken generators.

The fermions of the SM acquire mass through renormalisable and gauge-invariant interactions between the Higgs field and the fermions, known as Yukawa interactions. These interactions are described by the Lagrangian defined in equation 2.6. The Yukawa coupling matrices encode the strength of the interactions between the Higgs field and the fermions. When the Higgs field acquires a vev, the Yukawa terms generate mass matrices,

$$M_{ij}^e = \frac{Y_{ij}^e v}{\sqrt{2}}, \quad M_{ij}^u = \frac{Y_{ij}^u v}{\sqrt{2}}, \quad M_{ij}^d = \frac{Y_{ij}^d v}{\sqrt{2}}. \tag{2.20}$$

These mass matrices, however, correspond to the gauge eigenstates of the fermions. Diagonalising the Yukawa-generated mass matrices through unitary transformations leads to the physical masses. For the quark sector, the mass matrices M^u and M^d are diagonalised separately by the unitary matrices $V_L^u, V_R^u, V_L^d, V_R^d$, giving,

$$\left(V_L^u\right)^\dagger M^u V_R^u = \mathrm{diag}\left(m_u, m_c, m_t\right), \quad \left(V_L^d\right)^\dagger M^d V_R^d = \mathrm{diag}\left(m_d, m_s, m_b\right). \tag{2.21}$$

ensuring that the mass terms in the Lagrangian correspond to quarks with definite masses. While the right-handed quarks (u_R, d_R) do not participate in weak interactions, the left-handed quarks (u_L, d_L) do through the weak interaction Lagrangian. This involves the matrix structure $V = (V_L^u)^\dagger V_L^d$, known as the Cabibbo-Kobayashi-Maskawa (CKM) matrix, which governs quark flavour mixing and introduces phenomena such as CP violation in the weak interactions CITE.

For leptons, the Yukawa-generated mass matrix M^e is diagonalised, simiraly as equation (2.21), with the unitary matrices V_L^e and V_R^e, leading to

$$(V_L^e)^\dagger M^e V_R^e = \operatorname{diag}(m_e, m_\mu, m_\tau) \tag{2.22}$$

In the SM, neutrinos remain massless, as no Yukawa terms are introduced for them.

2.2 Higgs physics

The Higgs boson is characterised by the cross sections of its production processes and the branching fractions of its decays. At the LHC [2, 70], the dominant production mode is gluon-gluon fusion (ggF), which involves two gluons fusing through a top-quark loop to produce a Higgs boson. The next most significant mode is vector boson fusion (VBF), where two quarks interact via the exchange of W or Z bosons, which subsequently fuse to generate the Higgs boson. Other production modes include associated production with a vector boson, commonly referred to as Higgsstrahlung (WH, ZH), where the Higgs boson is produced alongside a W or Z boson, and associated production with top or bottom quarks (tH,ttH, bbH). The Feynman diagrams for the production modes are ilustrated in Figure 2.2. The produced Higgs boson rapidly decays into various final states, with the leading decay modes including a pair of fermions (e.g., $b\bar{b}$ or $\tau^+\tau^-$), a pair of heavy bosons (e.g., WW^* or ZZ^*), or a pair of photons $(\gamma\gamma)$, shown in Figure 2.3. Additionally, it can decay into gluon pairs (gg) via a loop-mediated process (the time reversal of diagram (a) in Figure 2.2) or into rarer channels such as γZ, as illustrated in Figure 2.3.

A key tool in this discovery was the utilisation of the *signal strength* parameter, denoted by μ. This parameter quantifies the level of agreement between observed signal yields and the expectations from the SM. The signal strength μ is defined as the ratio of the observed rate of a specific process to the rate predicted by the SM. More formally, for a production mode i and a decay channel f, the signal strength is expressed as,

$$\mu_i^f = \frac{\sigma_i^f}{\sigma_{i,\mathrm{SM}}^f}, \tag{2.23}$$

Higgs boson production modes

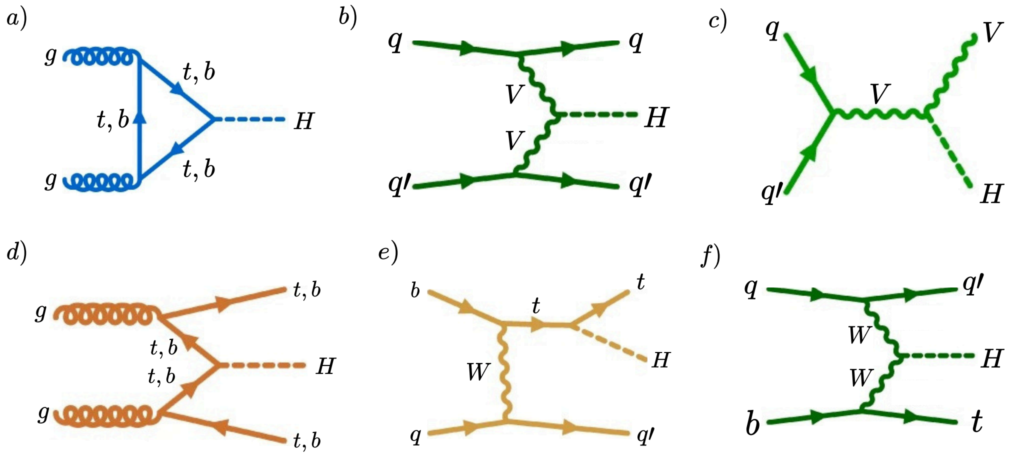


FIGURE 2.2: Higgs boson production mechanisms: (a) gluon-gluon fusion (**ggF**), (b) vector boson fusion (**VBF**), (c) associated production with a W or Z boson (**WH, ZH**), (d) production with a top or bottom quark pair (**ttH**, **bbH**), and (e, f) production with a single top quark (**tH**).

Higgs boson decay channels

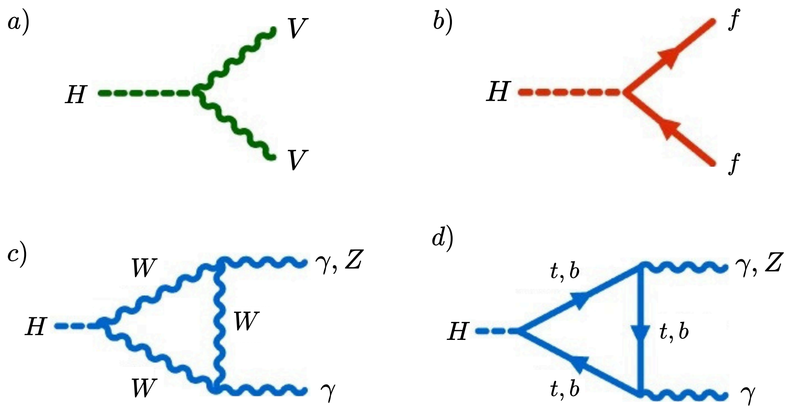


FIGURE 2.3: Higgs boson decay channels: (a) into heavy vector boson pairs, (b) into fermion-antifermion pairs, and (c, d) into photon pairs or $Z\gamma$.

where σ_i^f is the observed cross section for the process $i \to H \to f$, and $\sigma_{i,\mathrm{SM}}^f$ is the corresponding SM prediction. The observed cross section σ_i^f can be factorised into the product of the production cross section (σ_i) and the branching fraction (B^f) for the decay mode f,

$$\mu_i^f = \frac{\sigma_i \cdot \mathcal{B}^f}{\sigma_{i,\mathrm{SM}} \cdot \mathcal{B}_{\mathrm{SM}}^f}. \tag{2.24}$$

This factorisation is valid under the narrow-width approximation (NWA) [71], which assumes that the intrinsic width (Γ_H) of the Higgs boson is much smaller than its mass (m_H). Then, the individual signal strength for production and decay can be defined as $\mu_i = \sigma_i / \left(\sigma_i\right)_{\mathrm{SM}}$ and $\mu^f = \mathcal{B}^f / \left(\mathcal{B}^f\right)_{\mathrm{SM}}$.

In the CMS, at the time of discovery the combined μ was found to be 0.87 ± 0.23, currently, $\mu = 1.002 \pm 0.057$ [2], showing excellent agreement with the SM expectation.

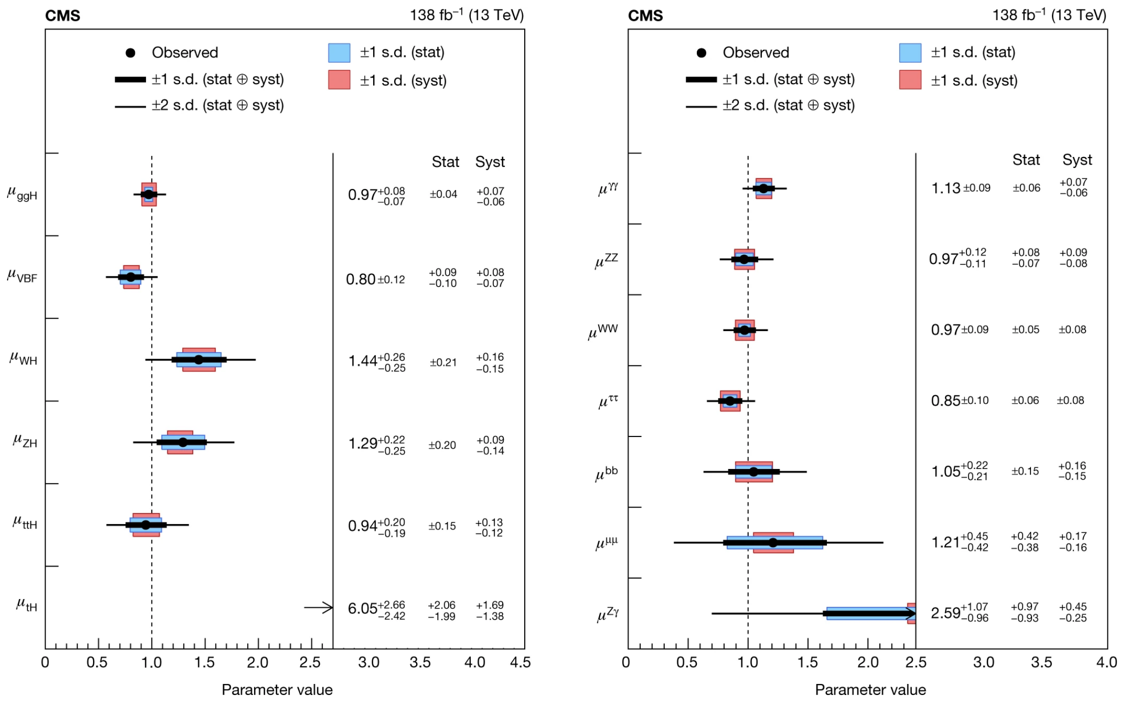


FIGURE 2.4: Signal strength parameters (μ) for various Higgs boson production modes (left) and decay channels (right) from CMS with $138 fb^{-1}$ of data at 13 TeV. The dashed line indicates the Standard Model prediction ($\mu = 1$), with statistical ($\pm 1\sigma$, blue) and systematic ($\pm 1\sigma$, red) uncertainties shown, alongside combined ($\pm 2\sigma$) confidence intervals in black. Image obtained from [2].

Figure 2.4 illustrates the signal strength parameters (μ) for various Higgs boson production modes (left) and decay channels (right), as measured by the CMS collaboration using $138\,\text{fb}^{-1}$ of proton-proton collision data at a centre-of-mass energy of 13 TeV. A value of $\mu = 1$, indicated by the vertical dashed line, corresponds to perfect agreement with the SM prediction. The left plot shows μ values for different production processes (as shown in Figure 2.2), while the right plot displays μ values for various decay processes (as shown in Figure 2.3). The confidence intervals for each μ value are represented as error bars: blue and red bands indicate $\pm 1\sigma$ statistical and systematic uncertainties, respectively, and black lines denote the combined $\pm 2\sigma$ intervals.

On the other hand, Figure 2.5 shows the signal strength modifier (μ) measured by the ATLAS experiment in 2012. The reported combined result of $\mu = 1.4 \pm 0.3$ represents a combination of the channels displayed in the figure. Currently, the measured value stands at $\mu = 1.05 \pm 0.06$ [70].

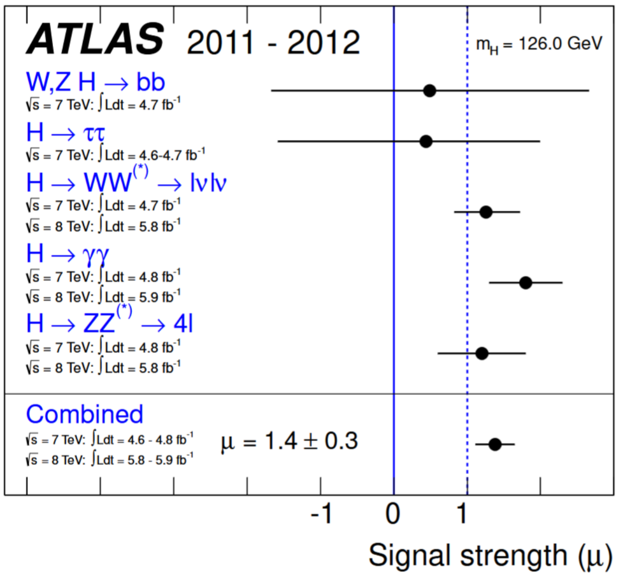


FIGURE 2.5: The signal strength modifier (μ) measured by the ATLAS experiment for a Higgs boson mass of 126 GeV, utilising data from 2011-2012. Individual measurements are shown for various Higgs decay channels, along with the combined result ($\mu = 1.4 \pm 0.3$). The vertical dashed line indicates the Standard Model prediction of $\mu = 1$. Image obtained from [3].

Measurements from both CMS and ATLAS show strong consistency with SM predictions for a Higgs boson with a mass around 125 GeV. Crucially, the discovery was confirmed with a statistical significance of approximately 5σ, signifying the extremely low probability of the observed excess of events at 125 GeV arising from random fluctuations. This level of significance provided compelling evidence for the existence of the Higgs boson.

2.3 Challenges and New Physics

The SM has long been the cornerstone of our understanding of fundamental particles and their interactions. While it provides a robust framework for describing the behaviour of known particles, several experimental and theoretical challenges remain unresolved. These include the discovery of non-zero neutrino masses, the mysteries of dark matter and dark energy, hierarchy problems associated with the Higgs boson, and the grand challenge of developing a unified theory that incorporates all fundamental forces, including gravity. Beyond these, anomalies observed in experimental data suggest potential deviations from the SM's predictions, hinting at the existence of New Physics (NP).

This section introduces these challenges, exploring the limitations of the SM and the evidence necessitating the construction of theoretical models that go beyond the SM, either by extending or modifying it, and even by expanding fundamental principles.

Neutrino masses

In the SM, neutrinos are assumed to be massless left-handed particles. However, studies of solar, atmospheric, reactor, and accelerator neutrinos have conclusively demonstrated that neutrinos undergo flavour oscillations, a phenomenon known as neutrino oscillations (for a review, see [72]). This behaviour is only possible if neutrinos have non-zero masses and different mass eigenstates. These observations challenge the original framework of the SM, necessitating extensions or modifications that can explain neutrino masses.

Neutrino oscillations arise due to a misalignment between the neutrino flavour eigenstates $(\nu_e, \nu_\mu, \nu_\tau)$, which interact with the weak force, and the mass eigenstates (ν_1, ν_2, ν_3), which have definite masses (m_1, m_2, m_3). These two bases are related by a unitary transformation described by the Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix U,

$$\nu_\alpha = \sum_{i=1}^{3} U_{\alpha i} \nu_i, \quad (\alpha = e, \mu, \tau), \tag{2.25}$$

here, $U_{\alpha i}$ are the elements of the PMNS matrix. As neutrinos propagate, the difference in masses between the eigenstates leads to phase differences, resulting in oscillations. The probability of a neutrino, which travels through vacuum, of flavour α being detected as a neutrino of flavour β at a distance L from its source is given by:

$$P\left(\nu_\alpha \to \nu_\beta\right) = \sum_{i,j} U_{\alpha i} U^*_{\beta i} U^*_{\alpha j} U_{\beta j} \exp\left[-i \frac{\Delta m^2_{ji}}{2} \frac{L}{E}\right] \tag{2.26}$$

where $\Delta m^2_{ij} = m^2_i - m^2_j$ is the mass-squared difference between the eigenstates, E is the energy of the neutrino.

Dark Matters

So far, the SM is responsible for explaining all the known elementary particles, including those that make up radiation and baryonic matter. However, these constitute a small fraction of the total composition of the universe. Astronomical observations and cosmological theory suggest that baryonic matter contributes only $\sim 5\%$ to the mass density of the universe. The main component is dark energy $\sim 69\%$ followed by dark matter with $\sim 25\%$. As illustrated in Figure 2.6.

Dark matter (DM) is the name for the problem of an invisible non-baryonic physical entity observed in the universe. Invisible, because it does not interact with the electromagnetic force, only with gravity. Evidence for the existence of DM comes from a wide range of astronomical scales, from individual galaxies to the entire universe [73]. The three main lines of observational evidence are: rotation curve measurements of

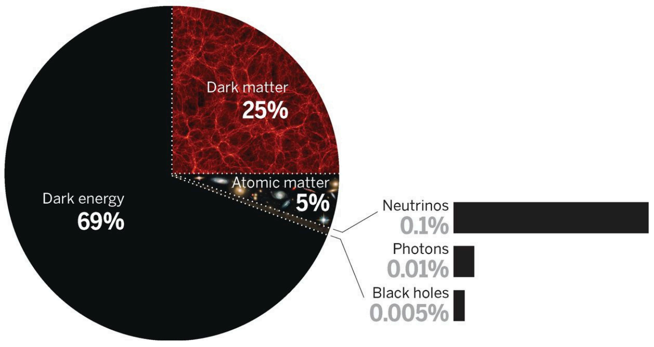


FIGURE 2.6: Universe composition: Dark energy makes up 69% of the universe's energy density, dark matter 25%, and atomic matter 5%. Minor components include neutrinos (0.1%), cosmic radiation (0.01%), and black holes (0.005%). Image adapted from [4].

individual spiral galaxies, which show that the outer regions of galaxies rotate much faster than can be explained by the visible mass of stars and gas; gravitational lensing and velocity dispersion of galaxies within clusters; and precise measurements of the Cosmic Microwave Background (CMB) anisotropies and the distribution of matter in the universe. All these observations demonstrate that the total observable matter is insufficient to account for the total mass of the specific physical systems, whether galaxies, clusters, or large-scale structures (LSS).

Moreover, the data can be explained by incorporating specific properties into DM models. These include: DM behaves like matter and as a non-relativistic fluid (cold); DM is non-interacting, meaning its interactions with itself or with ordinary matter are negligible; and DM is stable, implying it has been present since the early phases of the universe and has a lifetime greater than the age of the universe.

Dark energy, in contrast to dark matter, is an enigmatic force driving the accelerated expansion of the universe. Unlike matter, it is evenly distributed across space and does not form clusters. Its existence is supported by measurements of distant supernovae, which indicate an accelerating expansion rate. Further evidence comes from studies of the CMB and LSS. For a more extensive review, see [74]. Despite being the dominant component of the observable universe, the true nature of dark energy remains one of the greatest mysteries in modern physics.

Fine-tuning, Naturalness and Hierarchy problems

The hierarchy problem is a fundamental issue in QFT concerning the stability of the Higgs boson mass under radiative corrections. In the SM, the Higgs mass parameter receives quantum corrections from loop diagrams involving fermions, gauge bosons, and the Higgs itself. These corrections generically introduce quadratic divergences, making the Higgs mass highly sensitive to any new physics at high energy scales, such as the Planck scale ($M_P \sim 10^{19}$ GeV).

A conventional way to present the hierarchy problem is through loop-level calculations, where the leading quantum correction to the Higgs mass squared, using a momentum cut-off Λ, takes the form

$$\delta m_H^2 \sim \frac{\lambda}{16\pi^2}\Lambda^2, \tag{2.27}$$

This suggests that, unless finely tuned, the Higgs mass should naturally be of the order of Λ, which is many orders of magnitude larger than the observed electroweak scale ($\mathcal{O}(100)$ GeV). However, this formulation depends on the choice of regularisation. In particular, if dimensional regularisation is used, quadratic divergences formally disappear, as this method expresses divergences as poles in $\epsilon = 4 - d$, rather than explicit power-law divergences in Λ. While this avoids the problem in a technical sense, it does not resolve the fundamental issue: fine-tuning is still required to maintain a light Higgs boson.

More importantly, even if one regards the SM as an effective field theory valid up to a scale Λ, the hierarchy problem persists when new degrees of freedom appear at high energies. If, for instance, the SM is embedded in a more fundamental theory containing heavy scalars that couple to the Higgs [75], integrating them out generates additional corrections to the Higgs mass. Consider a scenario where a heavy scalar field S couples to the Higgs through the Lagrangian

$$\mathcal{L}_{\text{high}} \supset -\lambda_S |S|^2 |H|^2 - m_S^2 |S|^2, \tag{2.28}$$

When this heavy scalar is integrated out, the correction to the mass parameter is

$$\Delta\mu^2 = \frac{\lambda_S}{16\pi^2}\left[\Lambda^2 - 2m_S^2 \log\frac{\Lambda}{m_S}\right] + \dots \tag{2.29}$$

which explicitly demonstrates that even in a renormalisation scheme where quadratic divergences are absent, the Higgs mass remains sensitive to the scale m_S.

Thus, while dimensional regularisation alters the way divergences appear, it does not eliminate the underlying fine-tuning issue. Moreover, when new physics at high scales is included, threshold corrections from heavy fields reintroduce the problem, reinforcing the necessity of new physics beyond the SM to explain the smallness of the electroweak scale in a natural way. The later known as the naturalness problem.

In summary, fine-tuning, naturalness, and the hierarchy problem are all considered significant challenges because they undermine the principle of *separation of scales*. These problems are rooted in the Higgs mass's sensitivity to quantum corrections from high-energy scale physics [76], and motivate the search of new theories or mechanisms to address them.

Parameters

The SM, while highly predictive and experimentally validated, relies on a set of fundamental parameters that are not derived from the theory but must instead be determined through experiments and manually input into the framework. These parameters include particle masses, coupling constants, and mixing angles, leading to around 19. The SM offers no mechanism or theoretical insight to predict these values, they are external inputs rather than intrinsic outputs of the model.

Other Fundamental Challenges

In addition to the challenges posed by neutrino masses, dark matter, dark energy, and the hierarchy problem, several other unresolved questions driven either by experimental observations or theoretical proposals highlight the need for BSM models. The matter-antimatter asymmetry of the universe, or baryogenesis, where the SM alone cannot account for the observed matter-antimatter imbalance. The strong CP problem, which raises questions about the absence of observable CP violation in strong interactions. Grand Unified Theories (GUTs) represent another theoretical challenge, aiming to unify the three fundamental forces described by the SM into a single force governed by a larger symmetry group at a high energy scale, Λ_{GUT}.[1]

Ultimately, the grand challenge is to construct a theory that incorporates all interactions, including gravity. Gravity, unlike the other forces, is described by General Relativity (GR), a classical theory that explains it as the curvature of spacetime caused by mass and energy. However, integrating gravity into a quantum framework has proven to be an immense challenge. Developing a consistent theory of quantum gravity requires combining the principles of GR with those of quantum mechanics, a task that has inspired many efforts and ideas, such as string theory and loop quantum gravity. This unification will be essential for understanding extreme phenomena where both gravitational and quantum effects are significant, such as black-hole physics and the conditions of the early universe [80, 81].

2.3.1 New Physics Anomalies

We have discussed some of the major and fundamental problems of the SM, which have posed challenges for several decades. However, there are other types of hints that

[1] For detailed discussions on these challenges, refer to the extensive lectures and reviews on BSM physics, such as [77, 78, 79].

call for the construction of BSM frameworks. These include anomalies observed in current experimental results. An anomalous effect refers to a deviation in a specific process or observable from the predictions made by our current understanding of the universe, in this case, the SM.

In [82], a deviation in an experimental measurement is categorised as an *anomaly* based on three criteria. The first, *Statistical Significance*, requires the observed deviation to have a global statistical significance of at least 3σ. Secondly, *Experimental Validation* emphasises that the signal should appear in multiple independent channels or be detected by more than one experiment, ensuring reproducibility and robustness. Finally, *Theoretical Consistency* mandates that the deviation be explained by a robust theoretical model that aligns with existing experimental constraints and established particle physics knowledge.

As noted in [10], *we are in an era of anomalies.* Numerous anomalous results have been reported, with a comprehensive summary and explanation provided in [9] and [82]. These anomalies span a wide range of energy scales, and include phenomena such as the anomalous magnetic moment of the muon (a_μ), nuclear decays suggesting a light ~ 17 MeV boson, excesses and deficits of electron neutrinos, β-decay anomalies, hints of CP violation in hadronic meson decays, and flavour-changing neutral current semi-leptonic B-decays, among others. In this work, however, we focus specifically on the anomalies observed in scalar searches.

2.3.2 Higgs Boson Searches

Results from new Higgs boson experimental searches are expressed as limits on the *signal strength* μ, introduced in Section 2.2. This parameter quantifies the scaling of the total SM rate for a specific signal channel or an ensemble of channels. Combined results from Higgs boson searches at CMS [11] and ATLAS [83] in the $H \to \gamma\gamma$ final state reported excesses of 2.9σ and 1.7σ, respectively, corresponding to a resonant mass value of 95.4 GeV. The measured signal strength for this result is given as follows:

$$\mu_{\gamma\gamma}^{\text{exp}} = \mu_{\gamma\gamma}^{\text{ATLAS+CMS}} = \frac{\sigma^{\text{exp}}(gg \to \phi \to \gamma\gamma)}{\sigma^{\text{SM}}(gg \to H \to \gamma\gamma)} = 0.27^{+0.10}_{-0.09}, \tag{2.30}$$

where ϕ is the possible particle behind the observed anomaly and H is a would-be SM Higgs boson, both with a 95.4 GeV mass. Additionally, though, two other search channels presented anomalies which support the possibility of such a $\gamma\gamma$ resonance. LEP [13] reported a now long-standing anomaly in searches for light Higgs bosons in the $e^+e^- \to Z(H \to b\bar{b})$ channel, corresponding to a 2.3σ local excess at a Higgs mass 98 GeV, leading to a signal strength modifier given by

$$\mu_{bb}^{\text{exp}} = \frac{\sigma\left(e^+e^- \to Z\phi \to Zb\bar{b}\right)}{\sigma^{SM}\left(e^+e^- \to ZH \to Zb\bar{b}\right)} = 0.117 \pm 0.057. \tag{2.31}$$

The CMS collaboration has detected an excess in the low-mass region for the gluon-fusion production mode and decay into $\tau^{\pm}\tau^{-}$pairs [12], which is consistent with the excess observed in the di-photon search by CMS. For a mass value of 95GeV, CMS has reported a local significance of 2.6σ. This corresponds to a signal strength

$$\mu_{\tau\tau}^{\text{exp}} = \frac{\sigma^{\text{exp}}\left(gg \to \phi \to \tau^{+}\tau^{-}\right)}{\sigma^{\text{SM}}\left(gg \to H \to \tau^{+}\tau^{-}\right)} = 1.2 \pm 0.5. \tag{2.32}$$

The various mass values reported are consistent with each other, given the limited mass resolutions, particularly for the $b\bar{b}$ and $\tau^{+}\tau^{-}$ final states. The results of searches for light neutral scalars at the LHC, using the $\gamma\gamma$ and $\tau^{+}\tau^{-}$ channels, and at LEP, using the $b\bar{b}$ channel, collectively provide compelling evidence supporting the interpretation of these signals as potential indicators of new physics within the framework of BSM theories.

2.4 Beyond the Standard Model

The historic success and decades of development have shown that, on one hand, the SM is the most successful theory we have for describing the universe at its smallest scale. On the other hand, it is not complete. Many of the fundamental challenges outlined in the previous section highlight the gaps in the SM across a wide range of energy scales. Theoretical questions remain open, as theory has often been at the forefront of discovery in the history of physics, a notable example being symmetry. The current task is to build a successor to the SM, a comprehensive high-energy scale model that addresses as many of these challenges as possible while containing the SM at low energy scales, in accordance with the correspondence principle. Such a model is known as BSM.

In retrospect, the theoretical framework developed for the SM suggests a kind of recipe, as follows [84]: choose the general gauge symmetry group and the corresponding gauge bosons, define the number of matter fields, specify the field representation for each field under the symmetry group, add scalar fields and define the potential to generate masses, define the covariant derivatives, and write the most general renormalizable Lagrangian invariant under the chosen group. Use the QFT machinery to calculate observables and make predictions about both old and new phenomena, then compare these with experimental observations. If there is a mismatch, revise and try again.

In this section, we outline some of the approaches explored in the literature to extend the SM, addressing key challenges and explaining new physics, including neutrino masses, dark matter, and anomalies observed in experimental searches. The landscape of BSM theories is immense, and this discussion is not intended to be an exhaustive review. Rather, the goal is to provide an overview of how extensions to different sectors of the SM can influence various aspects of particle phenomenology. Often, modifications to multiple sectors are required simultaneously to ensure theoretical consistency and alignment with experimental observations.

2.4.1 Modified Scalar Sector

Extending the scalar sector involves introducing additional scalar fields beyond the Standard Model Higgs boson. One of the most studied examples is the Two-Higgs-Doublet Model (2HDM) [85], which includes two scalar doublets. This results in additional physical states: two CP-even scalars (h and H), one CP-odd scalar (A), and two charged Higgs bosons ($H^{\pm}$). Another example is the Georgi-Machacek Model [86], which extends the scalar sector with scalar triplets, leading to new phenomena such as doubly charged Higgs bosons ($H^{\pm\pm}$). These additional scalar states can manifest in collider experiments as excesses or resonances in channels such as di-photon or multi-lepton final states.

New scalar fields can also act as dark matter candidates or mediate interactions between dark matter and the Standard Model. A simple extension involves adding a scalar singlet [87], which can be stable and serve as a dark matter particle. Scalars may also mediate interactions between Standard Model particles and a hidden sector, resulting in the so-called Higgs-portal dark matter models [88].

Moreover, scalars can facilitate the generation of neutrino masses through mechanisms like the seesaw mechanism [89]. In the Type II Seesaw, a scalar triplet is introduced that couples to leptons, giving rise to neutrino masses.

Extended scalar sectors can also enable mechanisms like electroweak baryogenesis, where additional scalar fields modify the Higgs potential to allow a first-order phase transition. This is a critical ingredient for explaining the observed matter-antimatter asymmetry in the universe. For a comprehensive review of extended scalar sector models and their implications, please see [90].

2.4.2 Extended Fermion Sector

In the SM, fermions include quarks and leptons, arranged in three generations. Extensions modify this structure by Adding right-handed neutrinos to explain neutrino masses. Introducing vector-like fermions that couple to the SM Higgs and gauge bosons without causing anomalies. Including exotic fermions, such as particles with unconventional charges or representations under the SM gauge group.

Adding right-handed neutrinos is a common approach in seesaw models to explain the smallness of neutrino masses and neutrino oscillations [89]. Right-handed neutrinos

(ν_R) are SM gauge singlets, but in some extensions like $U(1)B-L$, they carry charges under the additional symmetry. They can generate small neutrino masses through the *Type I seesaw mechanism*, where the light neutrino mass is approximately:

$$m\nu \approx \frac{m_D^2}{M_R},$$

with m_D as the Dirac mass (typically of the order of quark and charged lepton masses) and M_R as the large mass of the right-handed neutrino. The heavy neutrinos can also lead to lepton number-violating processes, such as neutrinoless double-beta decay.

The introduction of Vector-Like Fermions (VLFs) [91, 92] is another path for the extension of the SM fermion sector. In the SM left-handed and right-handed components transform differently under the $SU(2)_L \times U(1)_Y$ gauge symmetry. In the case of VLFs, they possess both left-handed and right-handed components that transform identically under these symmetries. This unique property allows VLFs to mix with SM fermions without introducing gauge anomalies, maintaining theoretical consistency. Such mixing can lead to observable deviations in precision measurements, including modifications to Higgs boson decays and alterations in electroweak couplings. For instance, VLFs can influence processes like $H \to \gamma\gamma$ and affect interactions involving the Z and W bosons.

2.4.3 New Gauge Groups

Extending the gauge sector involves introducing new gauge symmetries and associated gauge bosons beyond the SM.

A simple and widely studied example is to extend the SM group with an Abelian $U(1)'$ symmetry. A new boson, known as the Z' [93], arises as the gauge boson of this symmetry. The Z' can mix with the Z boson from the SM via kinetic or mass mixing, which affects its couplings to SM particles. The $U(1)'$ symmetry must be spontaneously broken, usually by introducing a scalar field that acquires a VEV, thereby also modifying the scalar sector. The properties of the Z' boson, such as its mass, couplings, and decay channels, depend on the specific realization of the $U(1)$ symmetry. Popular examples include $U(1)_{B-L}, U(1)_X$, and other flavor-dependent symmetries like $U(1)_{L_i-L_j}$, offering promising insights into dark matter interactions, flavor physics, and collider phenomenology [94].

Particularly, the $U(1)_{B-L}$ extension, which corresponds to the difference between baryon number (B) and lepton number (L), modifies the fermion sector by introducing right-handed neutrinos (ν_R) to cancel anomalies and enable the *seesaw mechanism*, generating small neutrino masses through mixing with light SM neutrinos. The associated Z' boson couples to SM fermions in proportion to their $(B-L)$ charges, mediating interactions that can be tested in collider experiments or precision flavour studies.

Grand Unified Theories (GUTs) are motivated by the unification of the SM gauge interactions into a single framework, with groups like $SU(5)$ [95], $SO(10)$ [96], or E_6 [97]. In these models, a symmetry breaking chain is required to reduce the higher-dimensional gauge group to the SM gauge group.

GUTs predict new gauge bosons associated with the larger symmetry group, such as X and Y bosons (in $SU(5)$). These heavy bosons mediate interactions that violate baryon and lepton number, leading to proton decay. GUTs also introduce Z' bosons through additional $U(1)'$ symmetries in $SO(10)$ or E_6, with unique couplings to SM particles. GUTs require additional scalar fields for symmetry breaking, typically involving fields in higher-dimensional representations.

The fermion sector is also extended, as GUTs unify all SM fermions (quarks and leptons) into a single representation of the larger symmetry group, often introducing right-handed neutrinos and exotic fermions. These exotic fermions may include stable particles that can serve as dark matter candidates. Hence, GUTs offer a rich and complex phenomenology that can address many unresolved challenges described in Section 2.3, such as proton decay via new gauge bosons, neutrino masses via the seesaw mechanism with right-handed neutrinos, and stable exotic fermions as dark matter candidates.

2.5 Supersymmetry

Supersymmetry (SUSY), proposed in the 1970s [98, 99], is considered as one of the most appealing candidates for physics beyond the SM. Rather than modifying a single sector of the SM, it extends all sectors by proposing the existence of **superpartners** for every known particle by introducing a symmetry that connects fermions and bosons.

SUSY offers several theoretical and phenomenological advantages [100], addressing key challenges in particle physics. It provides a solution to the hierarchy problem between the electroweak and Planck scales by introducing superparticle corrections to the Higgs mass, stabilising it without fine-tuning. It predicts a naturally light Higgs boson mass, consistent with experimental results, and facilitates the unification of SM gauge couplings at high energy scales. SUSY also offers a natural candidate for dark matter (the Lightest Supersymmetric Particle, or LSP) and incorporates gravity via supergravity (SUGRA), connecting it to string theory.

The aim of this chapter is to introduce the fundamental concepts of supersymmetric theories and provide an overview of SUSY, its simplest realisation known as the Minimal Supersymmetric Standard Model (MSSM), and an extension of the MSSM with the potential to explain light scalar anomalies, neutrino masses, and other phenomena. The later corresponds to the $(B-L)$SSM model, a $U(1)_{(B-L)}$ extension of the MSSM, which serves as the primary BSM framework used in this work.

2.5.1 SUSY Essentials

SUSY transforms particles of integer spin (bosons) to those with spin $\frac{1}{2}$ (fermions) and vice versa via the generators $\mathcal{Q}$, which acts on states as follows,

$$\mathcal{Q} \mid \text{Fermion}\rangle =\mid \text{Boson}\rangle, \quad \mathcal{Q} \mid \text{Boson}\rangle =\mid \text{Fermion}\rangle. \tag{2.33}$$

This relationship implies that each particle has a corresponding superpartner with spin differing by $\frac{1}{2}$. Extensive literature on SUSY, including its algebra and theoretical foundations is available, for example, see [101, 102, 103].

To incorporate SUSY into a field-theoretical framework, the *superfield formalism* is employed. A *superfield* is a mathematical object that combines all the *component fields* of a *supermultiplet* (scalars, fermions, gauge fields, etc.) and their interactions, ensuring consistency under supersymmetry transformations. Examples of supermultiplets include:

- **Chiral supermultiplets**, which contain a scalar (spin-0 boson) and its fermionic partner (spin- $\frac{1}{2}$).
- **Gauge supermultiplets**, which include a gauge boson (spin-1) and its fermionic partner, the gaugino (spin- $\frac{1}{2}$).

These supermultiplets form the building blocks of supersymmetric theories, providing a unified framework for bosons and fermions within the Lagrangian. The kinetic terms for the component fields are derived from the superfields, leading to a canonical kinetic Lagrangian of the form:

$$\mathcal{L}_{\text{kin}} = \sum_i \left\{ \left(D_\mu S_i^*\right)\left(D^\mu S_i\right) + i\bar{\psi}_i D_\mu \gamma^\mu \psi_i \right\} + \sum_a \left\{ -\frac{1}{4} F_{\mu\nu}^a F^{\mu\nu a} + \frac{i}{2} \bar{\lambda}_a \sigma^\mu D_\mu \lambda_a \right\} \tag{2.34}$$

where D_μ is the standard covariant derivative and $\sigma_{1,2,3}, -\sigma_0$ are the 2×2 Pauli and unit matrices. The first term corresponds to the kinetic contribution of the chiral supermultiplet, which includes a scalar field S_i and its fermionic partner ψ_i. The second term describes the gauge supermultiplet, featuring $F_{\mu\nu}^a$, the gauge field strength tensor for the spin-1 gauge bosons, and their fermionic partners λ_a , the gauginos.

Names	Superfield $\hat{S}$	Spin 0	Spin 1/2	$SU(3)_C, SU(2)_L, U(1)_Y$
squarks, quarks	$\hat{Q}$	$\widetilde{q} = \left(\widetilde{u}_L,\ \widetilde{d}_L\right)$	$q = (u_L,\ d_L)$	$\left(\mathbf{3}, \mathbf{2}, \frac{1}{6}\right)$
($\times$3 families)	$\hat{U}$	$\widetilde{u}^*_R$	$u^\dagger_R$	$\left(\overline{\mathbf{3}}, \mathbf{1}, -\frac{2}{3}\right)$
	$\hat{D}$	$\widetilde{d}^*_R$	$d^\dagger_R$	$\left(\overline{\mathbf{3}}, \mathbf{1}, \frac{1}{3}\right)$
sleptons, leptons	$\hat{L}$	$\widetilde{L} = (\widetilde{\nu},\ \widetilde{e}_L)$	$L = (\nu,\ e_L)$	$\left(\mathbf{1}, \mathbf{2}, -\frac{1}{2}\right)$
($\times$3 families)	$\hat{E}$	$\widetilde{e}^*_R$	$e^\dagger_R$	$(\mathbf{1}, \mathbf{1}, 1)$
Higgs, higgsinos	$\hat{H}_u$	$H_u = (H^+_u\ H^0_u)$	$\widetilde{H}_u = \left(\widetilde{H}^+_u\ \widetilde{H}^0_u\right)$	$\left(\mathbf{1}, \mathbf{2}, +\frac{1}{2}\right)$
	$\hat{H}_d$	$H_d = \left(H^0_d\ H^-_d\right)$	$\widetilde{H}_d = \left(\widetilde{H}^0_d\ \widetilde{H}^-_d\right)$	$\left(\mathbf{1}, \mathbf{2}, -\frac{1}{2}\right)$

TABLE 2.1: Chiral supermultiplets in the Minimal Supersymmetric Standard Model (MSSM), showing the names of the fields (squarks, sleptons, and Higgs bosons), their corresponding superfields, spins, and representations under the SM gauge groups. Adapted from [6].

Interactions between the fields are dictated by the superpotential, which contributes to the scalar potential and generates Yukawa interactions. The form of the superpotential is theory-dependent, shaped by the field content, symmetries, and objectives of the specific supersymmetric model.

It is important to note that SUSY cannot be an exact symmetry in nature, as this would require superpartners to have the same masses as their SM counterparts, which contradicts experimental observations. To address this, explicit SUSY-breaking terms are introduced into the theory. These terms prevent the reappearance of quadratic divergences while effectively parametrising our current ignorance of the fundamental SUSY-breaking mechanism. Known as soft SUSY-breaking terms, they give rise to a low-energy effective SUSY theory.

2.5.2 The MSSM

The MSSM is the simplest realisation of SUSY, extending the SM with the minimal particle content and interactions required to preserve SUSY while incorporating SUSY-breaking. All particles in the MSSM are organised into supermultiplets, pairing each SM particle with its corresponding superpartner, as shown in Tables 2.1 and 2.2. Additionally, R-parity is imposed to ensure the conservation of lepton and baryon numbers and to stabilise the LSP.

The most general superpotential that satisfies gauge invariance, renormalisability, and R-parity conservation is given by [104]:

$$W = \sum_{i,j=gen} -Y^u_{ij}\hat{u}_{Ri}\hat{H}_u \cdot \hat{Q}_j + Y^d_{ij}\hat{d}_{Ri}\hat{H}_d \cdot \hat{Q}_j + Y^\ell_{ij}\hat{\ell}_{Ri}\hat{H}_d \cdot \hat{L}_j + \mu\hat{H}_u \cdot \hat{H}_d \tag{2.35}$$

Names	Superfield $\hat{S}$	Spin 1/2	Spin 1	$SU(3)_C, SU(2)_L, U(1)_Y$
gluino, gluon	$\hat{G}^a$	$\widetilde{G}$	g	$(\mathbf{8}, \mathbf{1}, 0)$
winos, W bosons	$\hat{W}$	$\widetilde{W}^{\pm}\widetilde{W}^{0}$	$W^{\pm}W^{0}$	$(\mathbf{1}, \mathbf{3}, 0)$
bino, B boson	$\hat{B}$	$\widetilde{B}^0$	B^0	$(\mathbf{1}, \mathbf{1}, 0)$

TABLE 2.2: Gauge supermultiplets in the Minimal Supersymmetric Standard Model (MSSM), showing the names of the gauge bosons and their superpartners (gauginos), their associated superfields, spins, and representations under the SM gauge groups. Adapted from [6].

where $H \cdot Q \equiv \epsilon_{ab} H^a Q^b$ is the product between $\mathrm{SU(2)_L}$ doublets with a, b are $\mathrm{SU(2)_L}$ indices and $\epsilon_{12} = 1 = -\epsilon_{21}$. The first three term contain the Yukawa couplings among generations denoted by $Y_{ij}^{u,d,\ell}$. In addition to the supersymmetric terms in the Lagrangian, the MSSM incorporates the explicit SUSY-breaking terms in the $\mathcal{L}_{\text{soft}}$ Lagrangian and is given by,

$$-\mathcal{L}_{\text{soft}}^{\text{MSSM}} =$$

$$\frac{1}{2}\left[M_1 \tilde{B}\tilde{B} + M_2 \sum_{a=1}^{3} \tilde{W}^a \tilde{W}_a + M_3 \sum_{a=1}^{8} \tilde{G}^a \tilde{G}_a + \text{ h.c. }\right] \tag{2.36}$$

$$+ \sum_{i=gen} m_{\tilde{Q}_i}^2 \tilde{Q}_i^\dagger \tilde{Q}_i + m_{\tilde{L}_i}^2 \tilde{L}_i^\dagger \tilde{L}_i + m_{\tilde{u}_i}^2 \left|\tilde{u}_{R_i}\right|^2 + m_{\tilde{d}_i}^2 \left|\tilde{d}_{R_i}\right|^2 + m_{\tilde{\ell}_i}^2 \left|\tilde{\ell}_{R_i}\right|^2 \tag{2.37}$$

$$+ m_{H_u}^2 H_u^\dagger H_2 + m_{H_d}^2 H_d^\dagger H_d + B\mu \left(H_u \cdot H_d + \text{ h.c. }\right) \tag{2.38}$$

$$+ \sum_{i,j=gen} \left[A_{ij}^u Y_{ij}^u \tilde{u}_{R_i}^* H_u \cdot \tilde{Q}_j + A_{ij}^d Y_{ij}^d \tilde{d}_{R_i}^* H_d \cdot \tilde{Q}_j + A_{ij}^\ell Y_{ij}^\ell \tilde{\ell}_{R_i}^* H_d \cdot \tilde{L}_j + \text{ h.c. }\right] \tag{2.39}$$

The soft SUSY-breaking Lagrangian in the MSSM includes several key terms. The gaugino mass terms (M_1, M_2, M_3) provide masses for the bino, winos, and gluinos, respectively, as seen in the first term (2.36). Scalar mass terms $(m_{\tilde{Q}i}, m_{\tilde{L}i}, m_{\tilde{u}i}, m_{\tilde{d}i}, m_{\tilde{\ell}i}, m_{H_u}, m_{H_d})$ correspond to the masses of squarks, sleptons, and the Higgs fields, appearing in the second (2.37) and third (2.38) terms. The bilinear $B\mu$ term represents Higgs mixing, which contributes to electroweak symmetry breaking, as shown in the third term (2.38). Finally, the trilinear terms $(A_{ij}^u, A_{ij}^d, A_{ij}^\ell)$ couple scalar fields through Yukawa couplings, ensuring proper interactions between generations, as outlined in the fourth term (2.39).

This framework is known as the unconstrained MSSM, where, in the general case, around 100 unknown parameters are introduced in addition to the 19 parameters of the SM. This large parameter space makes any phenomenological analysis in the MSSM highly complex. To address this, the constrained MSSM (cMSSM) introduces several simplifying assumptions, imposing constraints on the soft SUSY-breaking parameters based on a set of universal boundary conditions at the GUT scale.

These constraints are motivated by the minimal Supergravity (mSUGRA) model, where SUSY breaking occurs in a hidden sector that communicates with the visible sector through gravitational interactions mediated by Supergravity. These interactions are assumed to be flavor-blind, resulting in universal soft SUSY-breaking terms, as outlined in the equations above. Fixing the GUT scale at $M_U \sim 2 \times 10^{16}$ GeV, the unification conditions in mSUGRA lead to the following relations for the soft parameters:

$$M_1 = M_2 = M_3 \equiv m_{1/2} \tag{2.40}$$

$$m_{\tilde{Q}_i} = m_{\tilde{u}_{Ri}} = m_{\tilde{d}_{Ri}} = m_{\tilde{L}_i} = m_{\tilde{\ell}_{Ri}} = m_{H_1} = m_{H_2} \equiv m_0 \tag{2.41}$$

$$A^u_{ij} = A^d_{ij} = A^\ell_{ij} \equiv A_0 \delta_{ij} \tag{2.42}$$

where m_0 is the *universal scalar mass*, A_0 is the *universal trilinear coupling*, and $m_{1/2}$ is the *universal gaugino mass*. These parameters describe the supersymmetric sector at the GUT scale, along with the bilinear coupling B and the supersymmetric Higgsino mass parameter μ.

2.5.3 Beyond Minimality

While SUSY and the MSSM were introduced in this work as BSM scenarios that address several shortcomings of the SM discussed in Section 2.3, such as stabilising the Higgs mass, achieving gauge coupling unification, and providing a dark matter candidate, the MSSM is unlikely to be the final theory of nature due to persistent challenges. One issue is the μ problem [105, 106], which questions why the SUSY Higgs mass term μ in the superpotential is of the same order as the SUSY breaking scale, despite the two scales having different origins, and why it satisfies the hierarchy $\mu \ll M_{GUT}, M_{Pl}$.

Additionally, the little hierarchy problem emerges because the absence of light superpartners, as suggested by experimental searches, requires fine-tuning of parameters to maintain the Higgs mass near the electroweak scale despite large quantum corrections [76, 107]. Finally, the lack of experimental detection of superpartners at expected mass ranges, such as at the Large Hadron Collider [108, 109], significantly constrains the parameter space of these models.

Beyond theoretical considerations, the MSSM also faces challenges from light scalar search results, particularly the $\sim$ 95 GeV excess observed by CMS and ATLAS in the diphoton channel, as discussed in the previous sections. This anomaly may hint at the presence of an additional light scalar state. While the MSSM includes two CP-even Higgs bosons, accommodating a second light Higgs at 95 GeV, in a manner consistent with both this excess and existing exclusion limits, is difficult. Such a scenario typically requires a non-standard mass hierarchy between the Higgs states [104] and finely tuned mixing angles [110] to sufficiently suppress the scalar's couplings to vector bosons and fermions.

These challenges indicate that while minimal realisations of SUSY are valuable effective low-energy frameworks, they cannot be the ultimate theories of nature [36]. Extensions like the NMSSM [111], $(B-L)$SSM [36], or entirely new frameworks are required to address these unresolved issues and provide a more complete understanding of fundamental physics [111].

2.6 The $(B-L)$ SSM

The $(B-L)$SSM [36, 112, 113, 7] is essentially the MSSM extended by a $U(1)_{B-L}$ gauge symmetry,

$$\mathcal{G}_{(B-L)\text{SSM}} = U(1)_Y \otimes SU(2)_L \otimes SU(3)_c \otimes U(1)_{B-L},$$

wherein the $U(1)_{B-L}$ symmetry is spontaneously broken through the Higgs mechanism. The chiral superfields and their quantum numbers are summarized in Table 2.3. The Superpotential of the model is given by [7]

$$\begin{aligned} W_{(B-L)\text{SSM}} =& Y_u^{ij}\hat{U}_i\hat{Q}_j\hat{H}_u - Y_d^{ij}\hat{D}_i\hat{Q}_j\hat{H}_d - Y_e^{ij}\hat{E}_i\hat{L}_j\hat{H}_d + \mu\hat{H}_u\hat{H}_d \\ &+ Y_\nu^{ij}\hat{L}_i\hat{H}_u\hat{\nu}_j - \mu'\hat{\eta}\hat{\bar{\eta}} + Y_x^{ij}\hat{\nu}_i\hat{\eta}\hat{\nu}_j \end{aligned} \tag{2.43}$$

where, i, j are generation indices and all colour and isospin indices are suppresed. Here, the first four terms corresponds to the MSSM Superpotential, incorporating the Yukawa interactions with their respective Yukawa couplings, namely Y_u, Y_d and Y_e. Additional terms describe the interactions between the (s)neutrinos $\hat{\nu}$ and the singlet Higgs Superfield $\hat{\eta}$. The corresponding Yukawa coupling constants are denoted as y_v and y_N. Finally, the μ'-term is the bilinear mixing between the singlet Higgs fields $\hat{\eta}$ and $\hat{\bar{\eta}}$.

The additional soft-SUSY breaking terms are given by [7]

$$\begin{aligned} \mathcal{L}_{\text{soft}}^{(B-L)\text{SSM}} &= \mathcal{L}_{\text{soft}}^{\text{MSSM}} - \lambda_{\tilde{B}}\lambda_{\tilde{B}'}M_{BB'} - \frac{1}{2}\lambda_{\tilde{B}'}\lambda_{\tilde{B}'}M_{B'} \\ &- m_\eta^2|\eta|^2 - m_{\bar{\eta}}^2|\bar{\eta}|^2 - m_{\nu,ij}^2\left(\tilde{\nu}_i^c\right)^*\tilde{\nu}_j^c - \eta\bar{\eta}B_{\mu'} + T_\nu^{ij}H_u\tilde{\nu}_i^c\tilde{L}_j + T_x^{ij}\eta\tilde{\nu}_i^c\tilde{\nu}_j^c \end{aligned} \tag{2.44}$$

Superfield	Spin 0	Spin 1/2	Generations	$(U(1)_Y \otimes SU(2)_L \otimes SU(3)_C \otimes U(1)_{B-L})$
$\hat{Q}$	$\tilde{Q}$	Q	3	$1/6 \otimes \mathbf{2} \otimes \mathbf{3} \otimes 1/6$
$\hat{D}$	$\tilde{d}^c$	d^c	3	$1/3 \otimes \mathbf{1} \otimes \overline{\mathbf{3}} \otimes -1/6$
$\hat{U}$	$\tilde{u}^c$	u^c	3	$-2/3 \otimes \mathbf{1} \otimes \overline{\mathbf{3}} \otimes -1/6$
$\hat{L}$	$\tilde{L}$	L	3	$-1/2 \otimes \mathbf{2} \otimes \mathbf{1} \otimes -1/2$
$\hat{E}$	$\tilde{e}^c$	e^c	3	$1 \otimes \mathbf{1} \otimes \mathbf{1} \otimes 1/2$
$\hat{\nu}$	$\tilde{\nu}^c$	ν^c	3	$0 \otimes \mathbf{1} \otimes \mathbf{1} \otimes 1/2$
$\hat{H}_d$	H_d	$\tilde{H}_d$	1	$-1/2 \otimes \mathbf{2} \otimes \mathbf{1} \otimes 0$
$\hat{H}_u$	H_u	$\tilde{H}_u$	1	$1/2 \otimes \mathbf{2} \otimes \mathbf{1} \otimes 0$
$\hat{\eta}$	η	$\tilde{\eta}$	1	$0 \otimes \mathbf{1} \otimes \mathbf{1} \otimes -1$
$\hat{\bar{\eta}}$	$\bar{\eta}$	$\tilde{\bar{\eta}}$	1	$0 \otimes \mathbf{1} \otimes \mathbf{1} \otimes 1$

TABLE 2.3: Chiral supermultiplets in the $(B-L)$ SSM and representations under the $SU(3)_C \otimes SU(2)_L \otimes U(1)_Y \otimes U(1)_{B-L}$ gauge groups. Adapted from [7].

The neutral components of the Higgs doublets H_u^0 and H_d^0, as well as the bilepton fields η and $\bar{\eta}$, acquire non-zero VEVs as follows:

$$\begin{aligned} H_d^0 &= \frac{1}{\sqrt{2}}\left(i\sigma_d + v_d + \phi_d\right), \quad H_u^0 = \frac{1}{\sqrt{2}}\left(i\sigma_u + v_u + \phi_u\right) \\ \eta &= \frac{1}{\sqrt{2}}\left(i\sigma_\eta + v_\eta + \phi_\eta\right), \quad \bar{\eta} = \frac{1}{\sqrt{2}}\left(i\sigma_{\bar{\eta}} + v_{\bar{\eta}} + \phi_{\bar{\eta}}\right) \end{aligned} \tag{2.45}$$

where each field is decomposed into its CP-odd component (σ), CP-even component (ϕ), and the VEV (v) responsible for symmetry breaking.

The VEVs of the bilepton fields η and $\bar{\eta}$ break the $U(1)_{B-L}$ symmetry, giving rise to a new Z' gauge boson associated with the $B-L$ group. In analogy with the MSSM, the ratio of the VEVs of the bilepton fields is defined as $\tan\beta' = \frac{v_\eta}{v_{\bar{\eta}}}$. This parameter plays a crucial role in determining the phenomenology of the $B-L$ sector, including the masses and mixings of the Z' boson and other particles, such as the neutral scalars.

After $B-L$ breaking, the term $Y_x^{ij}\hat{\nu}_i\hat{\eta}\hat{\nu}_j$ in the superpotential generates a Majorana mass for the right-handed neutrinos. This Majorana mass allows for small masses for the left-handed neutrinos ν_L through the *Type-I seesaw mechanism.*

2.6.1 Neutral Higgs bosons

In the Higgs sector, the CP-even and CP-odd components of the doublets mix with the corresponding CP eigenstates of the bilepton fields. This mixing results in four physical scalar Higgs particles:

$$(\phi_d, \phi_u, \phi_\eta, \phi_{\bar{\eta}}) \to (h_1, h_2, h_3, h_4) \tag{2.46}$$

where h_1 is typically identified as the Standard Model-like Higgs boson, accommodating the observed Higgs mass of approximately 125 GeV, and the others

(h_2, h_3, h_4) correspond to additional physical scalar states predicted by the $(B-L)$SSM. In the CP-odd sector, the pseudo-scalars are similarly mixed, leading to:

$$(\sigma_d, \sigma_u, \sigma_\eta, \sigma_{\bar{\eta}}) \to \left(G^Z, G^{Z'}, A_1^h, A_2^h\right) \tag{2.47}$$

Here, G^Z and $G^{Z'}$ are the Goldstone bosons that are *eaten* by the Z and Z' gauge bosons, respectively, providing their longitudinal components. A_1^h and A_2^h are the two physical pseudo-scalar Higgs particles.

These scalar and pseudo-scalar particles are prime candidates for exploration in scalar searches at colliders. The model naturally accommodates the observed 125 GeV Higgs boson while leaving room for other scalar resonances, which could explain anomalies observed in scalar searches.

Chapter 3

Automated BSM Phenomenology

The previous chapter reviewed what is called *model building* in particle physics, the process of constructing theoretical frameworks to describe physical phenomena. Once a model is defined, phenomenological studies translate its theoretical predictions into numerical results that can be compared with experimental data. This involves:

1. **Symbolic theoretical calculations**, such as defining the Lagrangian and deriving Feynman rules.
2. **Numerical evaluations**, which calculate observables like cross-sections and decay rates based on input parameters.
3. **Experimental verification**, where the numerical predictions of the model are compared with experimental data from particle colliders like the LHC.

Computational frameworks facilitate these tasks, offering both model-dependent and model-independent tools to simulate BSM scenarios. In this chapter, we examine the computational challenges associated with model building, highlighting the main focus of the present work, the *parameter scan problem*.

3.1 High Energy Physics Software

Modern particle physics relies heavily on software tools to connect theoretical models with experimental data. These tools play a crucial role in automating complex calculations, simulating particle interactions, enabling parameter scans across large theoretical spaces and verifying experimental measurements with theoretical predictions. High Energy Physics (HEP) software can be broadly classified into symbolic, numerical and experimental verification tools, each serving a distinct

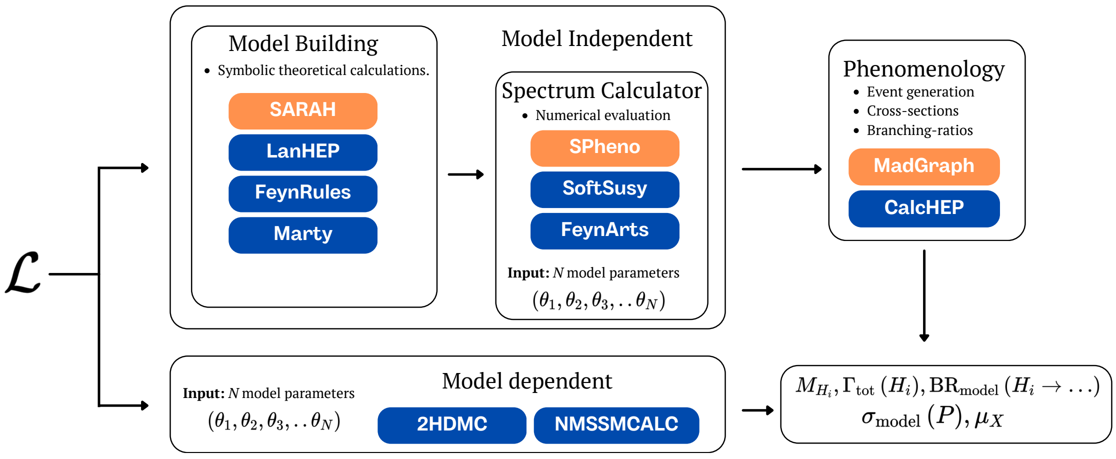

Model Building
Symbolic theoretical calculations.
SARAH
LanHEP
FeynRules
Marty
Model Independent
Spectrum Calculator
Numerical evaluation
SPheno
SoftSusy
FeynArts
Input: N model parameters
$(\theta_1, \theta_2, \theta_3, ..\theta_N)$
Phenomenology
Event generation
Cross-sections
Branching-ratios
MadGraph
CalcHEP
$\mathcal{L}$
Input: N model parameters
$(\theta_1, \theta_2, \theta_3, ..\theta_N)$
Model dependent
2HDMC
NMSSMCALC
$M_{H_i}, \Gamma_{\text{tot}}(H_i), \text{BR}_{\text{model}}(H_i \to \dots)$
$\sigma_{\text{model}}(P), \mu_X$

FIGURE 3.1: A schematic representation of the workflow in High Energy Physics (HEP) software for theoretical model implementation and phenomenological analysis. Starting from a Lagrangian, the process involves three key stages: Model Building, Spectrum Calculation, and Phenomenological Predictions. These tasks can be performed using general model-independent frameworks (top track) or model-specific tools (bottom track). Tools used in this thesis are highlighted in orange.

purpose in phenomenological studies. Symbolic tools are designed to handle the analytical aspects of theoretical physics, automating tasks that were traditionally carried out by hand. They allow researchers to define a model's Lagrangian, derive Feynman rules, and compute symbolic expressions for scattering amplitudes and other processes. Examples of widely used symbolic tools include:

- **SARAH**: SARAH [114, 115] is a Mathematica package designed for the analytical and numerical study of BSM theories, including both SUSY and non-SUSY models. Users can define various aspects of a model, such as particle content, interactions, symmetries, symmetry breaking, and particle mixing. Based on this input, SARAH automatically derives the Lagrangian, calculates one- and two-loop renormalization group equations (RGEs), determines the tadpole equations, and computes mass matrices for particles. Additionally, SARAH generates Fortran code compatible with other programs. It can export model information to software tools such as SPheno, CalCHEP, and MadGraph. **FeynRules** [116] is another Mathematica-based package that shares many features with SARAH, focusing on model implementation, the derivation of Feynman rules and integration with HEP software.

- **LanHEP**: LanHEP [117] is a model-building program written in the C programming language, featuring its own symbolic manipulation routines, distinguishing it from Mathematica-based packages. It supports a wide range of BSM theories, with the Lagrangian provided by the user in closed form. LanHEP automatically derives the corresponding Feynman rules and exports them to various programs within the HEP ecosystem.

Symbolic tools serve as the foundation for numerical evaluations by exporting results in standardised formats like Universal FeynRules Output [118] (UFO) or SUSY Les Houches Accord [119] (SLHA).

Numerical tools build on the results provided by symbolic tools and are responsible for computing physical observables using specific parameter inputs. These tools handle tasks such as calculating cross-sections for interactions, determining decay rates and particle lifetimes, and computing precision observables that include higher-order corrections. Examples of numerical frameworks include

- **SPheno**: SPheno [120, 121] is a Fortran-based program designed to compute precise numerical predictions for SUSY and non-SUSY models using analytical results generated by SARAH. It calculates the particle mass spectrum, decay widths, branching ratios, and various low-energy and flavour observables. By using SLHA files for input and output, SPheno integrates seamlessly into the broader HEP workflow, making it an essential tool for phenomenological studies.
- **CalcHEP**: CalcHEP [122] automates the calculation of production cross-sections, decay widths, and event simulations at the lowest order of perturbation theory. It supports both SUSY and non-SUSY models, utilising model information generated by tools such as LanHEP or SARAH. The software features a user-friendly, menu-driven graphical interface alongside a versatile batch interface for advanced workflows. Its symbolic module computes squared matrix elements, exports the results as C-code, and generates corresponding executables.
- **MadGraph5_aMC@NLO**: MadGraph [123, 124] is a versatile software tool for simulating high-energy particle collisions and generating events. It automates the calculation of matrix elements for particle interactions using Feynman diagrams, supporting precise cross-section computations, including next-to-leading-order (NLO) corrections. The software accommodates a variety of particle physics models, implemented using tools like SARAH or FeynRules and exported in the UFO format, enabling the integration of both renormalisable and effective theories. Written primarily in Python, MadGraph is accessible, making it widely adopted by the physics community. Its command-line interface allows users to define processes, configure parameters, and manage various aspects of simulations, establishing MadGraph as an essential tool for theoretical and phenomenological research in HEP.

CalcHEP and MadGraph5 serve as general purpose event generators simulate the outcomes of high-energy particle collisions, making them essential for connecting theoretical predictions to experimental data. These tools model various aspects of particle interactions, including hard scattering processes, parton showers, hadronisation, and even detector-level simulations. In this family of tools PYTHIA [125] is generally use for simulating parton showers and hadronisation. For detector-level effects, tools like Delphes [126] are employed to simulate how a detector records particles' trajectories, energies, and other properties.

HEP software tools can be categorised as either model-dependent or model-independent. The programs discussed above fall into the model-independent category. Examples of model-dependent tools include 2HDMC [127], which specialises in computing particle spectra, decay rates, and other observables specifically for the Two-Higgs-Doublet Model (2HDM), and NMSSMCALC [128], which is made for the Next-to-Minimal Supersymmetric Standard Model (NMSSM). Figure 3.1 provides a schematic representation of the workflow of HEP software for theoretical model implementation and phenomenological analysis.

On top of the model building software, tools for experimental validation are crucial to ensure that theoretical predictions are consistent with existing data. Two essential tools in this context are HiggsBounds [129] and HiggsSignals [130], which are used to validate the Higgs sector of BSM theories against experimental constraints[1]. Specifically,

- **HiggsBounds (HB)**: Tests whether a given model configuration is excluded by collider experiments at a specified confidence level, typically 95% C.L. It requires inputs such as the masses, decay widths, branching ratios, and production cross-sections of the Higgs bosons predicted by the model computed by tools like SPheno. These predictions are compared with exclusion limits from experiments like the LHC and LEP. HB computes a test statistic, k_0^{HB}, representing the ratio of the predicted signal cross-section to the experimental exclusion limit. Configurations with $k_0^{\text{HB}} > 1$ are excluded, while those with $k_0^{\text{HB}} \leq 1$ are consistent with the data, ensuring that the explored parameter space adheres to existing experimental constraints.

- **HiggsSignals (HS)**: Evaluates the compatibility of a model's Higgs sector with the properties of the observed Higgs boson in the LHC experiments ATLAS [56] and CMS [57] in 2012.. Uses the same input as HB to perform a χ^2 analysis, calculating χ^2_{HS} as the sum of squared differences between predicted and measured values, normalised by experimental uncertainties.

The research project developed in this thesis uses SARAH to generate BSM model files, SPheno to calculate the mass spectrum, HB and HS to assess experimental viability, and MadGraph for cross-section calculations and other BSM observables, as illustrated in Figure 3.1 on the orange track.

[1]These two tools were recently unified into the HiggsTools [131] framework.

3.2 Parameter Scans

Challenges in BSM phenomenological analyses rare closely related to the computational complexity of the underlying theoretical models. Typically, the numerical evaluation of a BSM model configuration involves a sequential use of various HEP software packages, collectively referred to as the *HEP-Stack* and denoted by $\mathcal{H}_{\text{Model}}$. For instance, consider a *HEP-Stack* consisting of SPheno, HB and HS, where the goal is to compare the predicted Higgs boson masses of a model with their experimental constraints using HB and HS. The HEP stack can be expressed as:

$$\mathcal{H}_{\text{Model}}\left(\boldsymbol{\theta}\right) = \text{HB} \circ \text{HS} \circ \ \text{SPheno}\ \left(\boldsymbol{\theta}\right) \to \left\{m_{h_1}, m_{h_2}, k_0^{\text{HB}}, \chi^2_{\text{HS}}\right\}. \tag{3.1}$$

Here, $\boldsymbol{\theta}$ represents a parameter space configuration, with $\boldsymbol{\theta} \in \Theta$, where Θ denotes the full parameter space. This process highlights one of the major challenges in BSM phenomenology studies: *the parameter scan problem (PSP).*

The PSP [38] involves a systematic exploration of the multidimensional parameter space of a model. This process includes calculating numerical values for the predictions of a model at various points in the parameter space, applying experimental and theoretical constraints, and identifying satisfactory regions that can explain multiple phenomena. These satisfactory regions are determined by verifying whether the theoretical predictions align with measured experimental values within specified error margins or satisfy exclusion limits when no experimental observations exist. Then, *parameter Scan (PS)* methods must overcome several computational challenges, including high-dimensional parameter spaces, the integration of multiple physical constraints and the computational cost of single point evaluation of the HEP-Stack. These challenges will be explored in more detail in the following subsections.

Consequently, selecting an appropriate PS method for a specific phenomenological study is far from trivial. It requires expert knowledge of the BSM model as well as the computational aspects involved to ensure a successful and efficient analysis.

3.2.1 Baseline parameter scan methods

Grid scan

The grid scan is the most basic parameter scan method. It is convenient because it is easy and quick to implement. The method divides the parameter space into a regular grid of evenly spaced points, with a fixed number of divisions per dimension. The *HEP-Stack* is evaluated at each grid point.

For a d-dimensional parameter space, if each dimension is sampled at n points, the total number of evaluations N is given by:

$$N = n^d$$

This exponential growth in the number of grid points makes grid scans impractical for high-dimensional parameter spaces. The problem is further intensified when the viable regions of the parameter space are sparse or disconnected, as many grid points may fall into non-satisfactory regions.

Uniform random scan

The uniform random scan samples points randomly from a uniform distribution across the parameter space. Unlike the grid scan, it does not require a fixed number of evaluations per dimension, offering more flexibility. However, it faces the same challenge as grid scans in high-dimensional spaces: the sparsity of satisfactory regions. As dimensionality increases, the probability of randomly hitting a viable region becomes exponentially small, making this method inefficient.

Both methods, grid scan and uniform random scan, also suffer from the computational cost associated with the HEP-Stack used in PS methods. Since a large number of samples are required to ensure sufficient coverage of the parameter space, the computational resource cost can become prohibitively high. These problems arise due to the *curse of dimensionality*, where the volume of the parameter space grows exponentially with the number of dimensions, as explained in Section 3.2.4.

Addressing these limitations requires more *adaptive* or *targeted* PS algorithms, which focus *computational resources* on promising regions of the parameter space.

3.2.2 Computational cost

The computational cost of evaluating a single model configuration with a defined *HEP-Stack* plays a critical role in determining the efficiency and scalability of PS methods. This computational cost arises primarily from the sequential execution of HEP Software in a given study. For instance, in the context of the $(B-L)$SSM model, with the *HEP-Stack* denoted as $\mathcal{H}_{(B-L)SSM}$ following the structure of equation (3.1). As shown in Figure 3.2, the average time required per evaluation of the HEP-Stack is approximately 1 minute. This cost becomes even higher when other tools, such as those used for particle collision simulations, are included. This relatively high cost can be prohibitive when performing exhaustive scans over high-dimensional parameter spaces, where thousands or even millions of points may need evaluation. Points below the average time represent non-physical parameter space and points with evaluation times significantly above the average reflect parameter configurations where SPheno struggles to converge.

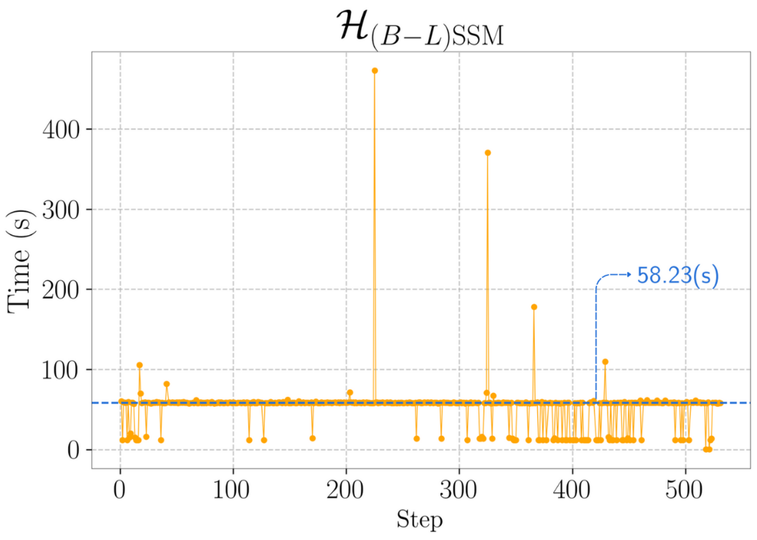


FIGURE 3.2: Chart showing the evaluation of the HEP software stack corresponding to SPheno, HiggsBounds, and HiggsSignals, denoted as $\mathcal{H}_{(B-L)\mathrm{SSM}}$. The average time per HEP stack call (Step) is approximately 1 minute. Points below the average represent non-physical parameter space, while points above the average correspond to unusual physical points where SPheno struggles to converge. This plot highlights that for complex Beyond Standard Model (BSM) scenarios, such as the $(B-L)$ SSM, the computational cost of HEP software must be considered during parameter scans.

3.2.3 Multiple constraints

PS methods face significant challenges when searching for regions of the parameter space that satisfy multiple experimental and theoretical constraints in BSM studies. The current knowledge of particle physics experimental data is systematically compiled by the Particle Data Group (PDG) [132], which provides the most up-to-date measurements and exclusion limits. These constraints are critical for validating BSM models and ensuring consistency with known observations.

The high dimensionality of the parameter space, combined with the complexity of multiple constraints, makes finding satisfactory model configurations particularly challenging. Multiple constraints narrow the acceptable parameter space to small, often disconnected regions, making viable solutions sparse.

3.2.4 High-dimensionality

High-dimensional parameter spaces often encounter the *curse of dimensionality*, a concept introduced in [133] that describes the exponential growth of volume with increasing dimensions. This phenomenon presents several challenges [134, 135, 136], including *sparsity*, where data points become increasingly sparse with the number of dimensions, making it difficult to identify statistically significant patterns; *computational complexity*, as algorithms that perform well in low-dimensional spaces often become intractable in high-dimensional ones; and issues with *distance metrics*, where the concept of proximity loses significance, affecting the performance of distance-based algorithms. To illustrate the sparsity challenge in the context of

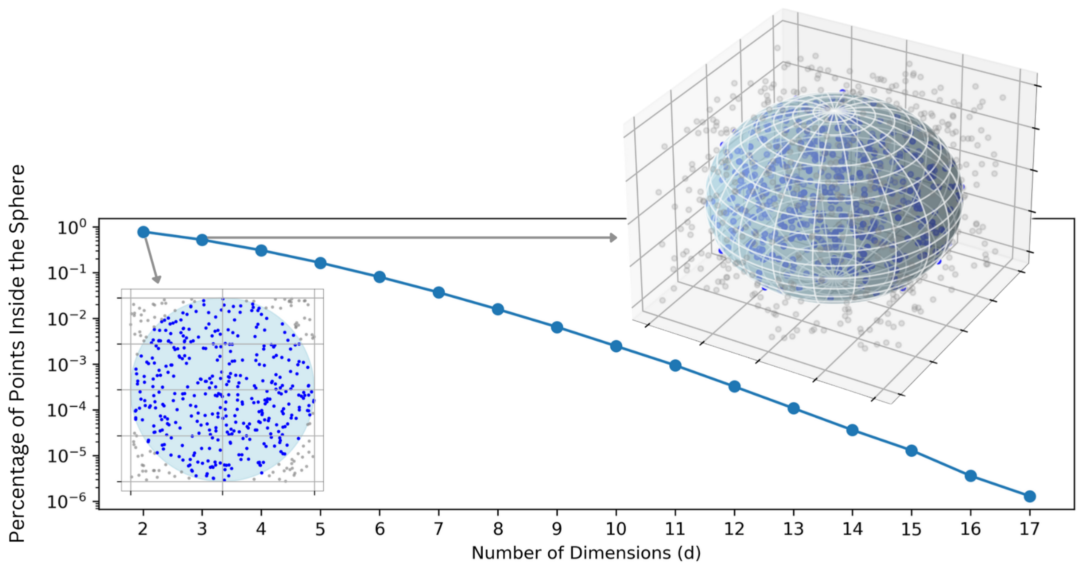


FIGURE 3.3: The relationship between space dimensionality and the percentage of uniformly generated points that lie inside the unit sphere. The main plot illustrates the rapid decay in the proportion of points within the sphere as the number of dimensions increases, highlighting the sparsity of high-dimensional spaces. Insets provide visual representations for specific cases: the 2D unit circle (left) and the 3D unit sphere (right), with points inside shown in blue and points outside shown in grey.

parameter scans, consider a two-dimensional parameter space of a model. Suppose the region within this space that yields physical observables matching certain experimental data corresponds precisely to the unit disk. To search for points that satisfy this criterion, we generate uniformly random points within the unit square. In two dimensions, the unit disk occupies a significant fraction of the unit square, making it relatively likely to generate satisfactory points.

However, as the dimensionality of the space increases, the situation becomes far more challenging. Figure 3.3 demonstrates the percentage of points that lie inside the hypersphere as the dimensionality increases. For each dimension, we generate 10^7 uniformly random points. In two dimensions, the percentage of points inside the sphere is approximately 80%, while in three dimensions, it decreases to 50%. For dimensions higher than eight, the percentage drops to less than 1%.

This demonstrates that the volume outside the hypersphere grows exponentially with the number of dimensions, making it exponentially harder to find points inside the unit hypersphere using a *uniform random sampling strategy.*

From a data analysis perspective [137], this phenomenon can be explained by the concept of *data orthogonality* in high-dimensional spaces. As the number of dimensions increases, data points tend to become nearly perpendicular to each other. This makes it harder to find meaningful patterns or correlations in the data. To achieve statistically reliable results, a much larger number of data points is needed compared to low-dimensional cases, which significantly increases computational and resource demands during parameter scans or similar analyses.

Chapter 4

Data Modelling

Data modelling is a fundamental aspect of Machine Learning (ML), involving the use of function approximators (FAs) to represent and understand complex relationships within data. FAs serve as surrogates for the original physics model, enabling approximate predictions of parameter space configurations at a significantly reduced computational cost. This approach allows a sequential decision-making algorithm to leverage these computationally efficient FAs to make informed decisions about where to target sampling. In this chapter, the focus is on regression techniques, specifically utilising Multi-Layer Perceptrons and Gaussian Processes as function approximators.

4.1 Regression with Machine Learning

Consider a dataset defined as,

$$\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \in \mathbb{R}^{N \times n}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} \in \mathbb{R}^{N \times m}, \tag{4.1}$$

where N is the number of data points. Here, $\mathbf{X}$ represents the data points in the input space, known as the *feature* matrix, where each row $\mathbf{x}_i \in \mathbb{R}^n$ is a feature vector of n dimensions corresponding to the i-th observation, $\mathbf{Y}$ represents the output *target* matrix, where each row $\mathbf{y}_i \in \mathbb{R}^m$ is the target output vector of m dimensions associated with the i-th observation.

The goal of regression is to construct a mapping function, known as the *model*, $f : \mathbb{R}^n \to \mathbb{R}^m$ that models the relationship between the inputs and outputs.

Specifically, the problem is to find a function f such that:

$$f\left(\mathbf{x}_i;\boldsymbol{\theta}\right) \approx \mathbf{y}_i, \quad \forall i \in \{1,2,\ldots,N\}, \tag{4.2}$$

where θ are the parameters of the model. ML approaches facilitate the construction of these models, with algorithms that automatically *learn* the parameters $\boldsymbol{\theta}$, also known as hyper-parameters, to best fit the observations in $\mathcal{D}$, without explicitly *hard-coding* the parameter values. This is achieved by minimising some predefined *loss function* $\mathcal{L}$, which quantifies the error between the predicted outputs $\hat{\mathbf{Y}} = f(\mathbf{X};\theta)$ and the true outputs. Typically, this involves solving:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{Y}, f(\mathbf{X};\boldsymbol{\theta})), \tag{4.3}$$

where $\boldsymbol{\theta}^*$ represents the optimal parameters of the mapping function f. The choice of the model and loss function depends on the regression task paradigms, two common paradigms are the *probabilistic* and the *deterministic.*

In probabilistic regression, the goal is to predict a distribution over possible outputs, rather than a single deterministic value. The mapping function $f\left(\mathbf{x}_i;\boldsymbol{\theta}\right)$ predicts parameters of the conditional probability distribution $p\left(\mathbf{y}_i \mid \mathbf{x}_i\right)$. The loss function in this context is often derived from the likelihood of the observed data, which quantifies how probable the true outputs are under the predicted distribution. The probabilistic approach naturally incorporates uncertainty, as it models not only the central tendency of the data but also the variance (or spread) of the predictions.

In deterministic regression, the goal is to directly predict the output $\mathbf{y}_i$ given the input $\mathbf{x}_i$. The mapping function $f\left(\mathbf{x}_i;\boldsymbol{\theta}\right)$ is assumed to provide a single value or vector that represents the predicted output. The loss function typically measures the error between the true outputs and the predicted values.

4.2 Gaussian Processes

Gaussian Processes (GPs) model the entire distribution of possible functions that can describe a given set of observations as a multivariate Gaussian distribution [52, 138]. This provides not only a point estimate of an objective but also quantifies the uncertainty associated with that estimate.

For a GP, datasets are considered as realisations of a stochastic process, where any finite subset of these variables has a joint Gaussian distribution[1]. A GP defines a

[1]A *stochastic process* [139], denoted as $\mathbf{f}(\mathbf{x})$, is a collection of random variables indexed by some parameter $\mathbf{x}$, which can represent time, space, or any other continuous or discrete domain. It is defined by a joint probability distribution for any finite N-dimensional set of values $\mathbf{f}\left(\mathbf{x}_1\right), \mathbf{f}\left(\mathbf{x}_2\right), \ldots, \mathbf{f}\left(\mathbf{x}_N\right)$

probability distribution over functions $f(\mathbf{x})$ which is specified completely by the mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ and can be written as,

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right) \tag{4.4}$$

with,

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k\left(\mathbf{x}, \mathbf{x}'\right) &= \mathbb{E}\left[\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)\right] \end{aligned}$$

where $\mathbf{x}$ are values in the input domain and $(\mathbf{x}, \mathbf{x}')$ all possible pairs of parameter values. The covariance function, also known as the *kernel function*, specifies how the output of the function at one input $\mathbf{x}$ covaries with the output at another input $\mathbf{x}'$. The choice of kernel function determines the smoothness, periodicity and other structural properties of the functions sampled from the GP.

4.2.1 Kernel functions

Many kernel functions have been studied [52], and examples of commonly used kernels are the Radial Basis Function (RBF), Matern and Periodic kernels. The RBF, also known as the Squared Exponential (SE) kernel, is widely used due to its simplicity and flexibility. It assumes that the function being modelled is smooth and is defined as:

$$k_{\mathrm{SE}}\left(x, x'\right) = \sigma^2 \exp\left(-\frac{\left\|x - x'\right\|^2}{2\ell^2}\right) \tag{4.5}$$

where σ^2 is the signal variance, controlling the amplitude of the variations and ℓ is the length-scale parameter, determining how quickly the correlation decays with distance. The Matern kernel is a generalisation of the RBF kernel, allowing for different degrees of smoothness. It is given as:

$$k_{\text{Matern}}\left(x, x'\right) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\left|x - x'\right|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}\left|x - x'\right|}{\ell}\right), \tag{4.6}$$

where ℓ is a parameter that controls the length-scale, $\Gamma(\nu)$ is the gamma function, K_ν is the modified Bessel function of the second kind and $\nu > 0$ controls the level of smoothness of the modified Bessel function. Common choices for ν are $\nu = 0.5, 1.5$, or 2.5, with $\nu = 0.5$ corresponding to an exponential kernel, and larger values of ν resulting in smoother functions.

Figure 4.1 shows prior function samples from a GP with the covariance matrix constructed using the SE and Matern kernel. Each function sample consists of 40 points, labelled by their index. The covariance matrix, displayed as a heatmap in the plot on the right, illustrates how the points covary with one another.

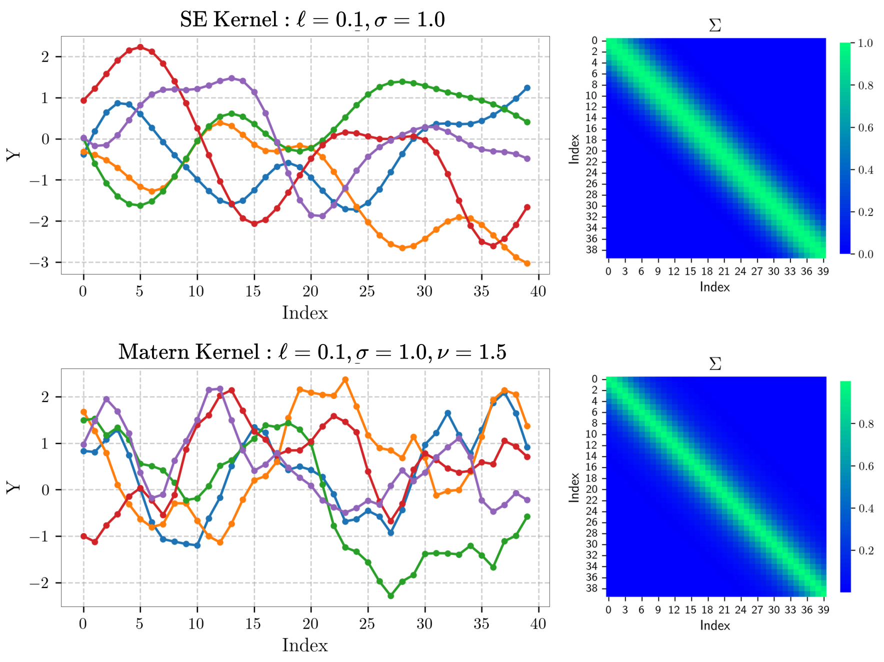


FIGURE 4.1: Prior function samples from a Gaussian Process (GP) using the Squared Exponential (SE) kernel (top) and the Matern kernel (bottom). Each function consists of 40 points indexed on the x-axis, with the corresponding covariance matrix shown as a heatmap on the right.

4.2.2 GP Predictions

A GP that fits an observed dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, can be used to predict new target values $\boldsymbol{Y}^*$ for new input points $\boldsymbol{X}^*$. GPs assume a joint Gaussian distribution for both sets of points, described as:

$$\begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{Y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{\boldsymbol{X}} \\ \mu_{\boldsymbol{X}^*} \end{bmatrix}, \begin{bmatrix} \Sigma_{\boldsymbol{X}\boldsymbol{X}} & \Sigma_{\boldsymbol{X}\boldsymbol{X}^*} \\ \Sigma_{\boldsymbol{X}^*\boldsymbol{X}} & \Sigma_{\boldsymbol{X}^*\boldsymbol{X}^*} \end{bmatrix} \right) \tag{4.7}$$

where $\Sigma_{\boldsymbol{X}\boldsymbol{X}}$ is the covariance matrix constructed using a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mu_{\boldsymbol{X}} \equiv \mu(\boldsymbol{X})$ is a shorthand notation, also used on Σ. Multivariate Gaussians are closed under conditioning, meaning that the conditional distribution of any subset of variables, such as $\mu_{\boldsymbol{X}}$, given another subset, such as $\mu_{\boldsymbol{X}^*}$, is also a multivariate Gaussian distribution. Specifically:

$$\mu_{\boldsymbol{X}} \mid \mu_{\boldsymbol{X}^*} \sim \mathcal{N}\left(\mu_{\boldsymbol{X}|\boldsymbol{X}^*}, \Sigma_{\boldsymbol{X}|\boldsymbol{X}^*}\right)$$

where the conditional mean and covariance are given as [140]

$$\begin{aligned} \mu_{\boldsymbol{X}^*|\boldsymbol{X}} &= \mu_{\boldsymbol{X}^*} + \Sigma_{\boldsymbol{X}^*\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} (\boldsymbol{Y} - \mu_{\boldsymbol{X}}) \\ \Sigma_{\boldsymbol{X}^*|\boldsymbol{X}} &= \Sigma_{\boldsymbol{X}^*\boldsymbol{X}^*} - \Sigma_{\boldsymbol{X}^*\boldsymbol{X}} \Sigma_{\boldsymbol{X}\boldsymbol{X}}^{-1} \Sigma_{\boldsymbol{X}\boldsymbol{X}^*} \end{aligned} \tag{4.8}$$

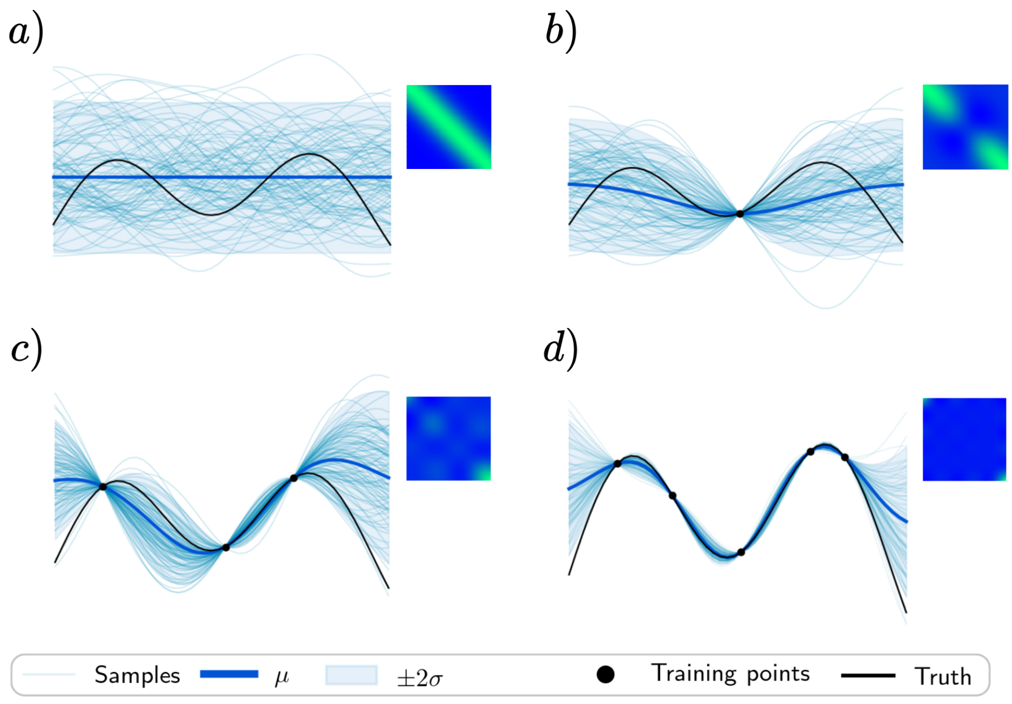


FIGURE 4.2: The behaviour of GP regression as the number of training points increases. The posterior mean (μ, blue curve) progressively aligns with the true function (black curve) as more training points (black dots) are added. Predictive uncertainty ($\pm 2\sigma$) decreases near observed points and across the input space with increasing data density. Insets show the covariance matrices (Σ), transitioning from a broad diagonal band in the prior to a constrained structure, where uncertainty diminishes near observed points.

In general, the prior mean is often assumed to be zero, $\mu_{\boldsymbol{X}} = \mu_{\boldsymbol{X}^*} = 0$, allowing the GP to only use the covariance function to capture the structure of the data. The predicted values for $\boldsymbol{Y}^*$ can then be estimated using the predictive mean equation from (4.8) for each input point $\boldsymbol{X}^*$. The variance $\sigma^2_{\boldsymbol{X}^*}$, of these predictions is given by the diagonal entries of the predictive covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{X}^*|\boldsymbol{X}}$.

Figure 4.2 illustrates the behaviour of GP regression as the number of training points increases, highlighting its impact on the posterior mean, uncertainty, and covariance structure. The posterior mean (μ), shown as a blue curve, progressively aligns with the true function (black curve) as more training points (black dots) are incorporated. Predictive uncertainty ($\pm 2\sigma$) decreases near observed points and across the input space with increasing data density. The covariance matrices (Σ) displayed as insets, demonstrate a transition from a broad diagonal band in the prior to a constrained covariance structure, where the predictive uncertainty decreases in regions close to the observed points.

Modelling noise

GPs can account for noisy datasets by explicitly incorporating a noise term into the model. This noise term assumes that the observed data includes both the underlying signal and additive noise. The noise term is added to observations as $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ where $\mathbf{f}$ is the latent function (true underlying signal) and $\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \sigma_n^2 \mathbf{I}\right)$ is the noise term, with σ_n^2 representing the variance of the noise, which is taken as a hyper-parameter. This implies that the covariance matrix of the observed data includes a noise variance term on the diagonal:

$$\Sigma_{\boldsymbol{XX}} \to \Sigma_{\boldsymbol{XX}} + \sigma_n^2 \mathbf{I}.$$

The noise term modifies the posterior mean and covariance in equation (4.8) to account for the uncertainty introduced by the noise.

4.2.3 Model selection

Kernel functions introduce a set of hyperparameters that govern the structure of the covariance function, such as the length scale, variance, and noise variance (if applicable). Proper selection of these parameters is essential for ensuring the accuracy and performance of GP regression. These optimal parameters are typically determined by maximising the marginal likelihood of the observed data.

For a dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, the marginal likelihood is the probability of the observed outputs $\boldsymbol{Y}$ given the inputs $\boldsymbol{X}$ and the hyper-parameters of the kernel. It is expressed as:

$$p(\boldsymbol{Y} \mid \boldsymbol{X}, \theta) = \mathcal{N}\left(\boldsymbol{Y} \mid \mu_{\boldsymbol{X}}, \Sigma_{\boldsymbol{XX}}\right) \tag{4.9}$$

where θ denotes the set of hyperparameters, $\mu_{\boldsymbol{X}}$ is the mean vector (often assumed to be zero in a standard GP prior), and Σ_{XX} is the covariance matrix constructed using the kernel function. Therefore, the likelihood is expressed as:

$$p(\boldsymbol{Y} \mid \boldsymbol{X}, \theta) = \frac{1}{(2\pi)^{n/2} \left|\Sigma_{\boldsymbol{XX}}\right|^{1/2}} \exp\left(-\frac{1}{2} \left(\boldsymbol{Y} - \mu_{\boldsymbol{X}}\right)^{\top} \boldsymbol{\Sigma}_{\boldsymbol{XX}}^{-1} \left(\boldsymbol{Y} - \mu_{\boldsymbol{X}}\right)\right) \tag{4.10}$$

For computational convenience, the logarithm of the marginal likelihood (LML) is often used:

$$\log p(\boldsymbol{Y} \mid \boldsymbol{X}, \theta) = -\frac{1}{2} \left(\boldsymbol{Y} - \mu_{\boldsymbol{X}}\right)^{\top} \Sigma_{\boldsymbol{XX}}^{-1} \left(\boldsymbol{Y} - \mu_{\boldsymbol{X}}\right) - \frac{1}{2} \log \left|\Sigma_{\boldsymbol{XX}}\right| - \frac{n}{2} \log 2\pi \tag{4.11}$$

where n is the number of training points. In this expression the first term quantifies how well the model fits the data, the second term penalises the model complexity, preventing over fitting and the third term is a constant term that depends only on n. The optimal hyper-parameters θ are obtained by maximising the log marginal likelihood:

$$\theta^* = \arg\max_{\theta} \log p(\boldsymbol{Y} \mid \boldsymbol{X}, \theta)$$

This optimisation is typically performed using gradient-based methods, as the derivatives of the LML with respect to the hyper-parameters can be computed efficiently.

The LML can be interpreted as a loss function, but it is fundamentally different from a manually imposed loss function like Mean Squared Error in common regression problems. It arises naturally from the probabilistic model where the data is assumed to follow a Gaussian distribution.

4.2.4 Space dimensionality

GPs inherently support vector-valued inputs, where each input $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional vector. This flexibility arises from the kernel function, $k(\mathbf{x}_i, \mathbf{x}_j)$, which computes the similarity between any two input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, regardless of their dimensionality.

GPs can also model vector-valued functions, where the data maps an input $\mathbf{x} \in \mathbb{R}^d$ to an output vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^q$, with q representing the dimensionality of the output. A straightforward approach is to model each output dimension independently, which is the method adopted in this research project for later chapters. However, GPs can also be extended to capture correlations between different output dimensions, enabling more expressive models [141].

4.2.5 Challenges

GPs are highly flexible models, but their computational complexity presents a significant challenge, especially for large datasets. The key computational bottleneck arises from the need to invert the $n \times n$ covariance matrix during training, which scales as $O(n^3)$, and the subsequent prediction step, which scales as $O(mn^2)$ for m test points. This makes exact GP inference and prediction infeasible for datasets with thousands or millions of points. To address this, various approaches have been developed to make GPs more scalable [142].

4.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are parametric models inspired by the structure and functioning of the human brain. ANNs consist of interconnected units called neurons or nodes, which are organised into layers. These layers work together in a hierarchical manner, to process data and perform tasks such as regression or classifications. The number of layers of a ANN is known as the depth of the model, the larger this number the deeper it is, giving the name of the class of models as Deep Learning models. Learning comes from the way the parameters are optimised, a loss function is defined based on how well the model, either learns the correct mapping for a given dataset for regression or classify accurately. The parameters are optmised in order to minimise this loss.

Multi-Layer Perceptrons (MLPs) are a class of ANNs that are among the most fundamental architectures in deep learning, also known as Fully-Connected Forward Networks. To understand neural networks, is useful to start with Linear Regression, one of the simplest form of modelling.

4.3.1 Linear Regression

When the dataset presents a linear trend, linear regression models the relationship between an input vector $\mathbf{x} \in \mathbb{R}^n$ and an output vector $\mathbf{y} \in \mathbb{R}^m$ using a linear mapping[2]:

$$\hat{y}_i = w_{ij} x_j + b_i, \tag{4.12}$$

where w_{ij} are the elements of the weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, y_i and x_i are the components of $\mathbf{y}$ and $\mathbf{x}$, respectively, and b_i is the i-th component of $\mathbf{b}$.

The objective is to find the weights $\mathbf{W}$ and biases $\mathbf{b}$ such that the predicted $\hat{\mathbf{y}}$ closely approximates the true target values $\mathbf{y}$. This is typically achieved by minimising a loss function, such as the mean squared error (MSE):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 . \tag{4.13}$$

In this equation, i denotes the data index, and N is the number of data points. In linear regression, the minimisation condition can be obtained analytically by taking $\nabla_w \mathcal{L} = 0$. However, in practice, a general data-driven approach often optimises the parameters using gradient information. This is achieved through gradient descent, an iterative algorithm that updates the parameters in the direction of the negative gradient of the loss function. The gradient descent update for each parameter $\theta \in \{\mathbf{W}, \mathbf{b}\}$ is:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}, \tag{4.14}$$

where $\eta > 0$ is the learning rate, controlling the step size of the updates. This process is repeated until the loss converges or a maximum number of iterations is reached.

[2]Einstein summation convention is used, for example, $w_{ij} x_j = \sum_j w_{ij} x_j$.

4.3.2 Shallow Neural Networks

Linear mappings are insufficient for when a given dataset, as defined in equation (4.1), exhibits non-linear dependencies. Shallow neural networks (SNN) address this limitation by extending linear regression through the incorporation of non-linearities.

A SNN introduces a non-linear *activation function*, denoted as $\sigma(\cdot)$, which transforms the output of a linear combination of the input vector. This transformation is expressed as $h = \sigma(z)$, where:

$$z_k = w_{kj}^{(1)} x_j + b_k^{(1)} \tag{4.15}$$

Here, $k = 1, \ldots, N_h$ indexes the nodes of the *hidden layer*, and $w_{kj}^{(1)}$ are the elements of the weight matrix $\mathbf{W}^{(1)}$. The *hidden layer* output is subsequently mapped to the predicted output $\hat{y}_i$ through:

$$\hat{y}_i = w_{ik}^{(2)} h_k + b_i^{(2)} \tag{4.16}$$

where $w_{ik}^{(2)}$ are the elements of the weight matrix $\mathbf{W}^{(2)}$. These weights define the dependencies between the hidden layer's nodes and the desired output vector $\hat{\mathbf{y}}$. By expanding h_k, the output can be rewritten as:

$$\hat{y}_i = w_{ik}^{(2)} \sigma\left(w_{kj}^{(1)} x_j + b_k^{(1)}\right) + b_i^{(2)}. \tag{4.17}$$

The non-linearity introduced by σ enables the network to model complex non-linear patterns in the data, capturing intermediate features or transformations critical for predicting the target output. An illustration of a SNN is shown in Figure 4.3.

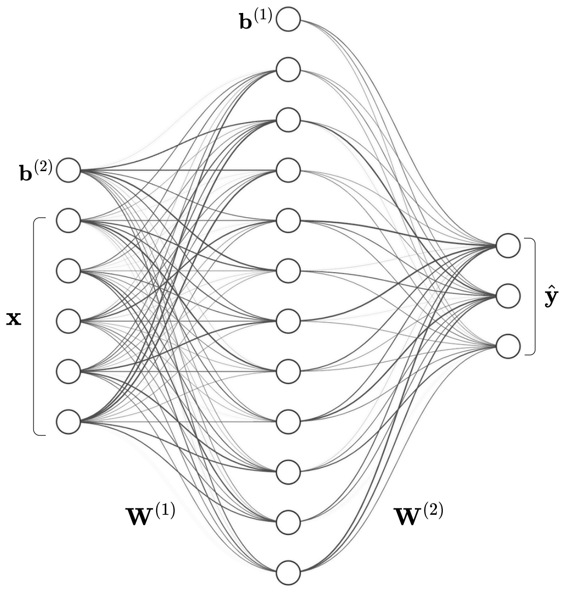


FIGURE 4.3: Representation of a shallow neural network (SNN) with a single hidden layer. Input features $\mathbf{x}$ are transformed through weights $\mathbf{W}^{(1)}$ and biases $\mathbf{b}^{(1)}$ into the hidden layer, which is further processed with weights $\mathbf{W}^{(2)}$ and biases $\mathbf{b}^{(2)}$ to produce the predicted output $\hat{\mathbf{y}}$.

The mean squared error (MSE) loss function, as defined in Equation (4.13), remains applicable. Gradient descent update rules from Equation (4.14) are used to optimise the parameters of the model, which include the weight matrices and biases, $\theta \in \left\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}\right\}$. The gradients are computed using the chain rule, which propagates the error through the layers. For the loss function $\mathcal{L}$, the gradients with respect to the parameters are:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_{ik}^{(\alpha)}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}_l} \cdot \frac{\partial \hat{y}_l}{\partial w_{ik}^{(\alpha)}} \\ \frac{\partial \mathcal{L}}{\partial b_i^{(\alpha)}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}_l} \cdot \frac{\partial \hat{y}_l}{\partial b_i^{(\alpha)}} \end{aligned} \tag{4.18}$$

where $\alpha = 1, 2$. The individual derivatives are computed as follows:

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial w_{ik}^{(2)}} &= \sigma\left(w_{kj}^{(1)} x_j + b_k^{(1)}\right), & \frac{\partial \hat{y}_i}{\partial b_i^{(2)}} &= 1, \\ \frac{\partial \hat{y}_i}{\partial w_{kj}^{(1)}} &= w_{ik}^{(2)} \cdot \sigma'\left(w_{kj}^{(1)} x_j + b_k^{(1)}\right) \cdot x_j, & \frac{\partial \hat{y}_i}{\partial b_k^{(1)}} &= w_{ik}^{(2)} \cdot \sigma'\left(w_{kj}^{(1)} x_j + b_k^{(1)}\right). \end{aligned} \tag{4.19}$$

Here, $\sigma'(\cdot)$ represents the derivative of the activation function $\sigma(\cdot)$. These expressions allow us to evaluate a single input, compute the loss function given the current parameter values $\boldsymbol{\theta}$, and calculate the derivatives to propagate the loss information back through the network. This propagation enables the adjustment of each parameter based on the error for that input. This procedure is known as back-propagation.

4.3.3 Multi-layer Perceptron

MLPs generalise SNNs by introducing multiple hidden layers between the input and output layers. These additional layers define the architecture of the MLP and allow them to model even more complex patterns in data by using hierarchical non-linear transformations by using the activation functions $\sigma(\cdot)$ in each layer.

For an MLP with L layers, the output at layer α is computed as:

$$h_k^{(\alpha)} = \sigma\left(w_{kj}^{(\alpha)} h_j^{(\alpha-1)} + b_k^{(\alpha)}\right) \tag{4.20}$$

where α indexes the layers, $w_{kj}^{(\alpha)}$ and $b_k^{(\alpha)}$ are the weight matrix components and bias vector for layer α and $h^{(0)} = \mathbf{x}$ represents the input vector. The final output of the MLP is given by:

$$\hat{y}_k = w_{kj}^{(L+1)} h_j^{(L)} + b_k^{(L+1)}, \tag{4.21}$$

where $w_{kj}^{(L+1)}$ and $b_k^{(L+1)}$ are components of $\mathbf{W}^{(L+1)}$ and $\mathbf{b}^{(L+1)}$, respectively, which map the last hidden layer's output to the predictions.

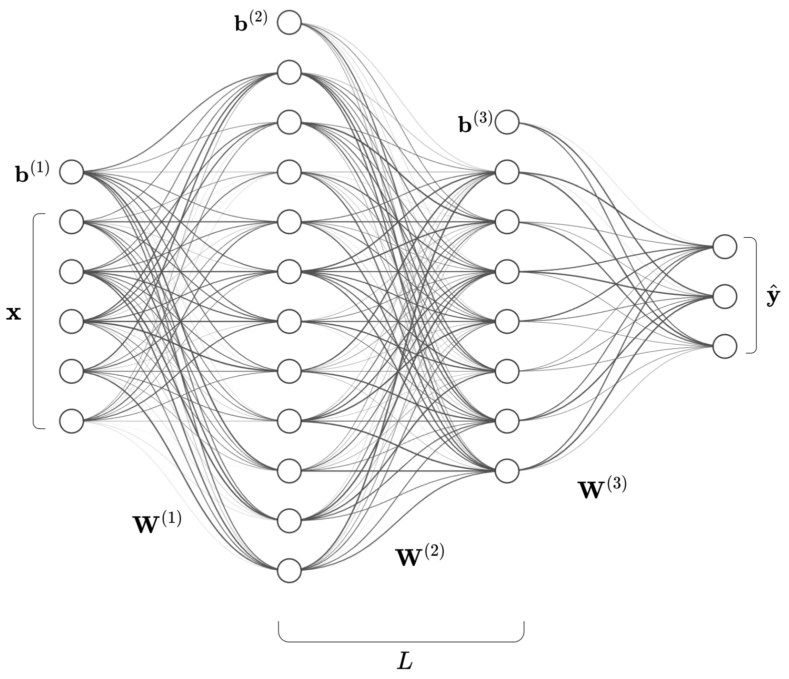


FIGURE 4.4: Illustration of a deep neural network (DNN) or Multi-Layer Perceptron (MLP) with multiple hidden layers, with a fully connected architecture. The input features $\mathbf{x}$ are successively transformed through layers using weights $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$ and biases $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$, $\mathbf{b}^{(3)}$. The network outputs the prediction $\hat{\mathbf{y}}$, with the depth of the network represented by L, the total number of hidden layers.

4.3.4 Training MLPs

Training MLPs follows the same fundamental principles as SNNs but involves additional computational complexity due to the deeper architecture. Backpropagation, extended to handle multiple layers, computes the gradients of the loss function with respect to all parameters $\boldsymbol{\theta} = \mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}$ for $\ell = 1, \ldots, L+1$. These gradients are used to iteratively update the parameters via gradient descent or its variants. The use of deeper networks with multiple non-linear transformations enables MLPs to approximate any continuous function, a property known as the **universal approximation theorem**.

4.3.5 Activation Functions

The activation function $\sigma(\cdot)$ is a critical component of an ANN, as it introduces non-linear transformations that enable the network to capture patterns in data. Various activation functions have been studied in the literature [143].

The sigmoid function, given by,

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{4.22}$$

was one of the first activation functions used in ANNs. It maps input values to the range $(0, 1)$, making it particularly useful for probabilistic interpretations. Its smooth shape made it suitable for binary classification problems. However, the sigmoid function suffers from the vanishing gradient problem, where the derivative approaches zero for large positive or negative inputs. This slows down learning in deep networks. The Rectified Linear Unit (ReLU), defined as

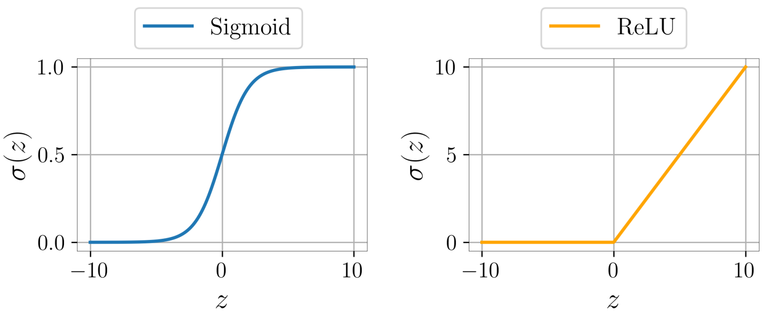


FIGURE 4.5: Comparison of the Sigmoid and ReLU activation functions. The Sigmoid function smoothly transitions values between 0 and 1, while the ReLU function outputs zero for negative inputs and scales linearly for positive inputs.

$$\text{ReLU}(z) = \max(0, z), \tag{4.23}$$

is the most commonly used activation function in modern neural networks due to its simplicity and effectiveness. Unlike the sigmoid function, ReLU introduces sparsity by outputting zero for negative inputs while maintaining linearity for positive inputs. This behaviour enables efficient gradient flow, computational efficiency, and better feature extraction, which enhances learning in deep networks. Figure 4.5 illustrates both activation functions.

However, ReLU can encounter the dying neuron problem, where certain neurons output zero for all inputs and become inactive. Variants like Leaky ReLU and Parametric ReLU address this issue by allowing small gradients for negative inputs.

4.3.6 Modern optimisers

Gradient Descent randomly initialises the parameters θ and updates them by computing the gradients of the loss function $\mathcal{L}$ over the entire dataset. While effective for small datasets, this method is computationally expensive for larger ones and highly dependent on the initial starting point. Stochastic Gradient Descent (SGD) addresses these issues by computing gradients using a random subset (batch) $\mathcal{B}$ of the training data. Then, the parameter update rule in SGD is given by:

$$\theta \leftarrow \theta - \eta \sum_{i \in \mathcal{B}_i} \frac{\partial \mathcal{L}}{\partial \theta_i}, \tag{4.24}$$

Modern optimisers [144], such as ADAM, further enhance performance by incorporating adaptive learning rate techniques.

4.4 Modelling Techniques Comparison

A comparison of modelling techniques, including linear regression, SNNs, MLPs, and GPs, applied to a synthetic two-dimensional input and one-dimensional output dataset, is presented in Figure 4.6. The ground truth function is shown in panel (a) and is defined as:

$$y = \sin(2\pi x_1) \cdot \cos(2\pi x_2) \tag{4.25}$$

with 200 input samples $x \in \mathbb{R}^2$ drawn from a uniform distribution over $[-1, 1]^2$. The

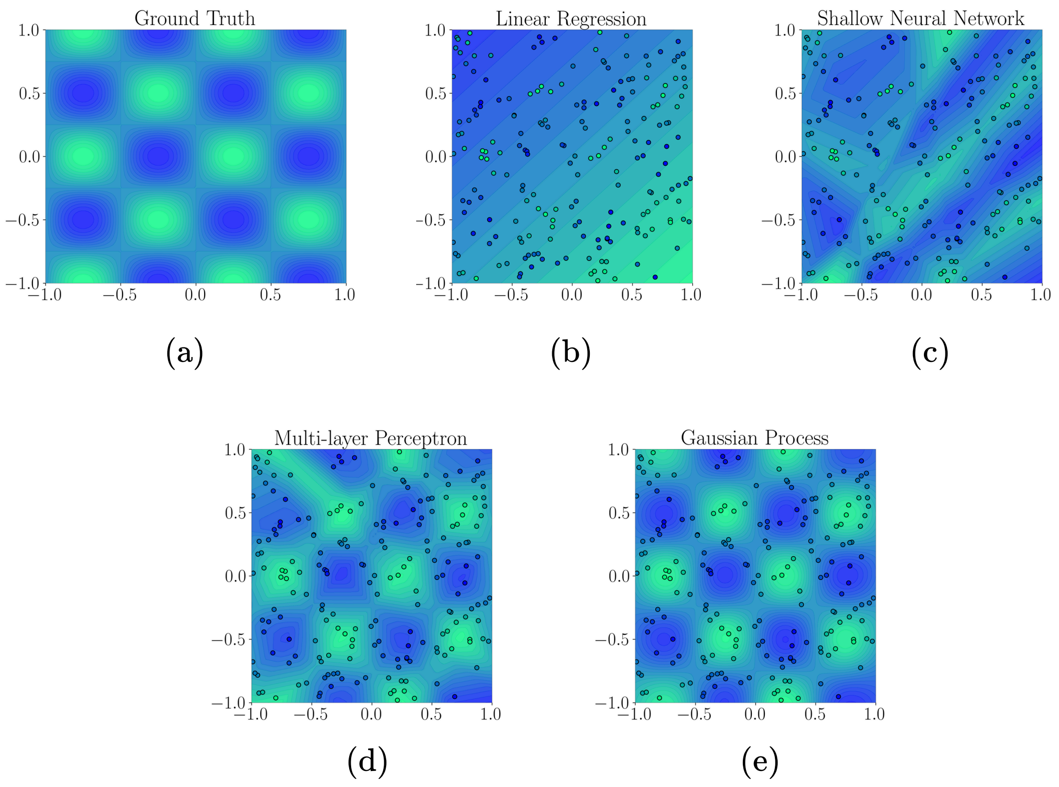


Figure 4.6: Comparison of modelling techniques with 200 training samples drawn from a uniform distribution. (a) Ground truth function. (b) Linear regression. (c) Shallow neural network (SNN). (d) Multi-layer perceptron (MLP). (e) Gaussian Process (GP).

modelling techniques are compared as follows:

- **Linear Regression**: Panel (b) illustrates the results of linear regression. As expected, this model cannot reproduce the oscillatory patterns of y due to its assumption of linearity.
- **Shallow Neural Networks** : Panel (c) demonstrates an SNN with a single hidden layer containing 16 nodes and ReLU activation functions. The model

captures some non-linear patterns but is limited in its representational capabilities.

- **Multilayer Perceptrons**: Panel (d) shows an MLP with two hidden layers, each containing 32 nodes and ReLU activations. MLPs can model the underlying patterns of the ground truth function more effectively than SNNs. However, even for this relatively simple two-dimensional example, MLPs require substantial data to perform satisfactorily, highlighting their data dependency.
- **Gaussian Processes**: Panel (e) illustrates GPs, which excel at capturing the entire structure of the ground truth function with fewer data points. This efficiency arises from the prior assumptions encoded in the kernel function, allowing GPs to infer patterns more effectively from limited data. However, these assumptions, referred to as inductive bias, can themselves pose challenges if they are poorly chosen or misaligned with the underlying data.

4.4.1 Theory to practice

As discussed in the previous section, the gradient flow of the loss function $\mathcal{L}$ with respect to the parameters θ is crucial for the optimisation of ANNs. The practical implementation of ANNs has been revolutionised by automatic differentiation frameworks, which automate the computation of these gradients.

In this work, PyTorch [145] is utilised to implement ANNs. PyTorch is a widely used, open-source machine learning library that provides robust support for tensor computation and automatic differentiation.

For GPs, we use both GPyTorch [146] and Scikit-Learn [147]. GPyTorch is a specialised library for GPs that seamlessly integrates with PyTorch. It uses PyTorch's functionalities to provide scalable and efficient implementations of GPs, particularly for large datasets. This integration ensures compatibility with PyTorch's GPU acceleration. Scikit-Learn is used for its baseline implementations of GPs, allowing for quick experimentation and prototyping. For instance, in panel (d) of Figure 4.6, Scikit-Learn's GP regression functionality was utilised.

Additionally, Scikit-Learn serves as a valuable tool for general machine learning pipeplines, offering utilities such as dataset transformation and normalisation.

4.5 Density Estimation

Since this work focuses on modelling techniques that guide parameter scan algorithms, estimating density over the available data in the search space is also an option. As the last modelling technique, Kernel Density Estimation (KDE) [148], a baseline density estimation method, is introduced, KDE is a non-parametric method for estimating the

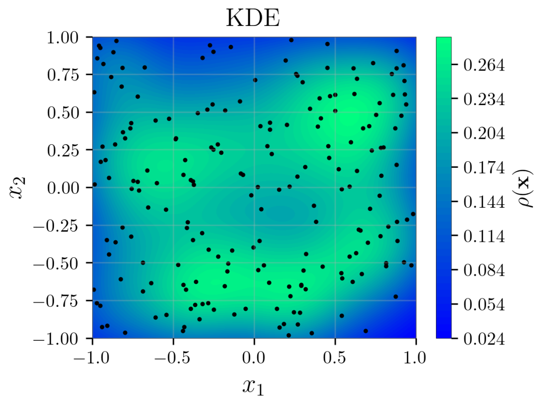


FIGURE 4.7: Kernel Density Estimation (KDE) for the modelling test example for the ground truth equation (4.25)

PDF of a random variable based on observed data. KDEs are essentially smooth histograms, useful for visualising data distributions without making assumptions about their underlying form. For multivariate data $\mathbf{x} \in \mathbb{R}^d$, the density $\rho(\mathbf{x})$ is expressed as:

$$\rho(\mathbf{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{4.26}$$

where N is the number of data points, h is the bandwidth parameter and h^d is the bandwidth scaling factor for d-dimensional space. The parameter h controls the smoothness of the density estimate by determining the influence range of each data point. A larger h results in a smoother estimate across a broader region, while a smaller h captures finer details but risks over-fitting. The h^d scaling ensures normalisation of the density estimate by accounting for the increased volume of d-dimensional kernels. Challenges of KDEs include the selection of the bandwidth parameter and high-dimensional spaces, frequently requiring dimensionality reduction techniques. Figure 4.7 shows the density estimation for the modelling test example for the ground truth equation (4.25).

4.6 Modelling Takeaway*

GPs benefit from prior knowledge encoded through kernel functions, which can be combined to represent complex patterns. This characteristic makes GPs highly data-efficient, as they can generalise effectively with relatively small datasets.

An additional advantage of GPs is their ability to model uncertainty in the output space. This uncertainty provides valuable information about the noise in the data or, in cases where the data represents a deterministic yet expensive-to-evaluate function (as is the case in this work), highlights unexplored regions in the input space. This feature makes GPs particularly promising for parameter scanning algorithms and even for general optimisation tasks.

MLPs, and ANNs more broadly, are versatile as they do not assume specific data patterns but instead learn them directly from the data. This flexibility comes at the cost of requiring more data for training. Furthermore, ANNs support various architectures, enabling tasks beyond regression, such as dimensionality reduction with auto-encoders [149] or generative modelling with networks architectures like Generative Adversarial Networks [150]. These approaches are especially powerful for large datasets involving images, videos, text, or audio. However, such tasks are outside the scope of this thesis.

In this work, we focus on the small-data regime, characterised by relatively low dimensional inputs ($D \lesssim 20$) and a modest number of data points ($N \lesssim 10^4$). This emphasis on data efficiency allows both MLPs and GPs to remain competitive in terms of representational capabilities within this context.

Chapter 5

Sampling and Decision Making

Sampling refers to the process of collecting the right information, while decision making focuses on using that information to choose the best actions. In this chapter, we review the foundations of baseline sampling algorithms, which are well-established for performing parameter space scans. Following this, we explore decision-making frameworks, specifically Bayesian Optimisation and Reinforcement Learning, both of which offer powerful approaches for adaptive and informed exploration.

5.1 Classical Sampling Techniques

Sampling involves generating random samples from a probability distribution, and the goal of sampling algorithms is for the empirical sample distribution to converge to the true underlying probability distribution as the number of samples increases. The resulting samples, are crucial for estimating properties of distributions such as mean and variance, simulate stochastic processes, approximate numerical integration or optimisation problems, serving as a fundamental tool in particle physics [39, 151] and science in general [152].

Specifically, sampling refers the process of drawing K parameter configurations $\{\theta_k\}_{k=1}^K$ from a probability density function (PDF) $p(\theta)$. This PDF, can be approximated by the samples by the *empirical distribution* $\hat{p}_K(\theta)$, defined as:

$$\hat{p}_K(\theta) = \frac{1}{K}\sum_{k=1}^{K} \delta\left(\theta - \theta_k\right), \tag{5.1}$$

where $\delta(\cdot)$ is the Dirac delta function, which places all the probability mass at the sampled points $\{\theta_k\}_{k=1}^K$. When $K \to \infty$, the empirical distribution $\hat{p}_K(\theta)$ converges to the true distribution $p(\theta)$:

$$\hat{p}_K(\theta) \to p(\theta), \quad \text{as } K \to \infty. \tag{5.2}$$

While this holds inherently for *independent and identically distributed* (i.i.d.) sampling, other methods may require additional assumptions. For practical purposes, this empirical distribution is often visualised using a histogram, which provides a binned approximation of the sampled data.

Using the sampled $\{\theta_k\}$ mean and variance can be estimated by,

$$\mathbb{E}_{p(\theta)}[\theta] \approx \frac{1}{K}\sum_{k=1}^{K}\theta_k, \quad \mathrm{Var}_{p(\theta)}[\theta] \approx \frac{1}{K}\sum_{k=1}^{K}\left(\theta_k - \bar{\theta}\right)^2, \tag{5.3}$$

where $\bar{\theta} = \frac{1}{K}\sum_{k=1}^{K}\theta_k$, in general for any function $f(\theta)$, the expectation under $p(\theta)$ can be approximated as:

$$\mathbb{E}_{p(\theta)}[f(\theta)] \approx \frac{1}{K}\sum_{k=1}^{K} f\left(\theta_k\right). \tag{5.4}$$

In the context of parameter scans, approximating and understanding the underlying distribution is crucial because it corresponds to the *likelihood.* The *likelihood* is a function of the parameters that quantifies how well the observables predicted by a model align with experimental constraints. More details about the *likelihood* in the context of particle physics are provided in the next chapter.

Challenges

While direct sampling is ideal for estimating expectations and properties of a distribution, it is often impractical or infeasible for complex distributions. Many distributions $p(\theta)$ are badly normalised, meaning they are defined only up to a normalising constant:

$$p(\theta) = \frac{\tilde{p}(\theta)}{Z}, \quad Z = \int \tilde{p}(\theta)d\theta \tag{5.5}$$

where Z is the normalising constant, unknown and difficult to compute. Additionally, as shown in the previous chapter, sampling methods also suffer from the *curse of dimensionality*, wherein the probability mass often concentrates in small regions of the parameter space, requiring specialised sampling techniques to explore these regions effectively.

5.1.1 Rejection Sampling

Rejection sampling (RS) [153] is one of the simplest Monte Carlo methods to sample from a complex target distribution $p(x)$. It relies on a proposal distribution $q(x)$,

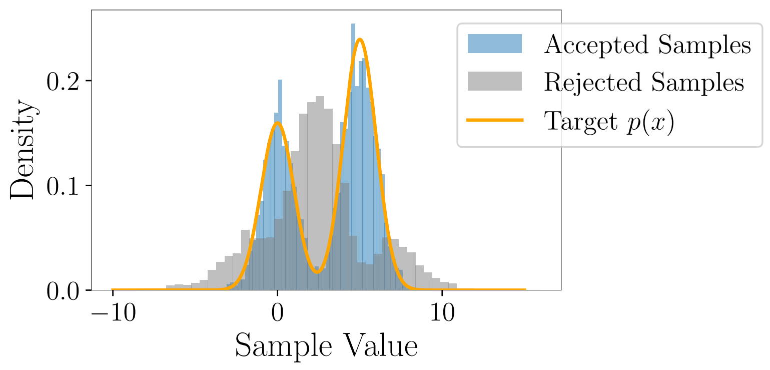


FIGURE 5.1: Rejection sampling example on a one-dimensional target distribution, showing density histograms of accepted (blue) and rejected (grey) samples, with the target distribution $p(x)$ depicted in orange.

which is easy to sample from, to approximate $p(x)$. The proposal distribution needs to cover the target distribution and satisfy the condition:

$$Mq(x) \geq p(x) \tag{5.6}$$

for all x, where $M > 1$ is a scaling factor. The algorithm works as follows,

1. Generate a sample $x \sim q(x)$.
2. Generate a uniform random number $u \sim \text{Uniform}(0, 1)$.
3. Accept x if $u \leq \frac{p(x)}{Mq(x)}$.
4. Repeat until enough samples are accepted.

Here, $\frac{p(x)}{Mq(x)}$ is the acceptance probability. The scaling parameter M controls the trade-off between validity and efficiency. A large M ensures that the inequality condition in equation (5.6) holds for every x but results in many rejected samples, reducing the algorithm's efficiency. A small M increases the acceptance rate, improving efficiency, but requires careful tuning to avoid violating the inequality condition.

Determining M is one of the difficulties in RS [153]. Additionally, RS may struggle when the target distribution exhibits multiple modes or sharp peaks, features commonly encountered in particle physics phenomenology studies.

An example of RS on a one-dimensional target density is shown in Figure 5.1. The target distribution is constructed as a mixture of two Gaussian distributions, defined as:

$$p(x) = 0.4 \cdot \mathcal{N}(x \mid \mu_1 = 0, \sigma_1 = 1) + 0.6 \cdot \mathcal{N}(x \mid \mu_2 = 5, \sigma_2 = 1). \tag{5.7}$$

The proposal distribution used is a Gaussian $\mathcal{N}(0, 10)$, chosen to adequately cover both modes of the target density. In this experiment, 3,000 accepted samples were required, which required drawing $9,522$ samples from the target distribution. This corresponds to a simple efficiency metric, calculated as:

$$\text{Efficiency} = \frac{N_{\text{accepted}}}{N_{\text{drawn}}} \times 100 \tag{5.8}$$

resulting in an efficiency of 31.51%. If querying the target distribution is computationally expensive, efficiency becomes critical, as lower efficiency implies more wasted samples and higher computational cost.

5.1.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) [41] methods are a powerful class of algorithms for sampling from a complex target distribution $p(x)$, particularly those that are high-dimensional or unnormalised. The key idea behind MCMC is to construct a Markov chain that has the desired target distribution $p(x)$ as its *stationary distribution*, denoted by $\pi(x)$. By running the chain for a sufficient number of iterations, the samples generated by the chain approximate the target distribution.

A Markov chain is a sequence of states $\{x_t\}_{t=0}^{\infty}$, where the probability of transitioning to the next state depends only on the current state:

$$p\left(x_{t+1} \mid x_t, x_{t-1}, \ldots\right) = p\left(x_{t+1} \mid x_t\right). \tag{5.9}$$

This is known as the *Markov property* and simplifies the sampling process while ensuring flexibility and localisation in exploring the state space. The *stationary distribution* $\pi(x)$ of the Markov chain is the distribution that remains invariant under the chain's transitions:

$$\pi\left(x'\right) = \int \pi(x) p\left(x' \mid x\right) dx. \tag{5.10}$$

MCMC methods are designed to construct a Markov chain whose *stationary distribution* is the desired target distribution.

5.1.2.1 Metropolis Hastings

The Metropolis-Hastings (MH) algorithm is a cornerstone of the MCMC family of methods [39, 153]. The MH algorithm begins by selecting an initial state x_0 from the state space. Following initialisation, at each iteration t the algorithm performs the following steps:

1. **Proposal Generation**: A candidate state $\boldsymbol{x}'$ is proposed based on a proposal distribution $q\left(x' \mid x_t\right)$.

2. **Acceptance Probability**: The candidate state p' is evaluated using the acceptance probability:

$$\alpha\left(x', x_t\right)=\min \left(1, \frac{p\left(x'\right) q\left(x_t \mid x'\right)}{p\left(x_t\right) q\left(x' \mid x_t\right)}\right), \tag{5.11}$$

   where $p(x)$ is the target distribution.

3. **Acceptance or Rejection**: A uniform random number $u \sim \operatorname{Uniform}(0,1)$ is drawn. If $u \leq \alpha\left(x', x_t\right)$, the candidate x' is accepted, and the next state is set as $x_{t+1}=x'$. Otherwise, the candidate is rejected, and the chain remains in its current state: $x_{t+1}=x_t$.

This process is repeated until a sufficient number of samples are collected to approximate the target distribution.

A common choice for the *proposal distribution* $q\left(x' \mid x_t\right)$ is a Gaussian centred on the current state x_t, with variance σ^2. This choice simplifies the acceptance probability when $q\left(x' \mid x_t\right)=q\left(x_t \mid x'\right)$, satisfying the balance condition, as the proposal terms cancel out. However, the variance of the proposal plays a critical role in the algorithm's performance; a large variance increases the likelihood of proposing candidates far from the current state, which may lead to frequent rejections and slow convergence. A small variance results in limited exploration of the state space, potentially preventing the chain from adequately sampling the target distribution. The optimal choice of variance balances the exploration and acceptance rate, often tuned experimentally or using adaptive techniques. The standard deviation is also known as the step size or scale parameter.

Another element often used in the algorithm is the *burn-in* period. From the starting state to a pre-defined number of iterations $N_{\text{burn}-\text{in}}$ are discarded since these initial samples of the Markov chain may not represent the target distribution, as the chain has not yet approached to its stationary distribution.

The behaviour of the algorithm is illustrated in Figure 5.2, sampling from the target distribution defined in Equation 5.7 using a Gaussian proposal with $\sigma=0.2$. The total chain length is $5,000$, with $4,690$ proposals accepted. The grey portion indicates the burn-in period, during which the chain stabilises to a stationary distribution. This figure demonstrates the sensitivity of the Metropolis-Hastings algorithm to the standard deviation (σ) of the proposal distribution. Two cases are shown: $\sigma=0.2$ and $\sigma=1$. In the top panel ($\sigma=0.2$), the chain exhibits slow exploration of the target distribution and insufficient sampling of all modes, spending most of its time in the highest mode. In contrast, in the bottom panel ($\sigma=1$), the step size allows the chain to efficiently transition between modes, resulting in a more accurate approximation of the target distribution.

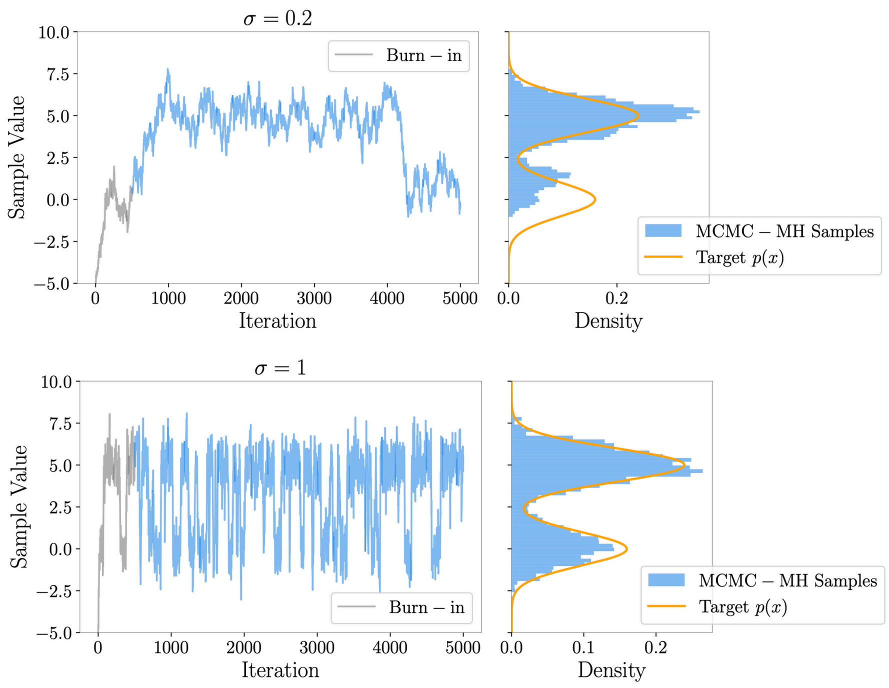


FIGURE 5.2: Metropolis-Hastings sampling from the target distribution in Equation 5.7 using Gaussian proposals with $\sigma = 0.2$ (top) and $\sigma = 1$ (bottom). The total chain length is 5,000 , with 4,690 proposals accepted for $\sigma = 0.2$. The grey portion represents the burn-in period. The figure highlights the sensitivity of the algorithm to σ, showing slow exploration and mode trapping for $\sigma = 0.2$, while $\sigma = 1$ achieves better mode transitions and a more accurate approximation of the target distribution.

Challenges

MCMC-MH performs well in higher dimensions because it generates candidate states using small, local moves, allowing efficient exploration of the local geometry of the *target distribution*. With a properly scaled step parameter, the algorithm can also transition between modes of the target distribution. However, determining the optimal step size is a significant challenge, as it depends on the specific characteristics of the *target distribution*. To address this, various tuning methods have been proposed, such as dynamically adjusting the step size based on the acceptance rate. Alternatively, running multiple chains in parallel can improve exploration but at the cost of increased computational demand [154]. Finally, the algorithm relies on the assumption that the chain is sufficiently long to reach the *stationary distribution*, ensuring an accurate approximation of the *target distribution*. As shown in the top panel of Figure 5.2, if the chain had been stopped around iteration 3000, the smaller mode at $\mu = 0$ would not have been discovered.

These methods can become computationally prohibitive when the underlying physics simulations are particularly complex or expensive. For example, both methods require querying the physics model in every iteration, even for rejected samples, which further increases computational costs. One alternative is to use adaptive MCMC methods, or advanced techniques such as MultiNest [42] or Hamiltonian MCMC [155], to improve efficiency; however, in this work, we focus on other active approaches. Specifically, we adopt a more efficient strategy by approximating the behaviour of the underlying physics model using function approximators.

5.2 Bayesian Optimisation

Bayesian Optimisation (BO) [156, 51] is a machine-learning-based optimisation framework designed to efficiently identify optimal solutions for an objective function. Formally, BO seeks to solve:

$$x^* = \arg\max_{x \in \mathcal{X}} f(x) \tag{5.12}$$

Where $\mathcal{X} \subseteq \mathbb{R}^d$ is the search space, typically a hyper-rectangle, and f is the objective function to be optimised. The objective function often represents a computationally expensive or difficult-to-evaluate real-world process or simulation. BO operates through three primary components:

- The **objective function** f: the function being optimised.
- The **surrogate model** $\hat{f}$: an approximation of f used to reduce computational cost.
- The **acquisition function** $\alpha(x)$: a mathematical construct guiding the selection of the next point to evaluate, defined such that high values correspond to potentially high values of f.

The surrogate model $\hat{f}$ approximates the expensive objective function f, enabling efficient optimisation.The *Bayesian* aspect of this method arises from its use of Bayes' theorem to update the probabilistic surrogate model of the objective function as new data becomes available (see Appendix A). The acquisition function $\alpha(x)$ leverages this surrogate model to identify promising regions for evaluation. Specifically, $\alpha(x; \hat{f})$ balances two competing aspects:

- **Exploitation**: Prioritising regions where the surrogate model predicts high objective function values.
- **Exploration**: Prioritising regions of high uncertainty in the surrogate model to gather more information about the objective function.

This balance is referred to as the **exploitation-exploration trade-off**.

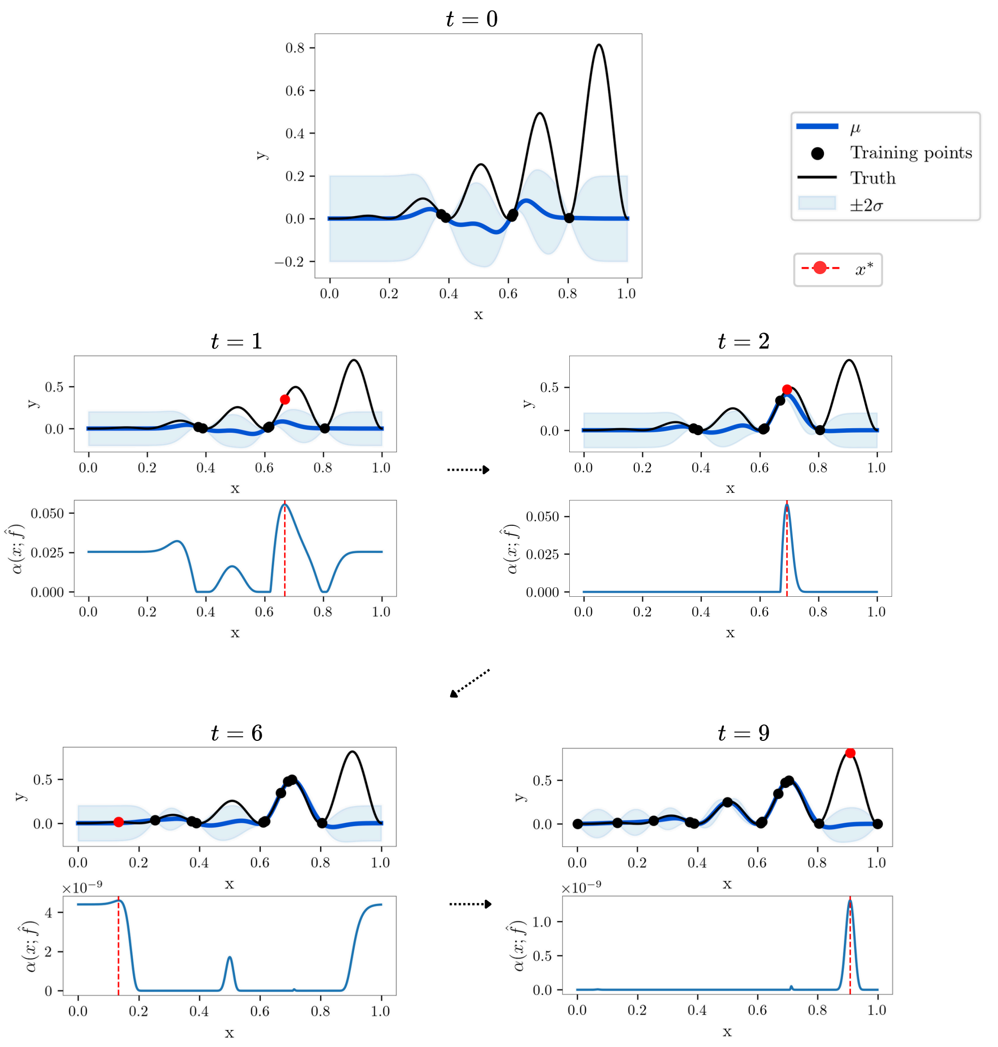


FIGURE 5.3: Bayesian Optimisation (BO) process starting with five random initial training points (black dots). The true objective function is shown as a solid black line, while the Gaussian Process (GP) surrogate model updates in each iteration are represented by the mean μ (solid blue line) and standard deviation σ (shaded region). The bottom panel shows the Expected Improvement (EI) acquisition function for iterations $t = 1, 2, 6$, and 9, with the next evaluation point x^* identified as the maximum of the acquisition function (red dot and vertical red line). The figure illustrates the iterative refinement of the surrogate model and the selection of evaluation points, converging towards the objective function's optimum.

5.2.1 Optimisation Process

BO typically begins with a small, randomly generated initial dataset to initialise the surrogate model. For probabilistic surrogate models, such as GPs described in Section 4.2, the posterior mean $\mu(x)$ and posterior standard deviation $\sigma(x)$ represent the surrogate model's predictions for $f(x)$ at any point $x \in \mathcal{X}$. The acquisition function $\alpha(x; \hat{f})$ combines these predictions to rank potential evaluation points. The point x^*, corresponding to the maximum of the acquisition function, is selected as the next evaluation point for the objective function, $f(x^*)$, in the current iteration.

Common acquisition functions [51] are described below.

Probability of Improvement (PI)

$$\alpha_{\mathrm{PI}}(x) = \Phi\left(\frac{\mu(x) - f_{\text{best}}}{\sigma(x)}\right) \tag{5.13}$$

where Φ is the cumulative distribution function of the standard normal distribution, and f_{best} is the best observed objective function value so far.

PI focuses on maximising the likelihood of improvement over f_{best} . This makes the optimisation process conservative, favouring safe incremental improvements.

Expected Improvement (EI)

$$\alpha_{\mathrm{EI}}(x) = \sigma(x)[z\Phi(z) + \phi(z)], \quad \text{where } z = \frac{\mu(x) - f_{\text{best}}}{\sigma(x)} \tag{5.14}$$

where $\phi(z)$ is the PDF of the standard normal distribution.

EI balances exploration and exploitation by considering both the magnitude and likelihood of improvement. As a result, it provides a more flexible optimisation approach, balancing incremental improvements with exploration of uncertain regions.

Figure 5.3 illustrates the BO process. The initialisation begins with five randomly chosen points, represented as black dots, which serve as the initial training set. The true objective function is depicted as a solid black line. During each iteration, the GP surrogate model is updated, shown as the mean μ (solid blue line) along with its standard deviation σ capturing the uncertainty, and fitted to the observed black points. The EI acquisition function is displayed in the bottom panel for iterations $t = 1, 2, 6$, and 9. The figure highlights how the maximum value of the acquisition function, x^* determines the next evaluation point for the objective function, marked by a red dot on the top plot and a vertical red line in the acquisition panel. This visualisation demonstrates the iterative refinement of the surrogate model and the selection of new evaluation points, converging towards the optimum of the objective function.

5.2.2 Tree-structured Parzen Estimator optimisation

The Tree-Structured Parzen Estimator (TPE) algorithm is a variant of BO methods [157, 158], commonly used for hyperparameter optimisation in machine learning. Instead of relying on uncertainty estimates from models like GPs, TPE employs probability density estimators, often referred to as Parzen Estimators or Kernel

Density Estimators (KDE), as described in Section 4.5. These estimators act as surrogate models to directly approximate $p(\boldsymbol{x} \mid y)$, formulated as follows:

$$p(\boldsymbol{x} \mid y) = \begin{cases} l(\boldsymbol{x}) & \text{if } y < y^* \\ g(\boldsymbol{x}) & \text{if } y \geq y^*, \end{cases} \tag{5.15}$$

where $l(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are the KDE probability densities modelling the two groups of observations. The value y^* is defined to be a quantile γ of the observed y values satisfying $p\,(y < y^*) = \gamma$. In TPE, $p(\boldsymbol{x}, y)$ is parameterised as $p(y)p(\boldsymbol{x} \mid y)$ to facilitate the optimisation of EI acquisition function, although an explicit model for $p(y)$ is not needed since with these considerations the EI acquisition function becomes proportional to [157]:

$$EI_{y^*}(x) \propto \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1} \tag{5.16}$$

Maximising this equation leads to selecting points x that predominantly align under the distribution $l(\boldsymbol{x})$ rather than $g(\boldsymbol{x})$.

5.2.3 Multi-objective BO

The BO description in the previous section focuses on a single objective function. However, many real-world problems involve multiple conflicting objectives that must be optimised simultaneously. In Multi-Objective BO (MOBO) [51], the goal is not to find a single optimal solution but rather a set of solutions that represent trade-offs among the objectives, known as the Pareto front. Formally, MOBO seeks to optimise a vector-valued objective function:

$$\mathbf{f}(x) = [f_1(x), f_2(x), \ldots, f_m(x)]\,, \tag{5.17}$$

where m is the number of objectives. Optimality is given by the Pareto set $\mathcal{P}^*$ is the set of solutions in search space that cannot be improved in one objective without degrading another. The Pareto front $\mathcal{F}^*$ is the image of the Pareto set in the objective space, corresponds to the trade-offs between objectives in the objective space.

In MOBO acquisition functions are adapted to handle multiple objectives and help find Pareto-optimal solutions. They focus on different goals, such as managing high-dimensional search spaces [159], ensuring diversity in both the inputs and outputs [160], and improving the size of the Pareto front's coverage [161].

5.2.4 BO Takeaway*

The introduction of the surrogate model $\hat{f}$ and the acquisition function $\alpha(x; \hat{f})$ transforms the optimisation problem for f, as defined in equation (5.12), into the following form:

$$x^* \approx \arg\max_{x \in \mathcal{X}} \alpha(x; \hat{f}). \tag{5.18}$$

The key difference is that $\hat{f}$ is inexpensive to evaluate at any point in the search space. This allows equation (5.18) to be solved using standard optimisation algorithms. Classical optimisation routines, such as L-BFGS-B [162], are often employed in BO computational libraries.

In MOBO, the Pareto set represents solutions where no objective can be improved without compromising another. While finding this set is valuable, it may not always capture the full diversity of potential solutions, potentially overlooking certain regions of interest. Ensuring a well-distributed set of solutions is crucial for the effectiveness of Parameter Scan (PS) algorithms.

The elements and language of BO are integral to implementing PS algorithms. The acquisition function can be used not only to search for optima but also for other purposes, such as searching for diverse solutions (see Chapter 7). Probabilistic models, such as GPs, play a key role in guiding exploration of unknown regions in the parameter space and evaluating the risk associated with sample evaluations based on their uncertainty estimates. In this thesis, we adopt this framework to develop and implement PS methods.

5.3 Reinforcement Learning

5.3.1 The Reinforcement Learning framework

Reinforcement Learning (RL) [163] is a generic framework to describe and solve any decision-making problem where a sequential strategy is needed to achieve a goal. The optimal strategy is found by learning from the interaction of two entities. The *agent*, the part of the system that learns from taking actions, and the *environment*, which reacts to the agent's actions and is defined as everything that lies outside the agent such as the configurations and dynamics of the problem. The environment generates a numerical signal called the reward, which characterises the task's final goal and guides the agent's learning process.

Focusing on discrete sequential decision problems, each interaction is performed in a time step t, an abstract unit of time defined for the task. In a given interaction between the agent and the environment, the agent observes a representation of the environment $s_t \in S$, called the state where S is the state space, the set of all possible states. Based on this state, the agent takes an action $a_t \in A(s_t)$ where $A(s_t)$ is the action space, a set of all the available actions at a given state s_t. The environment is affected by this action and generates a new state based on the one-step dynamics of

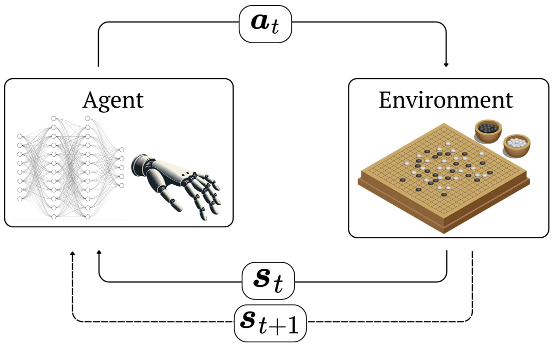


FIGURE 5.4: Illustration of the RL framework. The agent interacts with the environment by performing an action a_t, which leads to a state transition from s_t to s_{t+1}. The environment provides feedback to the agent based on its actions, enabling the agent to learn and adapt its policy over time.

the problem defined by the state-transition probability functions $p\left(s_{t+1} \mid s_t, a_t\right)$ and provides a scalar reward signal r_t through a reward function $R : S \times A \rightarrow \mathbb{R}$. The information generated in each time step $\left(s_t, a_t, r_{t+1}, s_{t+1}\right)$ is known as a transition or an experience tuple. A completed cycle from an initial state s_0 to a terminal state s_T is called an episode and the chain of sequential experience tuples is defined as the episode's trajectory. With these elements we can completely specify the environment as a Markov Decision Process (MDP) [164] minimally defined by the tuple

$$\mathcal{MDP}\left(S, A, P, R, \gamma\right), \tag{5.19}$$

where $0 \leq \gamma \leq 1$, is the discount factor and determines the value of future rewards at a times step t though the discounted return,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{T} \gamma^k R_{t+k+1}. \tag{5.20}$$

The *Markov* property, introduced in 5.1.2 on equation (5.9), of a decision process is due to the fact that the transition probabilities and the reward function depends only on the present information of the environment; specifically, given equation (5.9), the reward at the state s_t is:

$$R\left(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1} \cdots, s_1, a_1\right) = R\left(s_{t+1} \mid s_t, a_t\right). \tag{5.21}$$

The agent chooses an action a_t by observing a state s_t according to a distribution function $a_t \sim \pi(a_t|s_t)$ called the policy. The policy can be also a deterministic function, in that case is commonly denoted by $a_t = \mu(s_t)$. Thus, an complete episode's trajectory is given by,

$$\mathcal{T}_\pi := \prod_{t=0}^{T} p\left(s_{t+1} \mid s_t, a_t\right) \pi\left(a_t \mid s_t\right). \tag{5.22}$$

Then, the expected discounted return can be defined as,

$$J(\pi(s_t)) := \mathbb{E}_{\mathcal{T}_\pi} G_t \tag{5.23}$$

The goal of a RL algorithm is to find an optimal policy $\pi^\star$ for every state, which maximises the expected discounted return,

$$\pi^\star = \arg\max_\pi J(\pi) \tag{5.24}$$

5.3.2 Value functions

We can define additional information about actions and states which serves as a measure of the effectiveness of the policy. First the value of a state s under a policy π is given by,

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s\right] \tag{5.25}$$

which represents the expected return of a trajectory sampled from a policy π, starting from the state s, and can be written recursively in terms of the trajectory as,

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right], \forall s \in S \tag{5.26}$$

Second, the value of taking an action a in a state s, will help to compare different possible actions in the same state giving us information on what action is the best for a given state. This information is given by the action-value function, also known as the Q-function,

$$q_\pi(s,a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right] \tag{5.27}$$

which can be written as,

$$q_\pi(s,a) = \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right], \forall s \in S, \forall a \in A(s) \tag{5.28}$$

Therefore, finding the optimal value function,

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S \tag{5.29}$$

or the optimal Q-function function,

$$q_*(s,a) = \max_\pi q_\pi(s,a), \forall s \in S, \forall a \in A(s) \tag{5.30}$$

lead us to the optimal policy as well, solving in this way, the RL problem.

5.3.2.1 Deep Reinforcement Learning

Classical methods for finding optimal policies in Reinforcement Learning (RL) rely on tabular approaches, where value functions, such as V(s) and Q(s, a) , are represented explicitly as tables. These methods include algorithms like Dynamic Programming (e.g., Policy Iteration and Value Iteration) and Temporal Difference methods (e.g., Q-Learning and SARSA). These techniques iteratively compute the optimal values for all states or state-action pairs and derive policies based on these values. While effective for small-scale problems with discrete and manageable state and action spaces, these methods become computationally infeasible in scenarios with large or continuous state-action spaces due to the exponential growth of the table size, and suffering from the *curse of dimensionality* (see Section 3.2.4).

Deep Reinforcement Learning (Deep RL) uses deep ANNs to model the value functions $V(s)$, $Q(s,a)$, or policies $\pi(a|s)$, integrating the representational aspect of ANNs with the decision-making framework of RL to approximate optimal policies. This approach is advantageous in scenarios with large or continuous state and action spaces, where tabular methods [163] are computationally infeasible.

A notable breakthrough of Deep RL is *AlphaGo*, developed by DeepMind [165], which in 2016 became the first AI system to defeat a world champion Go player. AlphaGo employed a combination of deep ANNs, supervised learning from expert human Go plays, and RL to master the game's complex strategies, a task previously thought to be beyond the reach of AI due to the immense state and action spaces involved. This success not only demonstrated the practical capabilities of Deep RL but also underscored its ability to tackle highly strategic, long-horizon decision-making problems.

5.3.3 RL Takeaway*

RL has proven to be an effective approach for problems requiring sequential decision-making strategies to identify optimal policies. It has demonstrated historical milestones in AI development, such as mastering the game of Go. However, these advancements also highlight the immense potential of Deep RL in the physical sciences. The key distinction is that, unlike board games, the rules of *nature's game* are unknown, adding an additional layer of complexity and opportunity for exploration.

Deep RL has been applied to improve MCMC methods [166] and has been compared to variational inference approaches [167]. It has also been utilised for optimising hyper-parameters in ANNs [168]. Therefore, with an appropriate translation of the PSP into the RL framework, RL algorithms could be valuable for exploring the parameter spaces of BSM models. This experiment is detailed in Chapter 6.

Part II

Research and Results

Chapter 6

Space Exploration with Reinforcement Learning

In this chapter, we explore the application of RL to solving the PSP. While the results were unsatisfactory, primarily due to the sample-intensive nature of RL and the challenges in designing an effective reward function, this exploration provides valuable insights. Presenting these negative results is crucial for illustrating the limitations and challenges faced, as well as the lessons learned. This initial investigation also played a pivotal role in shaping the direction of this thesis, motivating the development of the main contributions presented in next chapters.

6.1 The Environment

As introduced in Section 5.3 and Equation (5.19), the RL environment is practically defined by formulating the state space, the action space, and the reward. In this chapter, we follow the same formulation as in previous chapters, where we consider the HEP-Stack or general objective function denoted by $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ represents the search space and $\mathcal{Y}$ the objective space.

To transform the parameter scan problem into a Markov decision process, a state is defined as a configuration $\mathbf{x} \in \mathcal{X}$. At each time step t, the state corresponds to the current tested parameters, denoted as $s_t = \mathbf{x}_t$. An action is defined as the difference vector for the current configuration, $a_t = \delta\mathbf{x}$.

The reward can then be defined in terms of a Gaussian likelihood over the objectives:

$$\mathcal{L}(\mathbf{y}, \mathbf{y}^*) = \prod_i \exp\left\{-\frac{(y_i - y_i^*)^2}{2\sigma^2}\right\} \tag{6.1}$$

where σ is the standard deviation. The primary difference from previous formulations of the PSP is that, instead of constraints on the objectives, a single set of optimal values is required. These optimal values are denoted as $y_i^* \in \mathbf{y}^*$ for $i = 1, \ldots, m$, where m is the dimensionality of the objective space. A simple configuration for the reward function can then be expressed as:

$$\mathcal{R}(s_t) = \kappa(\mathcal{L}_{t+1}(\mathbf{y_{t+1}}, \mathbf{y}^*) - \mathcal{L}_t(\mathbf{y_t}, \mathbf{y}^*)) \tag{6.2}$$

where $\kappa > 0$ is a proportionality constant. This configuration encourages the agent to prioritise improvements in the likelihood, thereby optimising it. An alternative configuration could use the likelihood directly as the reward. However, this approach results in a static environment reward, driving the agent to optimise the likelihood without accounting for diversity in the solutions.

The reward becomes dynamic, depending on the states visited within the episode at step t. This is achieved by calculating the density of the visited states in the search space using a kernel density estimation (KDE), as introduced in Section 4.5. Consequently, an alternative formulation for the reward function can be expressed as:

$$\mathcal{R}(s_t) = \mathcal{L}(\mathbf{y}_t, \mathbf{y}^*) \exp(-\rho_t(\mathbf{X}_t)) \tag{6.3}$$

where $\rho_t(\mathbf{X}_t)$ represents the KDE for $\mathbf{X}_t$, the set of states visited up to step t. Here, the likelihood provides a static reward that is modulated by the density of the visited states, encouraging exploration of a diverse set of new states during the episode. Therefore, starting from a random parameter configuration state s_t, an optimal policy *trained* in this environment would be capable of sampling N points per episode, covering the entire $\mathcal{S}$ region, without directly evaluating the objective function $\mathcal{H}$ during the testing phase.

6.1.1 Deep Deterministic Policy Gradient

The environment definition in the previous section determines the selection of the RL algorithm. Since the problem operates in a continuous action space, an RL algorithm capable of handling such spaces is required.

Deep Deterministic Policy Gradient (DDPG) [169, 170] is an off-policy, model-free RL algorithm designed for continuous action spaces. It extends the Deterministic Policy Gradient (DPG) algorithm by incorporating concepts from Deep Q-Networks (DQN), such as experience replay and target networks, to improve stability and learning efficiency.

DDPG employs two neural networks:

- The Actor network, $\mu_\theta\left(s_t\right)$, which represents a deterministic policy mapping states to actions.
- The Critic network, $Q_\phi\left(s_t, a_t\right)$, which estimates the Q-value of a state-action pair and evaluates how good an action is in a given state.

The parameters of these networks, θ for the Actor and ϕ for the Critic, are optimised through training. Additionally, target networks (slow-moving copies of the Actor and Critic) are maintained and updated gradually to improve training stability by reducing oscillations in learning.

6.1.2 Deep Deterministic Policy Gradient

DDPG follows an actor-critic framework where the Critic guides the Actor's learning process. The Critic network is trained using the Bellman equation to minimise the difference between its predicted Q-values and the target Q-values derived from the reward signal:

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)Q_{\phi_{\text{targ}}}\left(s', \mu_{\theta_{\text{targ}}}\left(s'\right)\right)\right)\right)^2\right], \tag{6.4}$$

where $\mathcal{D}$ is the dataset from Experience Replay Memory (ERM), which stores past experiences (s, a, r, s', d) and allows the model to learn from uncorrelated samples rather than sequential ones. This mitigates issues related to non-i.i.d. data in RL.

The Actor network is trained to select actions that maximise the Q-values given by the Critic:

$$\max_\theta \mathrm{E}_{s\sim\mathcal{D}}\left[Q_\phi\left(s, \mu_\theta(s)\right)\right] \tag{6.5}$$

By following the gradient of the expected Q-value, the Actor learns to improve its policy over time.

Since the policy is deterministic, DDPG introduces noise to encourage exploration:

$$\mu'\left(s_t\right) = \mu\left(s_t \mid \theta_t^\mu\right) + \mathcal{N}, \tag{6.6}$$

where $\mathcal{N}$ is an exploration noise term, often modelled using Gaussian noise or Ornstein-Uhlenbeck noise for temporally correlated exploration. A flow diagram for the DDPG algorithm is shown in Figure 6.1.

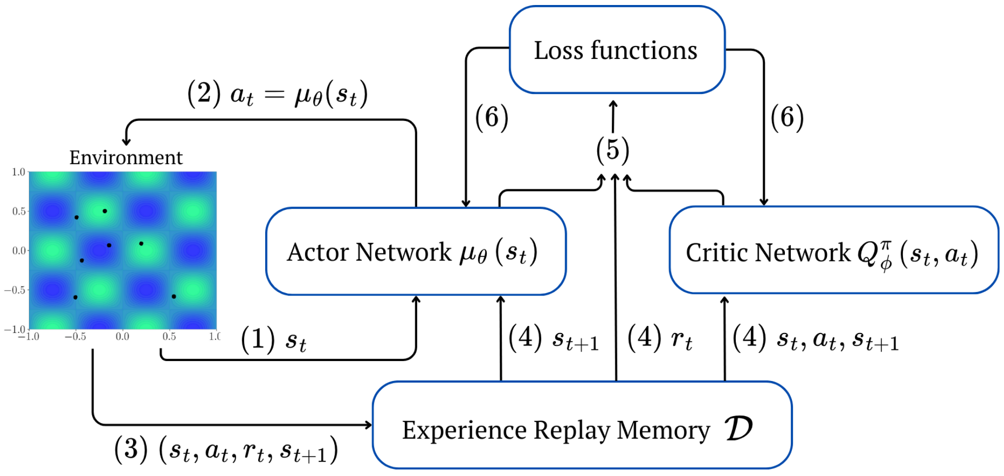

Loss functions
(2) $a_t = \mu_\theta(s_t)$
(6)
(5)
(6)
Environment
Actor Network $\mu_\theta(s_t)$
Critic Network $Q^\pi_\phi(s_t, a_t)$
(1) s_t
(4) s_{t+1}
(4) r_t
(4) s_t, a_t, s_{t+1}
Experience Replay Memory $\mathcal{D}$
(3) (s_t, a_t, r_t, s_{t+1})

FIGURE 6.1: Flow diagram of the Deep Deterministic Policy Gradient (DDPG) algorithm: (1) The environment provides the current state s_t. (2) The Actor network receives the state and generates an action a_t. (3) The environment returns the reward r_t and the next state s_{t+1}, storing the experience tuple (s_t, a_t, r_t, s_{t+1}) in the Experience Replay Memory (ERM) $\mathcal{D}$. (4) A batch of experiences is sampled from the ERM and processed by the Actor and Critic networks. (5) The loss functions are computed.

6.2 Experiments and Results

6.2.1 Toy models

As an initial test experiment, the environment setup was implemented using the Egg-box test function, defined as:

$$\mathcal{H}(x, y) = \left(2 + \cos\frac{x}{2}\cos\frac{y}{2}\right)^5 \tag{6.7}$$

The focus was on the range $(10, 15)$, with a ring-shaped likelihood. Figure 6.2 illustrate this setup for an example objective observable $\mathcal{H}^* = 100$. The distributed

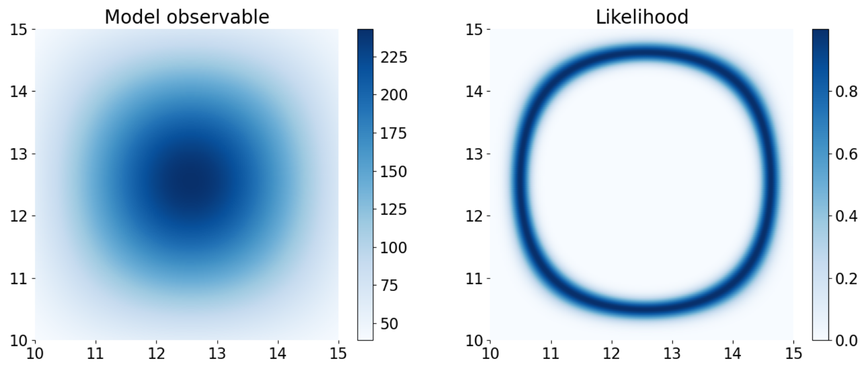


FIGURE 6.2: Span over a squared region $(10, 15)$ of the Egg-Box model (left) and the likelihood (right) for a objective observable $y^* = 100$

DDPG algorithm was executed using a reward configuration based on the likelihood,

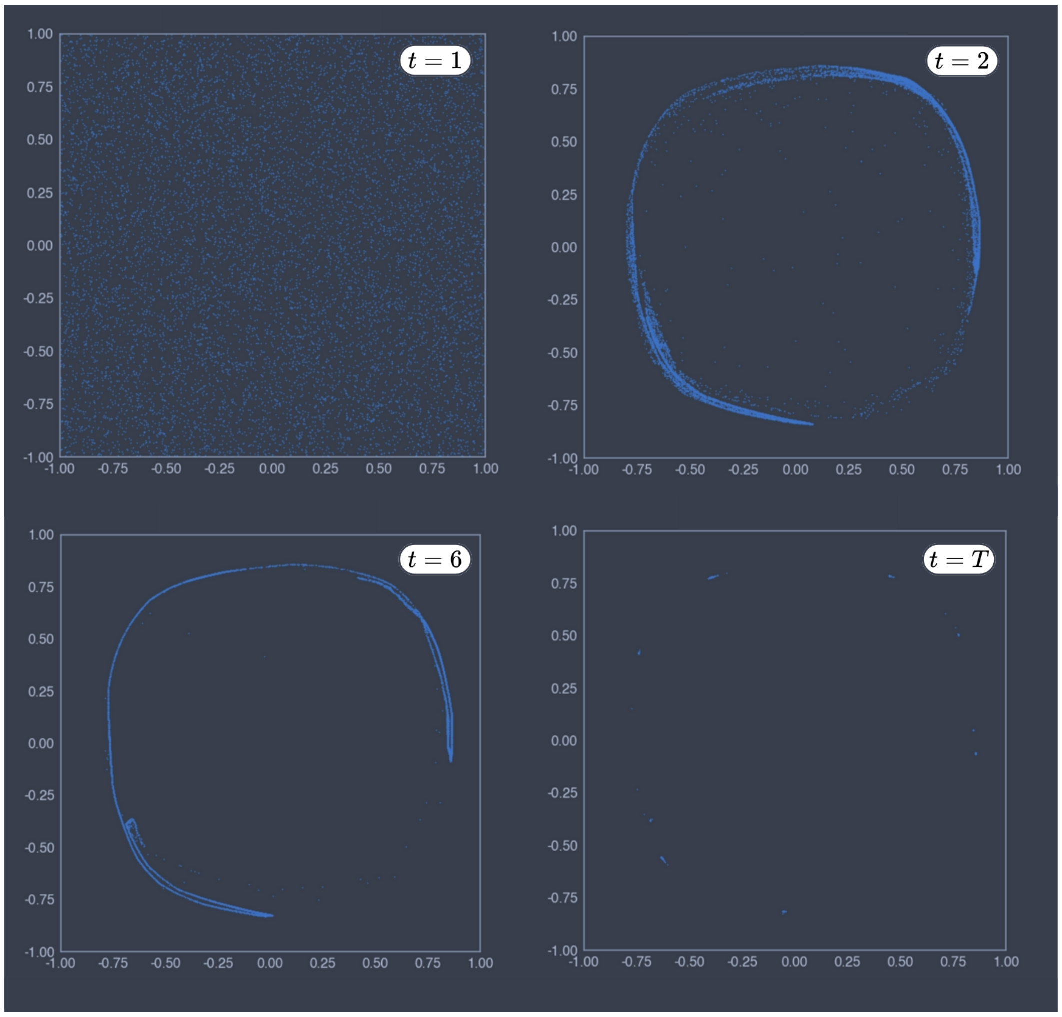


FIGURE 6.3: Evolution of states during a test episode after training.

modulated by the exponential density calculated via the KDE (defined in Equation (6.3)). The terminal state was set at $T = 100$. Within this environment setup, the objective function is not evaluated during the test phase. Consequently, the policy can operate on an arbitrary large number of randomised states at testing time, transforming the states at each time step of the episode. This behaviour is illustrated in Figure 6.3.

Results from the test episode showed that, although the reward function was designed to encourage sample diversity, the ring-shaped likelihood was captured at the second time frame. Subsequently, the agent transformed the states into seemingly optimal random points. However, based on the true objective function, it is evident that these points are not genuinely optimal due to the degeneracy of the ring-shaped likelihood. Repeated experiments demonstrated that the agent consistently converged to random points identified early in the training phase.

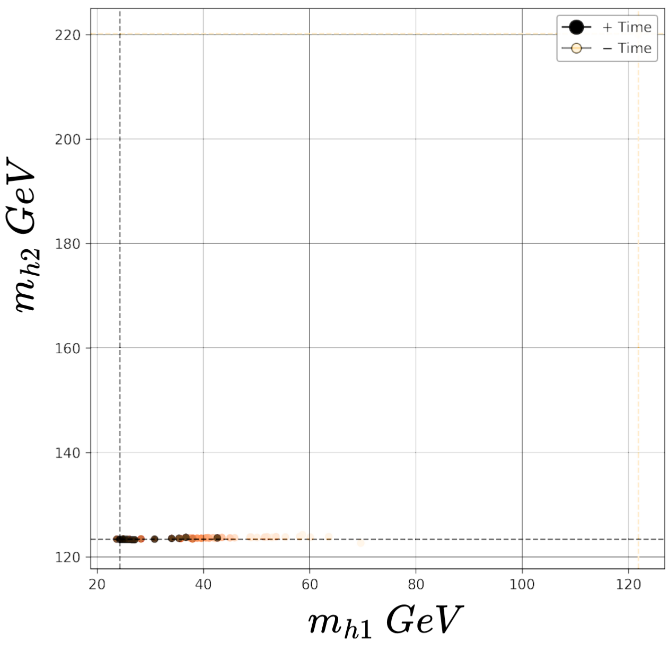


FIGURE 6.4: Trajectory of the agent in the output mass space during a single search episode. The colour gradient from light to dark orange indicates progression over time steps. The agent begins near $(m_{h'} = 125, , m_{h^{\rm SM}} = 220)$ and converges towards the target region $m_{h^{\rm SM}} = 125$, though it fails to reach the optimal value for $m_{h'}$ in this run.

6.2.2 BSM case

Focusing on a simplified phenomenological case study using the $(B - L)$SSM model, we aim to fit the two lighter Higgs states in the model to both the discovered Higgs boson at 125 GeV and the newly observed anomaly at 95 GeV. The search and output spaces are defined respectively by $\mathcal{X}$ and $\mathcal{Y}$:

$$\mathcal{X} : (M_0, M_{1/2}, \tan\beta, A_0)\,, \quad \mathcal{Y} : (m_{h'}, m_{h^{\rm SM}})$$

where the optimal output values used to compute the reward function (6.3) are defined as $m_{h'} = 95$ GeV and $m_{h^{\rm SM}} = 125$ GeV. The ranges for the parameters in the search space are defined in Table 7.2.

After training, we display the performance of the policy during a single search episode. The output space, corresponding to the Higgs masses, is shown in Figure 6.4, where lighter orange points represent earlier time steps and darker shades correspond to later ones.

We observe that the agent begins sampling from a random initial point, roughly at $(m_{h'} = 125,\ m_{h^{\rm SM}} = 220)$ and quickly moves to the vicinity of $m_{h^{\rm SM}} = 125$. However, in this particular test run, it approaches $m_{h'} \approx 20$ which deviates from the experimentally optimal value.

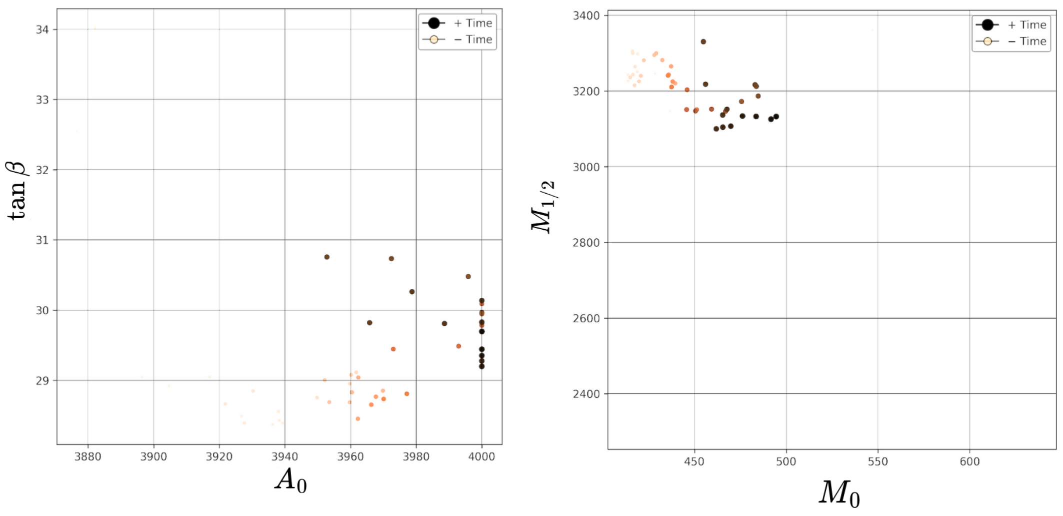


FIGURE 6.5: Search space exploration during a single episode, shown in $(A_0, \tan\beta)$ and $(M_0, M_{1/2})$ subspaces. The agent fixes A_0 and explores the other parameters over time, as indicated by the light-to-dark orange progression.

The search space for the same episode is shown in Figure 6.5. Here, we observe another unexpected behaviour: the policy learns to fix one parameter–in this case, A_0–while exploring the remaining dimensions. Although this strategy is viable in principle, as good parameter regions can indeed be found for a fixed A_0, it is not the desired behaviour of a search algorithm. Ideally, we aim to explore the full satisfactory region that yields the target outputs, rather than converge prematurely on a subspace.

6.2.3 Final remarks

Guiding the trained agent towards discovering a comprehensive description of the $\mathcal{S}$ region proved challenging, as it required careful construction of the reward function and advanced exploration techniques within reinforcement learning algorithms. Exploring alternative formulations of the environment presents a promising direction for future research.

The algorithm did not consistently converge to the same regions in the parameter space; this may be due to the algorithm's limited exploration capabilities. To enhance exploration and accelerate convergence, we experimented with a distributed version of DDPG, in which multiple copies of the policy network interact with the environment and collect experiences in parallel. While the distributed version improved training speed, it did not enhance exploration as anticipated. This outcome can be attributed to the fact that DDPG is a model-free algorithm [169]. In the terminology employed in this thesis, it does not utilise a surrogate model, relying instead directly on objective function evaluations for training.

Finally, an advantage of RL is its natural operation on discrete action spaces, which broadens its application to not only formulating the PSP as an RL problem but also exploring discrete model spaces. This approach is currently utilised in BSM phenomenology for model space exploration instead. For instance, it has been applied to investigate the Froggatt-Nielsen family of models for quark masses [171], as well as in Graph Reinforcement Learning [172], where BSM models are represented within a graph structure. In the latter, RL algorithms are employed to explore configurations such as vector-like leptons and dark $U(1)$ gauge symmetry models. Moreover, RL has been used in String Theory [173] to navigate the landscape of string vacua. However, in these applications, the PSP presents a computational challenge, as a parameter space scan is always required to evaluate the viability of a given model.

Chapter 7

Bayesian Active Search on Parameter Space

7.1 Introduction

The SM was introduced in Chapter 2, with its completion with the discovery of the Higgs particle with a mass of 125 GeV at the LHC in July 2012 [8, 3]. Section 2.3 reviewed the challenges of the SM including Neutrino masses, dark matter and hierarchy problems and many other theoretic fundamental problems, such as the explanation of Gravity at the quantum scales. Also, Section 2.3 introduced the concept of New signals, leaving clear that the construction for a theory Beyond the SM is key for the current status of particle physics. From the model landscape review in Section 2.4 for models explaining different combinations of these phenomena concluding we focus now the $(B-L)$ SSM model, as a distinctive example of a model realisation of Supersymmetry that can explain the anomalous experimental signals at ≈ 95 GeV reported in searches for new Higgs bosons as a study case for parameter scan algorithm development.

The research presented in this in this chapter proposes a newly developed PS methodology, improving sample efficiency for performing PS, where numerical evaluations of BSM models (with the aforementioned $(B-L)$SSM being out benchmark example) are computational expensive. We formulate the PSP as an Active Search (AS) problem, a framework using the elements of BO but with the aim of search instead of optimise. In this approach, multiple phenomenological signatures of a particular BSM model are set as the multiple objectives, constrained by experimental measurements. These can refer to particle masses, Branching Ratios (BRs), production cross-sections or any model prediction information. Thus, we introduce a batched Bayesian Multi-Objective (BMO) AS algorithm, which we name b-CASTOR, which stands for **b**atched **C**onstraint **A**ctive **S**earch with **T**PE **O**ptimisation and **R**ank based sampling. This algorithm provides a comprehensive characterisation of the entire satisfactory region in the parameter space of the BSM model under study. Additionally, it ensures sample diversity within the discovered region.

b-CASTOR uses GP surrogate models to approximate the objectives, as in BO and the Expected Coverage Improvement acquisition function, introduced in [174]. This policy aims to increase the volume coverage within the satisfactory region of the search space to propose new candidate solutions in a sequential manner, obtaining the sample diversity. Lastly, a sampling strategy that allows for multi-point evaluations is developed, leading to a dense collection of samples from the satisfactory region. This strategy also provides tunable control over the exploration and exploitation trade-off of the search strategy.

The methodologies and findings discussed in this chapter have been published in [175], and the accompanying code for result replication is accessible at [176].

7.2 Literature review

As mentioned in the introduction, Bayesian inference techniques are employed for parameter scans, framing the PSP as a sampling problem [39]. The goal is to sample from the posterior probability, which is proportional to the product of the likelihood and the prior, indicating how likely the BSM model is to explain certain objective observables. Monte Carlo methods [41] are used to sample from this target probability density, following the methodology outlined in Chapter 5. Although these methods scale well to higher dimensions, they typically require a large number of samples to reach the stationary target distribution. Moreover, due to the Markovian nature of these methods, the parameter space is explored step by step, with each move depending only on the current state rather than the full sampling history. MCMC methods such as Metropolis-Hastings also struggle when the target density is multi-modal, requiring more advanced exploration strategies.

MultiNest [177] is a Bayesian inference algorithm based on nested sampling [42], widely used in the physical sciences. It addresses the challenges of multi-modality and degeneracy in complex target distributions and also provides the Bayesian evidence, which is essential for model comparison. However, it still suffers from sample inefficiency, requiring many likelihood evaluations, and can struggle with narrow or isolated modes.

The sample efficiency can be improved by integrating a surrogate model into the sampling algorithm. This surrogate can be used to approximate either the likelihood function or the observables directly, making the evaluation of parameter configurations computationally cheaper. The behaviour of the sampling process is governed by a specific algorithm, which may use the surrogate to guide the selection of new parameter points. This guiding mechanism can be referred to as the decision strategy or policy. For instance, in Metropolis-Hastings MCMC, the decision strategy corresponds to the acceptance ratio criterion, while in BO, it is represented by the acquisition function. The choice and design of this policy determines whether the PS algorithm behaves more like a sampling method, an optimisation routine, or a search strategy.

The integration of these elements into PS algorithms is rapidly growing in the literature. One of the earliest examples is [5], where a ANN is used as a surrogate model to approximate multiple observables directly, after which a predefined likelihood is evaluated. This surrogate-based likelihood is then sampled using Rejection Sampling, which selects a desired number of configurations to be evaluated using the actual objective function — the computationally expensive physics simulation. The ANN is retrained continually with new evaluations. This approach essentially decouples the expensive likelihood evaluations from the sampling process and through continual retraining of the ANN enables the surrogate to adapt and improve over time. However, the algorithm remains fundamentally a sampling-based approach, rather than a directed search. Its sampling behaviour does not adaptively target underexplored or high-uncertainty regions. Although some diversity is introduced via uniform random sampling, this is generally inefficient and lacks a principled strategy to balance exploration and exploitation.

Following this, [45] introduced both regression and classification approaches, utilising a MLP as the surrogate model for each task. The decision strategy involves evaluating a fixed number of random samples using the surrogate, after which the selection process differs depending on the task. In the regression approach, points are selected based on a χ^2 test to target regions of interest. In the classification approach, the probability of belonging to the region of interest is used to identify the decision boundary. These approaches improves versatility by letting researchers tailor the task (best fit vs boundary), but remains fundamentally passive, relying on ML to guide sampling rather than actively searching for new, diverse, or informative points.

Another emerging ANN-based approach is presented in [46], which applies active learning to iteratively learn the decision boundary between ”good” and ”bad” regions of the parameter space using a neural network classifier as a surrogate. Unlike the previous decision strategies discussed, this method dynamically prioritises points with high classification uncertainty—typically located near the boundary—enabling the algorithm to focus computational resources where information gain is greatest. To avoid clustering and promote exploration, a diversity measure based on electrostatic repulsion between candidate points is incorporated into the selection process, controlled by a weighting parameter that balances uncertainty and diversity. This iterative process supports the efficient discovery and refinement of the region of interest. In addition, AL offers benefits such as robustness to sharply varying likelihoods and improved exploration of multiple disconnected regions.

Despite their differences in strategy, all of these approaches share a reliance on sample-intensive neural network surrogate models. These models require repeated training on expanding datasets and depend heavily on large numbers of evaluations to accurately approximate complex parameter space structures, which can limit overall sample efficiency.

Without considering sample efficiency, if the evaluation of the physics model is computationally inexpensive, one can implement a PS method that relies solely on a sophisticated decision strategy, focusing on the search for diversity and characterisation of the region of interest without the need for a surrogate model. This direction is explored by [48, 178], where the parameter space problem is framed as a black-box optimisation task. These studies employ evolutionary and genetic algorithms to navigate the parameter space, leveraging the inherent diversity and adaptability of these methods to efficiently explore complex, high-dimensional landscapes. By treating the model evaluations as black-box functions, these approaches can effectively identify viable regions that satisfy the desired physical constraints, even in scenarios where traditional sampling methods may struggle.

Therefore, at present, no existing method combines sample-efficient surrogate models with a decision strategy (or policy) that actively guides the search toward discovering a diverse set of parameter configurations by leveraging the entire history of collected data points. Such an approach would avoid oversampling in already well-explored regions while simultaneously offering best-fit proposals. The methodology proposed in this chapter introduces a PS algorithm that fulfils these characteristics. Finally, numerous other proposals are emerging, as this remains an active area of research; see [43, 44] for an extensive review and growing list of related methodologies.

7.3 Active Search Formulation

AS is a search methodology that utilises existing knowledge – a series of evaluations – of an objective function to identify points to sample that belong to a rare category. The rare category in this work refers to the subset of all available parameter values $\mathbf{x}$ of a BSM model whose corresponding observables $\mathbf{y}$ returned by a HEP-Stack, $\mathcal{H}(\mathbf{x})$, that satisfy a set of constraints denoted by $\boldsymbol{\tau}$.

An AS method is characterised by three main components, the *objective function*, the *search policy* and the *surrogate model*. The *objective function* represents the HEP-stack and is treated as a black-box function with n input dimension and m output dimension, where m is the number of objectives. The objective function is denoted by

$\mathbf{y} = \mathcal{H}(\mathbf{x})$ where $\mathcal{H} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ is the input vector, and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^\top$ is the output vector. Constraints on the objectives are denoted by the vector $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_m]$. Therefore, the goal of the *search policy* is to suggest candidate points that belong to the satisfactory set $\mathcal{S}$, the set of configurations that satisfy the set of constraints, $\boldsymbol{\tau}$,

$$\mathcal{S} = \{\mathbf{x} \mid \mathbf{y} = \mathcal{H}(\mathbf{x}) \wedge y_i \succeq \tau_i, i = 1, \ldots, m\}. \tag{7.1}$$

Depending on the parameter scan method, the *search policy* can suggest a batch of points $\boldsymbol{X}^* \sim \pi(\cdot)$ or a single point $\boldsymbol{x}^* \sim \pi(\cdot)$. Within the AS framework, the policy uses a *surrogate model* to propose this batch. The *search policy* direct exploration by proposing data points from uncertain, i.e. underrepresented, areas of the data in the search space, or to direct exploitation by choosing data points expected to yield the most useful information according to the predictions of the surrogate model, balancing the *exploration-exploitation trade-off.*

Surrogate models are introduced in Chapter 4. As a brief review, surrogates are defined as a function $f : \mathbb{R}^n \to \mathbb{R}^m$, where n is the dimension of the search space and m the number of objectives. This function aims to approximate $\mathbf{y} \approx f(\mathbf{x})$ based on the dataset $\mathcal{D}_t$ available at the iteration t of the search. The surrogate f is typically computationally less expensive than the *objective function* $\mathcal{H}$ and is well-defined across the entire parameter space $\mathcal{X}$, therefore allows the policy to explore the entire space at low computational cost to determine the batch $\boldsymbol{X}^*$. In this chapter Gaussian Processes (GPs) are assumed to serve as the probabilistic surrogate models (see Section 4.2).

Therefore, a PS algorithm within the AS framework, is an iterative process, where in each iteration t the dataset $\mathcal{D}_t := (\mathbf{X}_t, \mathbf{Y}_t) := (\{\mathbf{x}_j\}_{j=1}^t, \{\mathbf{y}_j\}_{j=1}^t)$ is constructed to fit the *surrogate model* f. The *search policy* then uses this surrogate model to propose a batch of candidate points, $\boldsymbol{X}^* \sim \pi(\cdot \mid f)$, that are likely to fall within the satisfactory region $\mathcal{S}$. Each sample point in $\mathbf{X}^*$ is then evaluated in the *objective function* $\mathbf{Y}^* = \mathcal{H}(\mathbf{X}^*)$ and the dataset $\mathcal{D}_{t+1}$ is updated according to $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (\mathbf{X}^*, \mathbf{Y}^*)$.

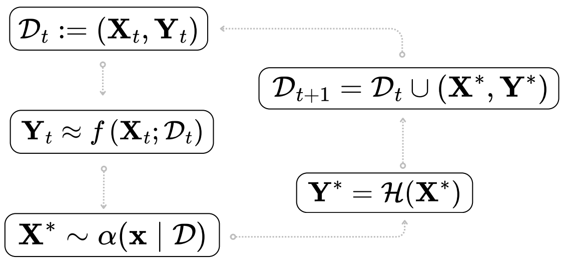

$$\begin{tikzcd}
\boxed{\mathcal{D}_t := (\mathbf{X}_t, \mathbf{Y}_t)} \arrow[d] & \\
 & \boxed{\mathcal{D}_{t+1} = \mathcal{D}_t \cup (\mathbf{X}^*, \mathbf{Y}^*)} \arrow[ul] \\
\boxed{\mathbf{Y}_t \approx f(\mathbf{X}_t; \mathcal{D}_t)} \arrow[d] & \\
 & \boxed{\mathbf{Y}^* = \mathcal{H}(\mathbf{X}^*)} \arrow[uu] \\
\boxed{\mathbf{X}^* \sim \alpha(\mathbf{x} \mid \mathcal{D})} \arrow[ur] &
\end{tikzcd}$$

FIGURE 7.1: A Parameter Scan (PS) algorithm formulated within the Active Search (AS) framework. AS is an iterative process where, in each iteration t , a surrogate model is fitted to the current dataset $\mathcal{D}_t$. A search policy $\alpha(x|\mathcal{D})$ then leverages the surrogate to suggest a batch of points $\mathbf{X}^*$ for evaluating the objective function $\mathcal{H}$. The results are added to the dataset, and the process repeats.

7.4 Expected Coverage Improvement

The Expected Coverage Improvement (ECI) search policy was developed for the Constraint Active Search method [174]. ECI provides a diversity measure in the search space $\mathcal{X}$ by defining a hyper-sphere of radius r around a parameter space point $\mathbf{x}$, called the neighbourhood, given by

$$\mathbb{N}_r(\mathbf{x}) = \{\mathbf{x}' : d(\mathbf{x}, \mathbf{x}') < r\} \tag{7.2}$$

where d is the Euclidean distance. The total neighbourhood for a set of input points $\mathbf{X}$ given a dataset $\mathcal{D}$ is defined as the coverage neighbourhood,

$$\mathbb{N}_r(\mathbf{X}) = \bigcup_{\mathbf{x} \in \mathbf{X}} \mathbb{N}_r(\mathbf{x}),$$

Thus, the volume utility function can be defined,

$$u_{\mathcal{S}}(\mathcal{D}) = \mathrm{Vol}\left(\mathbb{N}_r(\mathcal{D}) \cap \mathcal{S}\right) \tag{7.3}$$

where $u_{\mathcal{S}}$ measures the total volume of S covered by the neighborhood $\mathbb{N}_r(\mathcal{D})$. We also define the total volume covered as u_{T}. CAS aims to discover the dataset $\mathcal{D}$ that covers as much volume of the satisfactory region $\mathcal{S}$ as possible through the maximisation of the ECI policy function,

$$\alpha(\mathbf{x} \mid \mathcal{D}) = \mathbb{E}_{\mathbf{y}}\left[u_{\mathcal{S}}\left(\mathcal{D}_t \cup (\mathbf{x}, \mathbf{y})\right) - u_{\mathcal{S}}\left(\mathcal{D}_t\right)\right] \tag{7.4}$$

Therefore, at time step t of the search, the policy proposes a configuration $\mathbf{x}^*$ through

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x} \mid \mathcal{D}_t) \tag{7.5}$$

However, the equation (7.7) assumes the direct evaluation of the multi-objective function. Instead, probabilistic surrogate models can be used as follows. Given a point $\mathbf{x}$, an indicator variable $Z(\mathbf{x}) = \mathbb{1}[\mathbf{y}(\mathbf{x}) \succeq \tau]$ shows whether the point lies within the $\mathcal{S}$ region. The GP surrogate models are then used to calculate the probability of this occurring, denoted as $p(Z(\mathbf{x}) = 1 \mid \mathcal{D}_t)$, through,

$$p(Z(\mathbf{x}) = 1 \mid \mathcal{D}_t) = \int_{\tau}^{\infty} \phi(y \mid \mu(\mathbf{x}), v(\mathbf{x})) dy \tag{7.6}$$

where ϕ is the Normal PDF, and $\mu(\mathbf{x})$ and $v(\mathbf{x})$ are the GP's posterior mean and variance, respectively, evaluated at $\mathbf{x}$. Finally, for m objectives modeled by m independent GPs, $p(Z(\mathbf{x}) = 1 \mid \mathcal{D}_t)$ is the product of probabilities associated with each model. With these definitions, equation (7.7) can now be estimated using the

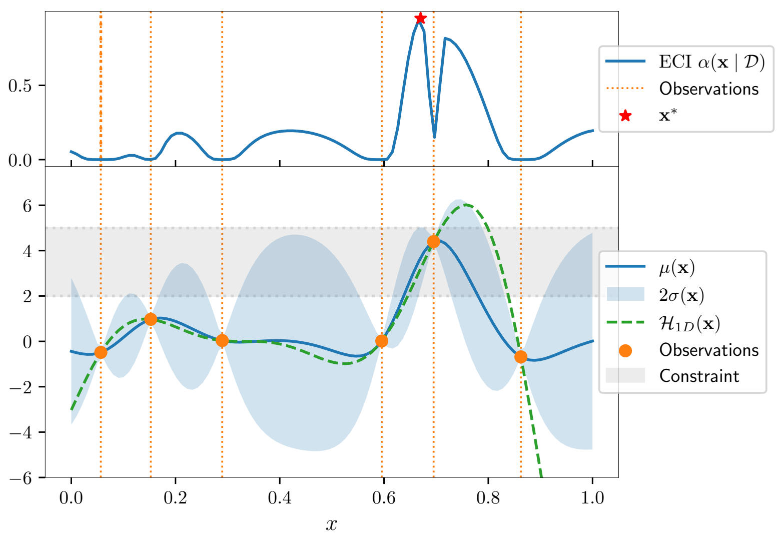


FIGURE 7.2: At the bottom, an example of a GP modelling a one-dimensional single-objective function $\mathcal{H}_{1D}$ is shown. The mean function and standard deviation are displayed, with low uncertainty in values near the observations. At the top, the ECI function, $\alpha(\mathbf{x} \mid \mathcal{D})$, is evaluated using the GP model depicted at the bottom. In this example the hyper-sphere radius parameter is $r = 0.01$. The figure demonstrates a CAS algorithm iteration, higher ECI regions lie within the constraints (grey band), with the optimal ECI point (red star) being the candidate point to query in $\mathcal{H}_{1D}$.

surrogate models, and is given by,

$$\alpha\left(\mathbf{x} \mid \mathcal{D}\right) = \mathbb{E}_Z\left[\text{Vol}\left(\{\mathbb{N}_r(\mathbf{x}) \cap \mathcal{S}_Z\} \setminus \mathbb{N}_r\left(\mathbf{X}\right)\right)\right] \tag{7.7}$$

Figure 7.2 shows $\alpha(\mathbf{x} \mid \mathcal{D})$ evaluated using the GP at the bottom for a one-dimensional single-objective function $\mathcal{H}_{1D}$. The figure demonstrates that regions with higher ECI values lie within the constraints, represented by the grey band. The star point marks the highest ECI evaluation, and sampling this point will increase the covered $\mathcal{S}$ volume in this one-dimensional example.

From a BO perspective, reviewed in Section 5.2, the ECI acquisition function fundamentally redefines the focus of optimisation, shifting it towards a search for diverse samples that lie within the satisfactory set.

7.5 b-CASTOR Batch Evaluation

Originally, the optimisation of ECI is made point-wise and sequentially, with well established routines, such as L-BFGS-B [162]. These classical optimisation methods are sufficient, given the original focus of ECI on experiment design, where the global search budget was relatively low, with order of $\mathcal{O}(10^2)$ points. However, in this work, our goal is to *densely populate* $\mathcal{S}$, *i.e.*, to collect as many samples from $\mathcal{S}$ as possible.

Since ECI, eq. (7.7), depends directly on the size of the dataset $\mathcal{D}_t$ at iteration t of the search process, the time required to optimise it increases accordingly. To evaluate equation (7.5) efficiently we use the Tree-structured Parzen Estimator (TPE) algorithm [157], introduced in Section 5.2.2.

The TPE optimisation process evaluates ECI a number of times to generate a historical dataset $\tilde{\mathcal{D}}_{\text{TPE}} := (\tilde{\mathbf{X}}, \boldsymbol{\alpha}(\tilde{\mathbf{X}}))$ known as trials, from which the optimal parameter value $\mathbf{x}^*$ in eq. (7.5) is identified. $\tilde{\mathcal{D}}_{\text{TPE}}$ contains parameter configurations with sub-optimal ECI values, but which lie within the satisfactory region $\mathcal{S}$. Hence, evaluating this subset of sub-optimal configurations on $\mathcal{H}$ accelerates the collection of parameter space points within $\mathcal{S}$. For this purpose, instead of selecting a single estimated maximal point $\mathbf{x}^*$ to evaluate $\mathcal{H}(\mathbf{x}^*)$, we sample a set of N_{batch} parameter points $\mathbf{X}^* = [\mathbf{x}_0^*, \mathbf{x}_1^*, .., \mathbf{x}_{N_{\text{batch}}}^*]$ from $\tilde{\mathcal{D}}_{\text{TPE}}$ and evaluate every point on $\mathcal{H}$ in each iteration of the search. This method, referred to as batch evaluation, accelerates the filling of the $\mathcal{S}$ region in the search process, as the HEP-Stack $\mathcal{H}$ allows parallel evaluations of each configuration in the batch.

The batch $\mathbf{X}^*$ is sampled according to a rank-based sampling strategy that interpolates between pure greedy prioritisation and uniform random sampling, initially developed in the context of Reinforcement Learning [179] and called stochastic prioritisation. In this scheme, each $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ is assigned a rank $rk(\tilde{\mathbf{x}})$ so that,

$$rk(\tilde{\mathbf{x}}_i) \leq rk(\tilde{\mathbf{x}}_j) \text{ for } \alpha(\tilde{\mathbf{x}}_i) \geq \alpha(\tilde{\mathbf{x}}_j). \tag{7.8}$$

This determines the probability of sampling

$$P(\tilde{\mathbf{x}}_i) = \frac{rk(\tilde{\mathbf{x}}_i)^{-\beta}}{\sum_{\tilde{\mathbf{x}}_j \in \tilde{\mathbf{X}}} rk(\tilde{\mathbf{x}}_j)^{-\beta}}, \tag{7.9}$$

where β determines the extent to which $\alpha(\tilde{\mathbf{x}})$ prioritises selection, with $\beta = 0$ corresponding to uniform sampling. Thus, points with higher ECI value will be more likely to be sampled, while also enabling exploration of the parameter space by sampling low value ECI.

7.5.1 b-CASTOR algorithm

We named this batched CAS algorithm as b-CASTOR, which stands for **b**atched **C**onstrained **A**ctive **S**earch with **TPE** **O**ptimisation and **R**ank based sampling. Here the search policy involves three main techniques: the ECI acquisition function, the TPE algorithm to optimise it, and the Stochastic Prioritisation sampling method from the optimisation history to provide the batch $\mathbf{X}^*$. TPE is used to accelerate the optimisation step in each iteration of b-CASTOR. This enables the collection of larger datasets during the search. Stochastic Prioritisation facilitates a faster convergence and provide control over the amount of exploration. The pseudo-code is described in Algorithm 1.

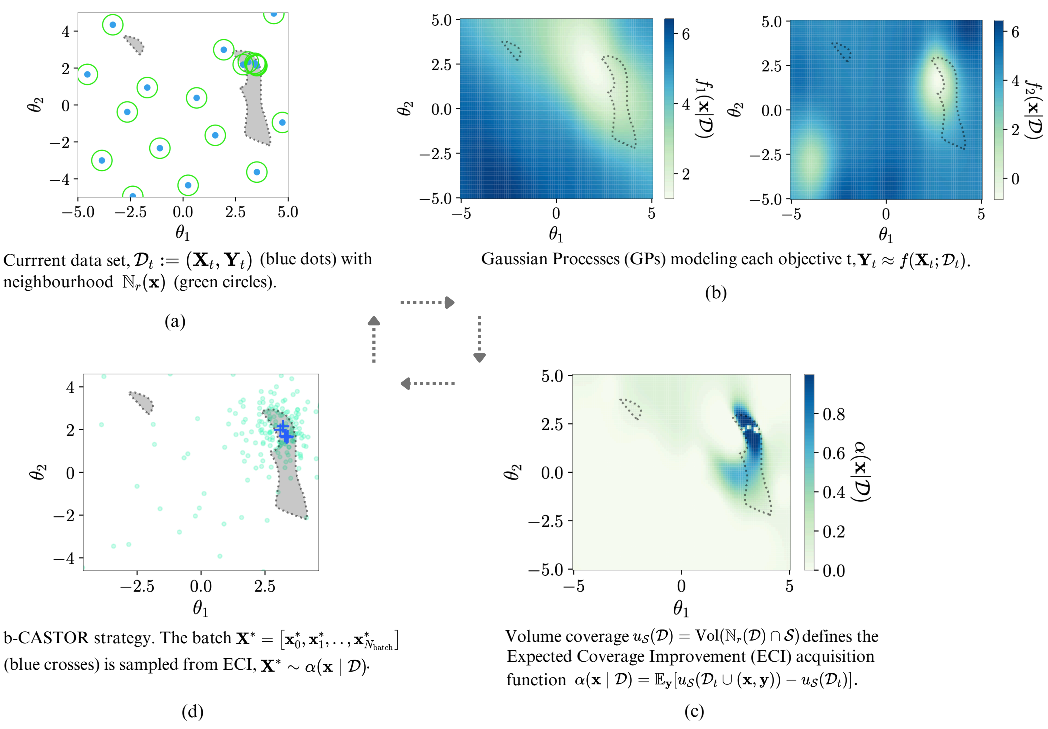
Algorithm 1 b-CASTOR

```
1: Initialise parameters: $N_0$ (number of initial points), $N_{\text{TPE}}$ (number of TPE trials), $N_{\text{batch}}$
   (batch size), $T$ (number of search iterations) and $\beta$ (prioritisation parameter).
2: Generate a initial dataset $\mathcal{D}_0$ with $N_0$ points.
3: for $i = 1, T$ do
4:     Fit Surrogate models to $\mathcal{D}_i$.
5:     Optimise ECI using TPE algorithm with $N_{\text{TPE}}$ trials generating $\tilde{\mathcal{D}}_{\text{TPE}}$.
6:     Assign a rank $rk(\tilde{\mathbf{x}})$ for each $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ in $\tilde{\mathcal{D}}_{\text{TPE}}$ following eq. (7.8).
7:     Sample a batch $\mathbf{X}^* = [\mathbf{x}_0^*, \mathbf{x}_1^*, .., \mathbf{x}_{N_{\text{batch}}}^*]$ from $\tilde{\mathcal{D}}_{\text{TPE}}$ using probabilities $P(\tilde{\mathbf{x}}_i)$ from eq.
       (7.9).
8:     Evaluate $\mathbf{X}^*$ in HEP-Stack $\mathcal{H}_{\text{Model}}$
9:     Update $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\mathbf{X}^*, \mathbf{Y}^*)$
10: end for
```

Specifically, the algorithm starts by initialising a specific number of points[1], denoted as N_0, creating the initial dataset $\mathcal{D}_0$. Each search iteration t involves fitting an independent GP model to each objective, using the current observation dataset $\mathcal{D}_t$. The ECI policy is then optimised and sampled by the b-CASTOR strategy, detailed in Section 7.5, obtaining the batch of points $\mathbf{X}^*$. Each sample point in $\mathbf{X}^*$ is then evaluated in the HEP-Stack $\mathcal{H}_{\text{Model}}$ under study and the dataset $\mathcal{D}_{t+1}$ is updated according to $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (\mathbf{X}^*, \mathbf{Y}^*)$. This iterative process continues until it reaches the pre-established number of total iterations, T_{iter}, or when a total number of samples is met, denoted by T_{samples}.

A schema illustrating an iteration at step t of the b-CASTOR Algorithm is shown in Figure 7.3. In this example, the search is conducted within a 2D double-objective function, defined in Section 7.6.1, where the region $\mathcal{S}$ is depicted as a the shaded area. The process is detailed as follows: (a) For each point in the current dataset, the neighbourhood is defined as a centred disk with a predefined radius r. (b) Independent GPs are used to model each objective dimension in $\mathbf{Y}$ given the current dataset. (c) These GPs are employed to evaluate the ECI function over the whole search space (possible in this toy example), where high ECI values suggest a greater likelihood of expanding the volume coverage of the $\mathcal{S}$ region by the current dataset. (d) Using the b-CASTOR sampling strategy, described in 7.5.1, a batch $\mathbf{X}^*$ is selected from the ECI to ensure a diverse collection of samples from areas with high ECI values, the batch is represented with the blue crosses. This batch is then evaluated on the objective function, updating the current dataset for the subsequent iteration, $t + 1$. During the search, the radius defining the neighbourhood undergoes a linear decay to ensure a dense filling of the $\mathcal{S}$ region.

[1]We use a Sobol sequence [180], a quasi-random low-discrepancy sequence of points in the search space.



Currrent data set, $\mathcal{D}_t := (\mathbf{X}_t, \mathbf{Y}_t)$ (blue dots) with neighbourhood $\mathbb{N}_r(\mathbf{x})$ (green circles).

(a)

Gaussian Processes (GPs) modeling each objective t, $\mathbf{Y}_t \approx f(\mathbf{X}_t; \mathcal{D}_t)$.

(b)

b-CASTOR strategy. The batch $\mathbf{X}^* = [\mathbf{x}_0^*, \mathbf{x}_1^*, .., \mathbf{x}_{N_{\text{batch}}}^*]$ (blue crosses) is sampled from ECI, $\mathbf{X}^* \sim \alpha(\mathbf{x} \mid \mathcal{D})$.

(d)

Volume coverage $u_{\mathcal{S}}(\mathcal{D}) = \text{Vol}(\mathbb{N}_r(\mathcal{D}) \cap \mathcal{S})$ defines the Expected Coverage Improvement (ECI) acquisition function $\alpha(\mathbf{x} \mid \mathcal{D}) = \mathbb{E}_{\mathbf{y}}[u_{\mathcal{S}}(\mathcal{D}_t \cup (\mathbf{x}, \mathbf{y})) - u_{\mathcal{S}}(\mathcal{D}_t)]$.

(c)

FIGURE 7.3: Schema of an iteration at step t of the b-CASTOR Algorithm. In this example, the search is performed in a $2D$ double-objective function, defined in section 7.6.1 with $\mathcal{S}$ displayed as the shaded area. (a) For each point in the current dataset, the neighbourhood is defined as a centered disk. (b) Independent GPs model each objective dimension in $\mathbf{Y}$. (c) The GPs are used to calculate the ECI function. Regions with high ECI values indicate where it is more likely to increase the volume coverage of the $\mathcal{S}$ region. (d) The b-CASTOR sampling strategy is employed to select a batch $\mathbf{X}^*$ from the ECI, aiming to obtain a diverse set of samples from regions with high ECI values. The objective function is then evaluated on this batch, updating the current dataset for the next iteration $t + 1$.

7.5.2 Performance Study

We compare our algorithm with a Markov chain Monte Carlo (MCMC) method, specifically, the Metropolis-Hastings (MH) algorithm [181, 39] (hereafter, denoted by MCMC-MH). The sampling with the MH is performed with the construction of a joint likelihood for the objectives. For each objective y_i, constrained by either a threshold a or a window with limits $[a, b]$, we define a likelihood given by

$$\mathcal{L}(y_i) = \begin{cases} \sigma(y_i, a) & y_i > a \\ 1 - \sigma(y_i, a) & y_i < a \\ \sigma(y_i, a) - \sigma(y_i, b) & a < y_i < b \end{cases} \tag{7.10}$$

where σ is the Sigmoid function and is defined as,

$$\sigma(y, a) = \frac{1}{1 + e^{-(y-a)/\epsilon}} \tag{7.11}$$

Here ϵ is a parameter that controls the smoothness of the Sigmoid function and a shifts the center of the Sigmoid. Then, the total likelihood for a point $(\mathbf{x}, \mathbf{y})$ used for MCMC-MH sampling is defined as,

$$\mathcal{L}(\mathbf{y}) = \prod_i \mathcal{L}(y_i) \tag{7.12}$$

Given an initial proposal step-size, we adjust this step-size actively to reach a target acceptance rate of 0.234 [182]. The scale is increased by 10% if the acceptance rate is above a the threshold and decreased by 10% if the acceptance rate is below.

Lastly, two main metrics are monitored for both search strategies. The satisfactory points per objective function call and the ratio of satisfactory points to total points in the dataset per call.

In section 7.6, we perform a grid hyper-parameter search for N_{TPE}, β and r (defined in eq. (7.2)) for a test objective function. We also implement a linear decay in $r = \{r_{\text{initial}}, r_{\text{final}}\}$ along the search process. This configuration on r enables an early discovery of the $\mathcal{S}$ region and subsequent fine resolution filling. The values of r_{initial} and r_{final} are also considered in the grid hyper-parameter search.

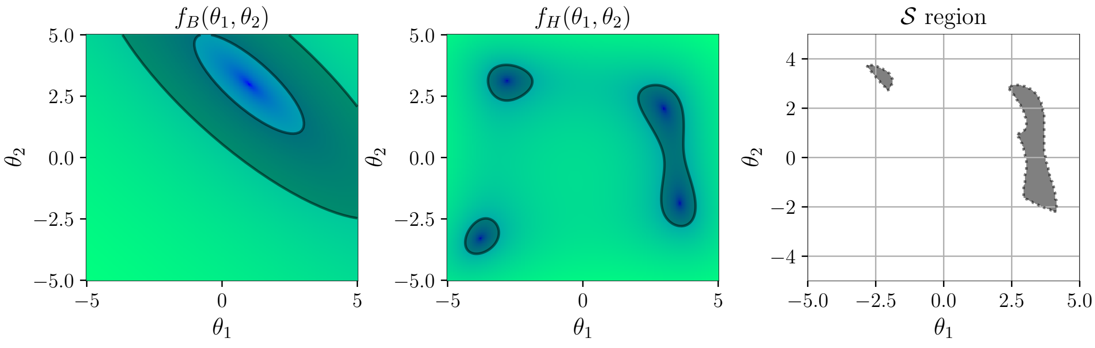


FIGURE 7.4: Ground truth for the 2D double-objective test function $\mathbf{f}_{BH}(\theta_1, \theta_2)$ as per eq. (7.13), featuring contour levels to demonstrate the constraints on the objectives. On the left, the $\mathcal{S}$ region within the search space is depicted.

7.6 Results

7.6.1 Double-objective 2D Test Function

We constructed a simple double-objective, 2D test function, denoted $\mathbf{f}_{BH}$(`), from the Booth and Himmelblau functions [183], a uni- and multi-modal function, respectively, defined as

$$\mathbf{f}_{BH}(\boldsymbol{\theta}) = \begin{cases} f_B(\theta_1, \theta_2) = \log\left[(\theta_1 + 2\theta_2 - 7)^2 + (2\theta_1 + \theta_2 - 5)^2\right], \\ f_H(\theta_1, \theta_2) = \log\left[\left(\theta_1^2 + \theta_2 - 11\right)^2 + \left(\theta_1 + \theta_2^2 - 7\right)^2\right]. \end{cases} \tag{7.13}$$

The parameter space is defined as ` $\in [-5, 5]$. Additionally, the search process is subject to the constraints on the objectives given by

$$\boldsymbol{\tau}_{\mathbf{f}_{BH}} = \begin{cases} f_B(\theta_1, \theta_2) & = 2 \pm 1, \\ f_H(\theta_1, \theta_2) & < 3. \end{cases} \tag{7.14}$$

This simple set-up imitates the complexity of sparse and disconnected satisfactory regions in the search space of multiple objectives. The ground truth satisfactory regions for both objectives in $\mathbf{f}_{BH}(\theta_1, \theta_2)$ are shown in Figure 7.4.

The original CAS algorithm, a sequential point-wise search process employing a classical optimisation method, is illustrated in Figure 7.5. The figure depicts the constraints on the objectives as solid black lines across all panels, while the dashed lines represent the GP-approximated $\mathcal{S}$ boundaries at the final iteration. The first two panels, from left to right, show the GP approximation of the objectives over the entire search space. The rightmost panel presents the search results, with sampled points displayed as pink scatter dots. Even without our modified optimisation strategy, CAS demonstrates promising results, characterising $\mathcal{S}$ with a pseudo-grid of points. The distance between points is approximately $\sim r$, where $r = 0.01$ (in this case, the search space is normalised to $[0, 1]$).

For b-CASTOR, we performed a hyper-parameter grid search for the number of policy evaluations N_{TPE}, priority scaling β (eq. (7.9)) as well as the parameter resolution limits r_{initial} and r_{final} in eq. (7.2). The hyper-parameter search space is defined by the set of values in Table 7.1, together with the fixed hyper-parameters. We selected $N_{\text{TPE}} = 500$, $\beta = 2$, $r_{\text{initial}} = 0.02$ and $r_{\text{final}} = 0.0002$. For the MCMC-MH algorithm, an initial step size of 0.4 was established, optimised for the potential discovery of the disconnected $\mathcal{S}$ regions. In order to evaluate the consistency of convergence for both the b-CASTOR and MCMC-MH algorithms, we performed 10 searches. For this we have set the hyper-parameters as mentioned and restricted the search to 2200 calls to the objective test function. This was done in anticipation of the application to querying the HEP-Stack, where the complexity of the search is dominated by time

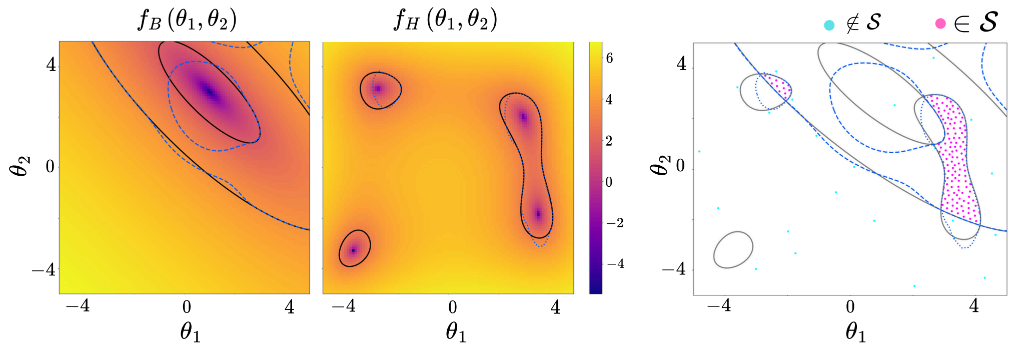
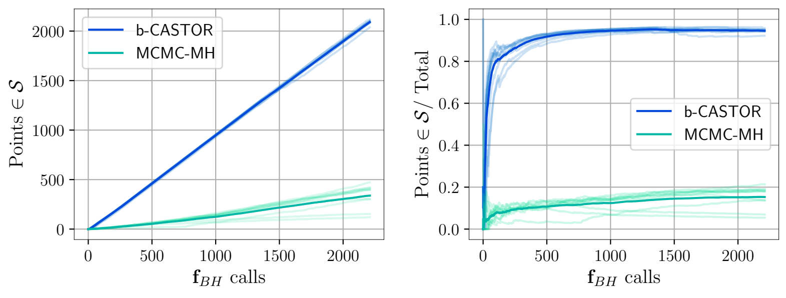
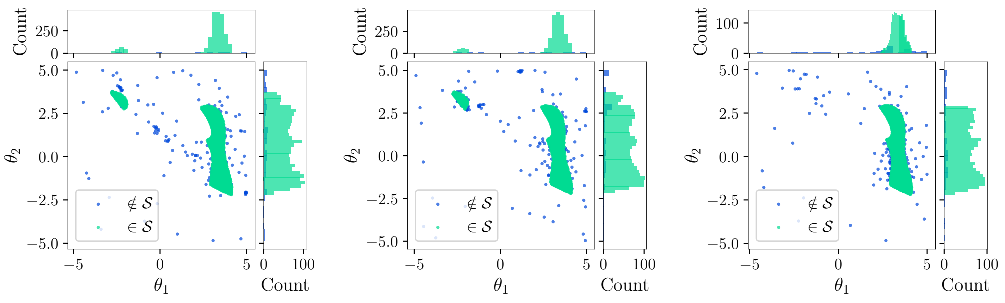


FIGURE 7.5: Original CAS algorithm: a sequential point-wise search using classical optimisation for the ECI function. The solid black line indicates constraints on the objectives, while dashed lines represent the GP-approximated $\mathcal{S}$ boundaries at the final iteration. The first two panels depict the GP approximation of objectives across the search space, and the rightmost panel shows the search result with satisfactory sampled points as pink scatter dots, non-satisfactory are light green dots.



(A) $\mathcal{S}$ points per call (B) $\mathcal{S}/T$ points per call

FIGURE 7.6: Performance metrics across ten independent runs for b-CASTOR (blue) and MCMC-MH (green), with mean values depicted in darker shades.



(A) (B) (C)

FIGURE 7.7: b-CASTOR results for different independent runs.

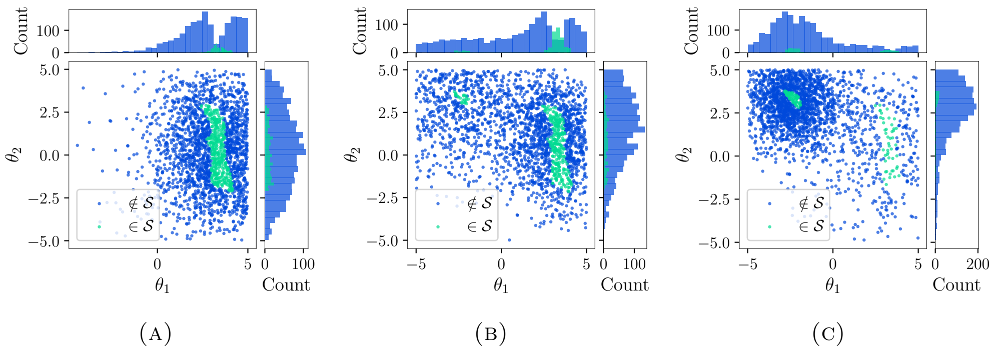


FIGURE 7.8: MCMC-MH results for different independent runs.

needed for evaluating each query. The performance metrics are presented in Figure 7.6. b-CASTOR achieved an average of 2090 satisfactory parameter values by the end of the search (excluding the initial dataset), indicating that 94.57% of the objective function calls resulted in parameter space configurations that satisfy the constraints on the objectives. In contrast, MCMC-MH recorded an average of 338 satisfactory points at the end of the search, corresponding to 15.29% of the total calls. Out of the 10 searches conducted, the results of three runs per algorithm are illustrated in Figures 7.7 and 7.8.

Hyper-parameter	Value
N_0	10
N_{batch}	10
T_{samples}	2200
N_{TPE}	$\{100, 300, 500\}$
β	$\{1, 2\}$
$(r_{\text{initial}}, r_{\text{final}})$	$\{(0.2, 0.02), (0.02, 0.002), (0.02, 0.0002)\}$

TABLE 7.1: b-CASTOR hyper-parameters for the search in $\mathbf{f}_{\text{BH}}$. Values in brackets define the hyper-parameter grid search space.

The superior performance of b-CASTOR over MCMC-MH is evident from the results shown in these plots. A high number of TPE trials allows the collection of a large number of samples, which are distributed across high values of the ECI acquisition function. By setting a quadratic priority parameter, $\beta = 2$, the ranked-sampling strategy is adjusted to strictly favour exploitation over exploration. Consequently, in each search iteration, the proposed batch $\mathbf{X}^*$ is more likely to contain a set of parameter values that meet all the constraints – a satisfactory set – and show sufficient variability in the parameters found – a diverse set – leading to a high *sample efficiency* of the search. Figure 7.7 demonstrates that with b-CASTOR the $\mathcal{S}$ region can be accurately characterised without having to explore the entire search space. In contrast, MCMC-MH in Figure 7.8 exhibits lower sample efficiency, with many evaluations spread across areas surrounding the $\mathcal{S}$ region.

b-CASTOR preserves the sampling efficiency of the original CAS algorithm, as illustrated in Figure 7.5. However, it leverages ECI information to sample more densely in uncovered regions of $\mathcal{S}$, rather than simply forming a pseudo-grid of points.

Furthermore, as mentioned in section 7.5.1, setting the radius parameter r in eq. (7.2) to have a linear decay allows early exploration when the initial size of the radius is relatively big compared to the search space leading to a comprehensive initial estimation of the satisfactory region. Subsequently, since the radius decreases with each iteration, the $\mathcal{S}$ region gets densely populated. However, sections of the $\mathcal{S}$ region not identified in the initial low-resolution phase are likely to remain undiscovered. This behaviour is observed in Figure 7.7 (c), where a small segment of the $\mathcal{S}$ region in the top-left quadrant remains undetected. Nonetheless, b-CASTOR extends its exploration to uncover additional sections of the $\mathcal{S}$ region once the previously discovered areas are fully covered, by leveraging the uncertainty estimation of the surrogate model on unexplored regions in the search space.

7.7 The $(B-L)$SSM and a 95 GeV Higgs Boson

Results for new Higgs boson experimental searches are introduced in Section 2.3.2 and are briefly reviewed here for convenience. Combined results from CMS and ATLAS in the di-photon decay channel reported excesses of 2.9σ and 1.7σ, respectively, at a mass of 95.4 GeV. LEP experiments observed a 2.3 σ excess in the $e^+e^- \to Z(H \to b\bar{b})$ channel at a Higgs mass of 98 GeV. Finally, CMS detected a 2.6σ excess in the gluon-fusion production mode with decay into $\tau^+\tau^-$ pairs at a mass of 95 GeV. The signal strength modifiers reported for those results are:

$$\mu_{\gamma\gamma}^{\text{exp}} = \mu_{\gamma\gamma}^{\text{ATLAS+CMS}} = \frac{\sigma^{\text{exp}}(gg \to \phi \to \gamma\gamma)}{\sigma^{\text{SM}}(gg \to H \to \gamma\gamma)} = 0.27^{+0.10}_{-0.09}, \tag{7.15}$$

$$\mu_{bb}^{\text{exp}} = \frac{\sigma\left(e^+e^- \to Z\phi \to Zb\bar{b}\right)}{\sigma^{SM}\left(e^+e^- \to ZH \to Zb\bar{b}\right)} = 0.117 \pm 0.057, \tag{7.16}$$

$$\mu_{\tau\tau}^{\text{exp}} = \frac{\sigma^{\text{exp}}\left(gg \to \phi \to \tau^+\tau^-\right)}{\sigma^{\text{SM}}\left(gg \to H \to \tau^+\tau^-\right)} = 1.2 \pm 0.5. \tag{7.17}$$

Together, these results provide strong indications of potential new physics, potentially involving a light neutral scalar.

The $(B-L)$SSM is introduced in , In Section 2.6, the neutral Higgs bosons of the $(B-L)$SSM model were introduced. In this model, the CP-even sector results in four physical scalar states (h_1, h_2, h_3, h_4) we then perform a parameter search, to fit the two lighter Higgs states in, as solutions consistent with the experimental reports in equation 7.15.

The dimensionality of the search space is reduced by fixing $m_{Z'} = 2500$ GeV, $\tan\beta' = 1.15$, $g_{BL} = 0.53$ and $g'_{BL} = 0.14$ [184, 113, 185], restricting the search space $\mathcal{X}$ to eight model parameters

$$\mathcal{X} = \left\{x \in \mathbb{R}^8 : x = (M_0, M_{1/2}, \tan\beta, A_0, \mu, \mu', B_\mu, B_{\mu'})\right\} \tag{7.18}$$

in the ranges described in Table 7.2.

Parameter	Range
M_0	$100 - 1000$ GeV
$M_{1/2}$	$1000 - 4500$ GeV
$\tan\beta$	$1 - 60$
A_0	$1000 - 4000$ GeV
μ	$1000 - 4000$
μ'	$1000 - 4000$
B_μ	$10^5 - 10^7$
$B_{\mu'}$	$10^5 - 10^7$

TABLE 7.2: Ranges defining the search space for each parameter in the $(B-L)$SSM.

We define the objective space $\mathcal{Y}$ as the space of physical observables and informative outputs generated by the HEP-Stack $\mathcal{H}_{(B-L)\text{SSM}}$. These outputs are the desired targets that we seek to constrain to specific values. Specifically, we aim for the masses of the lighter Higgs particles in the model, denoted as m_h and $m_{h'}$, to have mass values of 125 GeV and 95 GeV respectively, with a certain precision. Additionally, the signal strength modifier $\mu^{\gamma\gamma}$ should satisfy the experimental value defined in (2.30). Lastly, we require the experimental checks from HB and HS to yield positive outcomes, ensuring that $k_0^{\text{HB}} \leq 1$ and $\chi^2_{\text{HS}} \leq 136.6$, these are the default values of the two programs. Thus, we formulate a five-dimensional objective space $\mathcal{Y}$ as follows:

$$\mathcal{Y} = \left\{y \in \mathbb{R}^5 : y = \left(m_{h'}, m_{h^{\text{SM}}}, \mu^{\gamma\gamma}, \chi^2_{\text{HS}}, k_0^{\text{HB}}\right)\right\} \tag{7.19}$$

with constraints defined in a vector τ for latter reference,

$$\boldsymbol{\tau}_{\gamma\gamma} = \begin{cases} m_{h^{\text{SM}}} & = 125 \pm \delta m \text{ GeV} \\ m_{h'} & = 95 \pm \delta m \text{ GeV} \\ \mu^{\gamma\gamma} & = 0.27^{+0.10}_{-0.09} \\ \chi^2_{\text{HS}} & \leq 136.6 \\ k_0^{\text{HB}} & \leq 1 \end{cases} \tag{7.20}$$

where we take $\delta m = 5$ GeV as an acceptance window for the masses. As previously mentioned, for a particular parameter space configuration $\mathbf{x} \in \mathcal{X}$, the HEP-Stack

$\mathcal{H}_{(B-L)\text{SSM}}$ to evaluate is formed by SP, HB, HS and MG[2] (see Section 3.2.2). High-precision spectrum calculations typically require approximately 120 seconds on average for each query to the HEP-Stack. However, in certain parameter space configurations, this time can extend up to 300 seconds. The computational cost emphasises the need for the search algorithm to prioritise *sampling efficiency*, which means maximising the number of positive parameter space configurations per $\mathcal{H}_{(B-L)\text{SSM}}$ evaluations.

7.7.1 The $(B-L)$SSM and a 95 GeV Higgs Boson

We now examine the first phenomenology case study with $(B-L)$SSM model. The objective function is defined as follows:

$$\mathcal{H}_{(B-L)\text{SSM}} : (M_0, M_{1/2}, \tan\beta, A_0, \mu, \mu', B_\mu, B_{\mu'}) \to \left(m_{h'}, m_{h^{\text{SM}}}, \mu^{\gamma\gamma}, \chi^2_{\text{HS}}, k_0^{\text{HB}}\right)$$

where each objective is constrained by $\boldsymbol{\tau}$ specified in eq. (7.20). One issue that arises in sampling methods for BSM phenomenology is to guarantee the physical validity of each parameter configuration. In certain BSM models, SP fails to converge to a physical spectra for a significant portion of points within the search space, being the case for the $(B-L)$SSM. In [5] they addressed this challenge by employing a NN classifier as a preliminary step to regression on observables. In [186] they include these points as points outside the satisfactory set, assigning them a zero likelihood. In this work, for we discard the non-physical points and only work with valid parameter space configurations.

The hyper-parameters for the b-CASTOR search are outlined in table 7.3. We have allocated a greater number of TPE trials compared to the test function. This decision is based on findings from section 7.6.1, which demonstrated that an increase in TPE trials enhances the sample efficiency of the b-CASTOR search process.

Figure 7.9 illustrates the performance of both algorithms throughout the search process. b-CASTOR identified 1636 satisfactory configurations, constituting up to 50% of the total $\mathcal{H}_{(B-L)\text{SSM}}$ calls, which amounted to 3240. In contrast, MCMC-MH was able to find only 25 satisfactory configurations, representing a mere 0.008% of the total calls.

The results obtained from each algorithm, b-CASTOR and MCMC-MH, are depicted in Figures 7.10 and 7.11, respectively, using corner plots. These plots constitute a triangular grid of 2-Dimensional (2D) scatter plots for each pair of variables, supplemented by marginal histograms for individual variables. The figures integrate a

[2]In this analysis, the gluon fusion production of the 95 GeV Higgs boson was computed using an effective Higgs–gluon–gluon vertex. While this approach can lead to inaccuracies in scenarios with significant loop contributions or suppressed top couplings, it remains a reasonable approximation.

Hyper-parameter	Value
N_0	400
N_{batch}	30
T_S (Total samples)	3240
N_{TPE}	2500
β	2
$(r_{\text{initial}}, r_{\text{final}})$	$(10^{-2}, 10^{-6})$

TABLE 7.3: b-CASTOR hyper-parameters for the search in $\mathcal{H}_{(B-L)\text{SSM}}$.

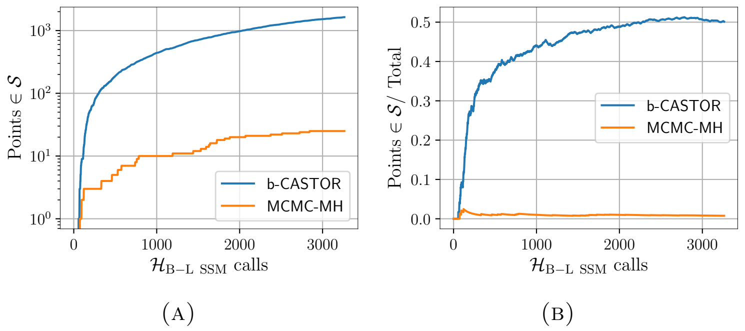


FIGURE 7.9: Performance metrics for b-CASTOR (blue) and MCMC-MH (orange), for the search in $\mathcal{H}_{(B-L)\text{SSM}}$ fitting $\mu_{\gamma\gamma}^{\text{exp}}$.

selection of relevant dimensions from both the search space, $\mathcal{X}$, and the objective space, $\mathcal{Y}$; specifically, $\{M_0, M_{1/2}, \tan\beta, A_0\}$ from $\mathcal{X}$ and $\{m_{h'}, m_{h^{\text{SM}}}, \mu^{\gamma\gamma}\}$ from $\mathcal{Y}$. The dimensions of interest in the objective space are highlighted with green axes titles and green bands marking the constraints, defined in eq. (7.20), within each plot.

Upon comparing Figures 7.10 and 7.11, it is evident that, with an equivalent number of function calls, b-CASTOR comprehensively characterises the $\mathcal{S}$ region in a sample efficient manner, in contrast to MCMC-MH, which encounters difficulties in accurately characterising this region. b-CASTOR concentrates on exploring the neighbourhood areas of the identified portion of the $\mathcal{S}$ region, simultaneously ensuring these portions are densely populated. Conversely, MCMC-MH explores a more extensive area within the valid search space, centred around a few $\mathcal{S}$ points, as we can read from the blue marginal plots of the search space dimensions $\{M_0, M_{1/2}, \tan\beta, A_0\}$ in Figure 7.11, without prioritising samples within the $\mathcal{S}$ region. This sampling behaviour results from the nature of the Gaussian proposal distribution in the MCMC-MH algorithm.

Subsequently, μ_{bb} can be included in the objectives, using the experimental value defined in eq. (2.31). The objective space is then defined as follows:

$$\mathcal{Y}_{\gamma\gamma+bb} = \left\{ y \in \mathbb{R}^6 : y = \left(m_{h'}, m_{h^{\text{SM}}}, \mu_{\gamma\gamma}, \mu_{bb}, \chi^2_{\text{HS}}, k_0^{\text{HB}} \right) \right\} \tag{7.21}$$

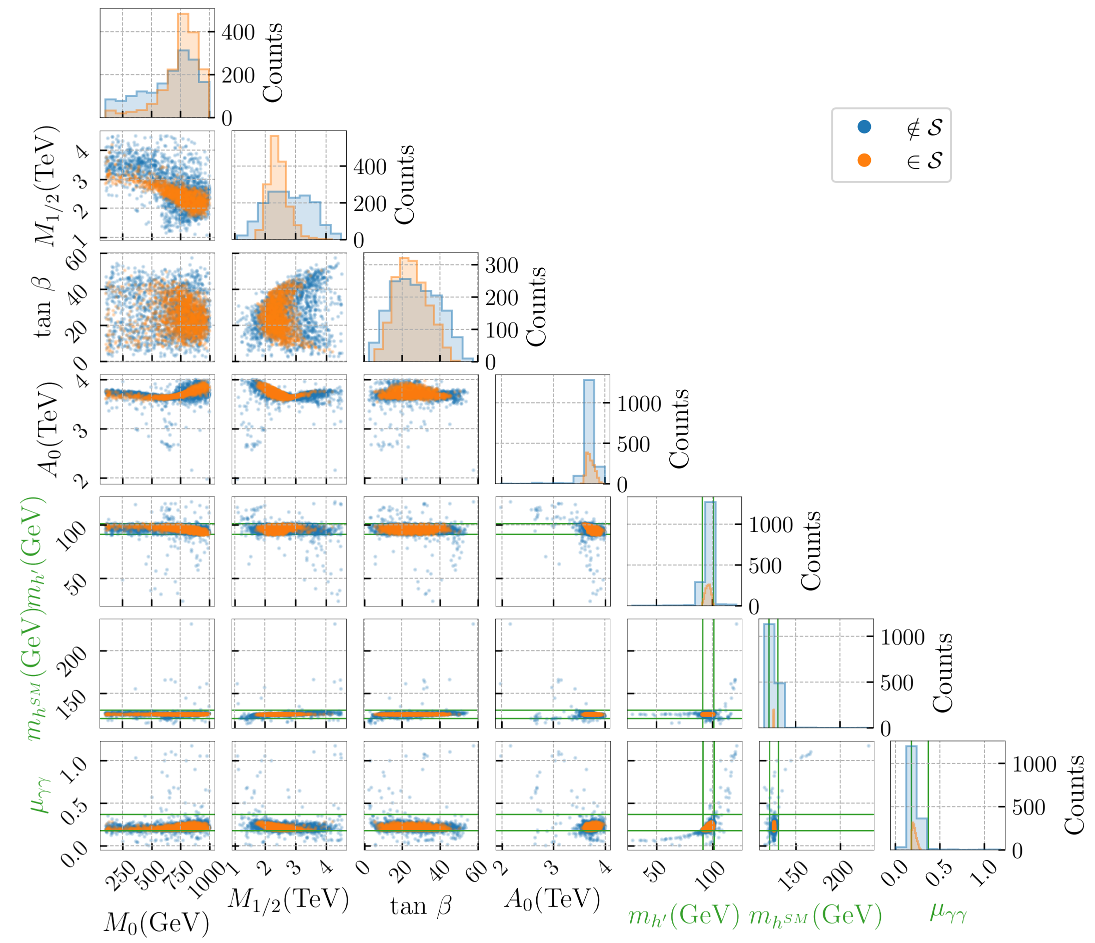


FIGURE 7.10: b-CASTOR results for $\mathcal{H}_{(B-L)\text{SSM}}$ fitting $\mu^{\text{exp}}_{\gamma\gamma}$. A selection of dimensions from both the search and objective spaces is presented; $\{M_0, M_{1/2}, \tan\beta, A_0\}$ (with black axis titles) and $\{m_{h'}, m_{h\text{SM}}, \mu^{\gamma\gamma}\}$ (with green axis titles), respectively. Green bands represent the experimental constraints on the objectives.

constrained to

$$\boldsymbol{\tau}_{\gamma\gamma+bb} = \begin{cases} m_{h^{\text{SM}}} & = 125 \pm \delta m \text{ GeV} \\ m_{h'} & = 95 \pm \delta m \text{ GeV} \\ \mu^{\gamma\gamma} & = 0.27^{+0.10}_{-0.09} \\ \mu_{bb} & = 0.117 \pm 0.057 \\ \chi^2_{\text{HS}} & \leq 136.6 \\ k_0^{\text{HB}} & \leq 1 \end{cases} \tag{7.22}$$

where we considered $\delta m = 5$ GeV. Incorporating μ_{bb} necessitates the addition of the relevant cross-section, computed by MG, thereby increasing the computational cost associated with $\mathcal{H}_{(B-L)\text{SSM}}$. Consequently, we conduct the b-CASTOR search utilising the same hyper-parameters as in Table 7.3, albeit with a reduced total sample budget of ~ 1000 function calls, owing to the increased computational cost. b-CASTOR achieved 5% of satisfactory points, which corresponds to ~ 50 points. Under identical

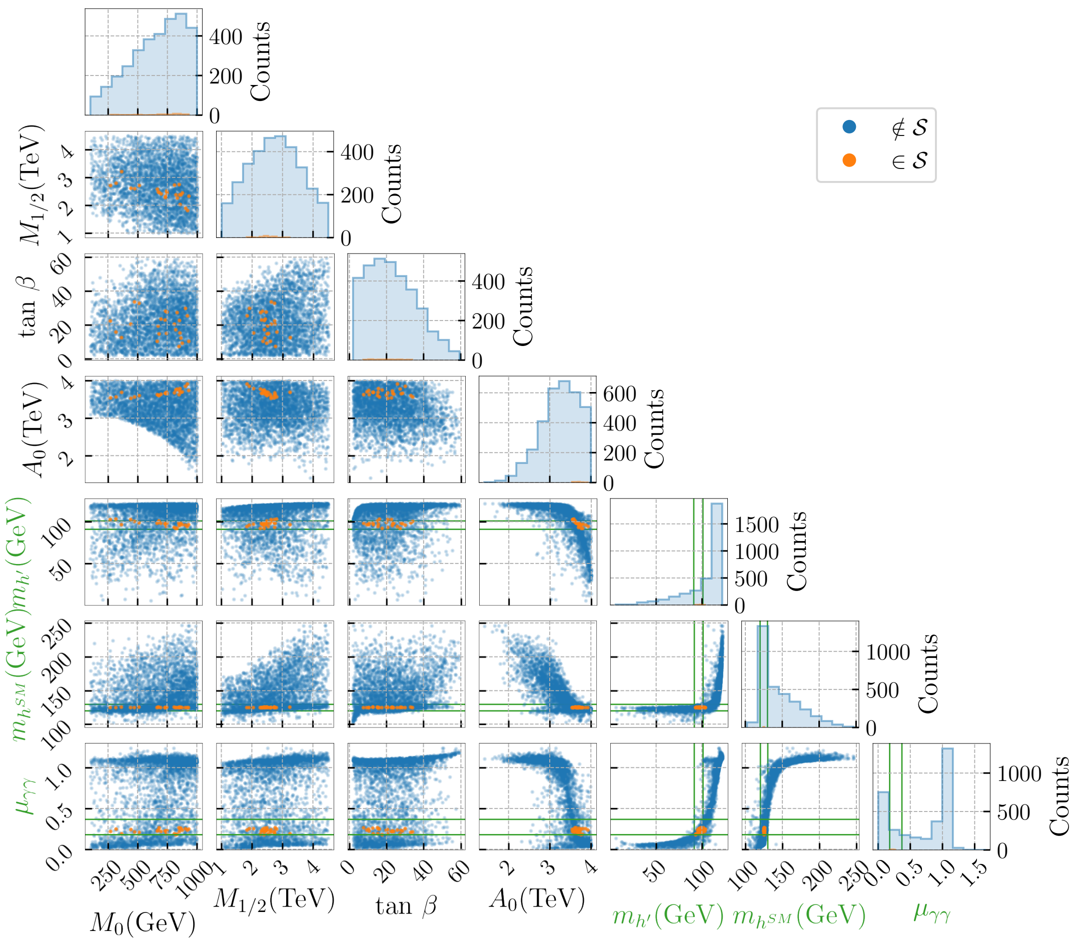


FIGURE 7.11: MCMC-MH results for $\mathcal{H}_{(B-L)\mathrm{SSM}}$ fitting $\mu^{\exp}_{\gamma\gamma}$. A selection of dimensions from both the search and objective spaces is presented; $\{M_0, M_{1/2}, \tan\beta, A_0\}$ (with black axis titles) and $\{m_{h'}, m_{h^{\mathrm{SM}}}, \mu^{\gamma\gamma}\}$ (with green axis titles), respectively. Green bands represent the experimental constraints on the objectives.

settings, MCMC-MH failed to identify any satisfactory configurations restricted to the same number of function calls. We allowed MCMC-MH to continue running until it located at least one satisfactory point, achieving a rate of approximately 1 in 4000. The results for the discovered $\mathcal{S}$ points by b-CASTOR are shown in Figure 7.12.

Finally, we conducted a b-CASTOR search for the three reported signal-strength modifiers $\mu_{\gamma\gamma}$, μ_{bb} and $\mu_{\tau\tau}$, with experimental values defined in eqs. (2.30), (2.31) and (2.32), respectively. The search was unable to identify satisfactory parameter space configurations, suggesting that the $(B-L)$SSM cannot simultaneously accommodate the three signals, within the designated search parameter space $\mathcal{X}$, as defined in eq. (7.18), with parameter ranges specified in Table 7.2.

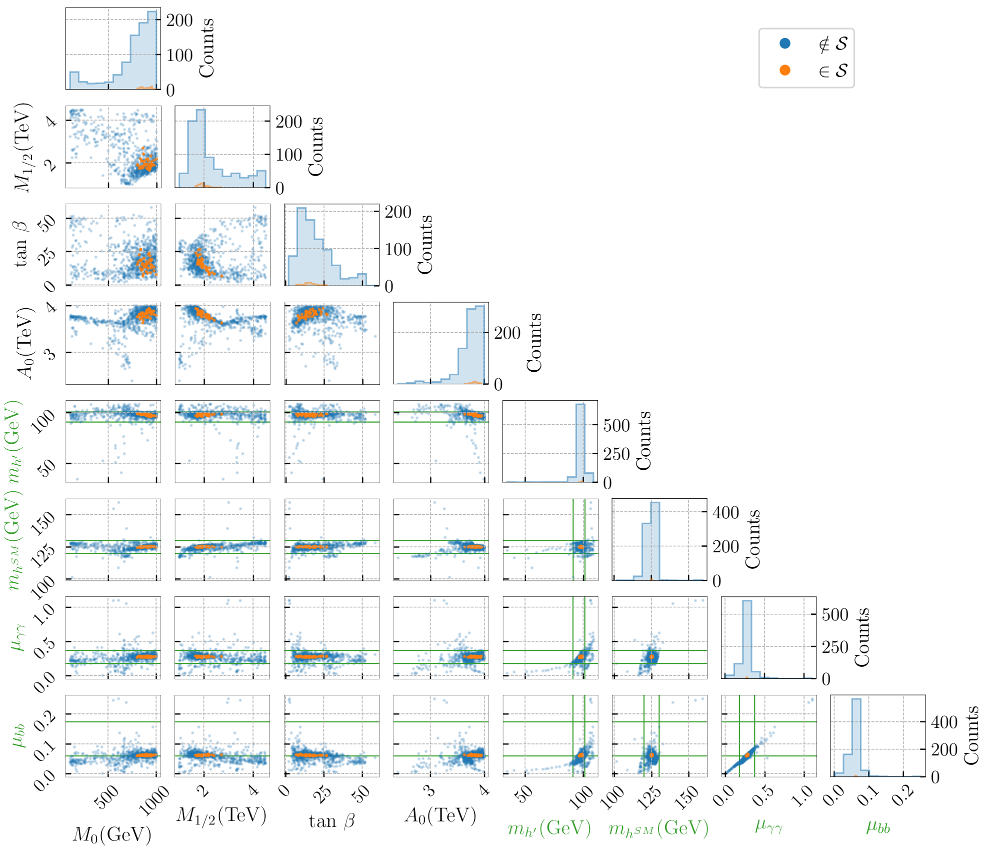


FIGURE 7.12: b-CASTOR results for $\mathcal{H}_{(B-L)\text{SSM}}$ fitting $\mu^{\text{exp}}_{\gamma\gamma}$ and μ^{exp}_{bb}. A selection of dimensions from both the search and objective spaces is presented; $\{M_0, M_{1/2}, \tan\beta, A_0\}$ (with black axis titles) and $\{m_{h'}, m_{h^{\text{SM}}}, \mu^{\gamma\gamma}, \mu^{bb}\}$ (with green axis titles), respectively. Green bands represent the experimental constraints on the objectives.

7.8 Computational Resources

For this work, Iridis 5 [187] was utilised. Iridis 5 represents the fifth generation of the University of Southampton's High-Performance Computing (HPC) facility. Specifically, the resources used to obtain the results presented in the previous section correspond to a single computing node. The number of CPU cores employed matched the size of the batch, denoted as N_{batch}. For example, in the case of Figure 7.10, where the b-CASTOR algorithm was employed with $N_{\text{batch}} = 30$, 30 CPUs within a single computing node were utilised over the course of two consecutive days.

For the comparison of algorithms focused on the number of samples, the MCMC-MH algorithm (Figure 7.11) was performed using 30 parallel chains, employing the same

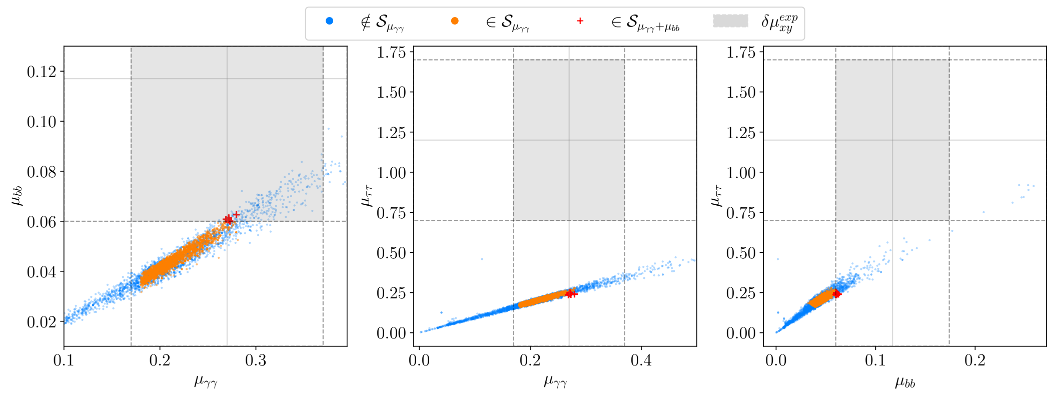


FIGURE 7.13: Scatter plots of the data set derived from the five-dimensional b-CASTOR search, evaluated in MG for $\mu_{\gamma\gamma}$, μ_{bb} and $\mu_{\tau\tau}$. It illustrates that while none of the points satisfy the three experimental signal-strengths modifier values simultaneously, a small set of points (marked with red crosses) meet the criteria for both $\mu_{\gamma\gamma}^{exp}$ and μ_{bb}^{exp}

number of CPUs. This was carried out until the number of points reached approximately 3200, as depicted in Figure 7.9.

7.9 Conclusions

In this chapter, we have introduced b-CASTOR, a novel multi-point multi-objective active search method for computationally expensive BSM scenarios. It effectively identifies $\mathcal{S}$, the satisfactory region of the corresponding parameter space that can accommodate a combination of desired values for the objectives while achieving high sample efficiency due to the use of probabilistic surrogate models for approximating the multiple objectives. It provides sample diversity in the search space by leveraging the ECI acquisition function, a volume based metric that operates to maximise the covered volume of the $\mathcal{S}$ region.

b-CASTOR was evaluated using two case studies. The first case involved a double-objective and a 2D test function designed to replicate the complexity of sparse and disjoint satisfactory regions. The second case focused on BSM phenomenology, specifically employing the $(B-L)$SSM scenario to allocate the observed signal at approximately 95 GeV from Higgs searches. Specifically, in this work, we attempt to find explanations to three possible data anomalies emerged at the above mass value in the $\gamma\gamma$, $\tau\tau$ (at the LHC) and $b\bar{b}$ (at LEP) invariant masses.

We conducted a comparative analysis between our proposed search method, b-CASTOR, and a MCMC-MH. Our findings illustrate the effectiveness of our algorithm in characterising the $\mathcal{S}$ region within the parameter space of the

$(B-L)$SSM model, by efficiently finding solutions herein in the case of the longest established anomaly, i.e. the one in the $\gamma\gamma$ channel, obtained in the form a light scalar state, h. However, considering updated experimental data for the channels $h \to \tau\tau$ and $h \to b\bar{b}$, our approach did not find points capable of simultaneously explaining all three channels. Nonetheless, it identified points capable of satisfying the signal strengths $\mu_{\gamma\gamma}$ and μ_{bb} concurrently.

b-CASTOR is robust to the choice of the resolution parameter r, since the optimisation of the ECI was replaced by a stochastic sampling strategy. However, when r is gradually reduced the search is benefited by filling finely the unexplored satisfactory areas. Our experiments have consistently found that this configuration on r causes a large scale discovery of the $\mathcal{S}$ region early in the early stages of the search. The *exploration-exploitation trade-off* in our algorithm is determined by the interplay between the trials used for each ECI optimisation, the quantity of samples obtained through the Rank-based sampling strategy, and their prioritisation level. An increase in the number of ECI optimisation trials improves the accuracy of policy optimisation but makes surrogate model inference less time efficient. Then, a higher degree of prioritisation leads to an increase in sample efficiency but diminishes exploration, consequently reducing the potential for discovery of the satisfactory region. However, the issue of low exploration can be mitigated by increasing the number of Rank-based samples. Nonetheless, adding more samples per iteration results in slower re-training of the surrogate model in each iteration. Therefore, utilising probabilistic surrogate models capable of scaling to larger datasets and with faster inference time efficiency, such as Bayesian Neural Networks (BNNs) [188], represents a potential avenue for further development.

Chapter 8

hep-aid: A new computational framework

8.1 Introduction

In this chapter, we introduce a new Python library, `hep-aid`, which provides a modular framework for developing PS algorithms for BSM phenomenology and also adopts the principles of related libraries. It manages the HEP software and provides components to ease the utilisation, implementation, and development of PS algorithms for phenomenological studies. The library comprises two main modules: the `hep` module and the `search` module.

The `hep` module facilitates the integration of the HEP software ecosystem into Python scripts. It allows users to perform a first-principles computation of observable quantities to compare with experimental data for each parameter space point using a stack of HEP software, collecting the output with a single function call. From the HEP tools reviewed in chapter 3, currently a subset of the SARAH [189, 114] family of programs is implemented in `hep-aid`. The `search` module manages PS algorithms, following an AS [190] paradigm in which a search policy and a surrogate model are employed to explore the parameter space of a multi-objective function to find parameter configurations where the objectives satisfy a set of constraints.

This framework allows the integration of potentially any PS method, such as MCMC or ML based sampling methods. The connection between the PS algorithms in the `search` module and the HEP software in the `hep` module is established through the construction of an *objective function*. The `search` module includes an *objective function* constructor, which defines the search space, objectives, and constraints based on a predefined configuration. It also maintains an internal dataset of samples with functionalities for saving, loading, and exporting datasets in formats such as a Pandas DataFrame [191] or a PyTorch [145] tensor.

We demonstrate the use of the library through quick tests and real BSM scan examples, with a focus on the $(B-L)$SSM benchmark phenomenology study case from Chapter 7. This library can effectively replicate those results, as the original code for their reproduction also utilises `hep-aid`.

8.2 The `hep-aid` Library

The `hep-aid` library provides a modular framework for performing parameter scans in BSM scenarios, currently using SPheno [121, 120], HiggsBounds (HB) [129], HiggsSignals (HS)[1] [130] and MadGraph (MG) [124], which we call the *HEP-stack* (see chapter 3, section 3.2). It is focused on the AS [190] paradigm, where a multi-dimensional multi-objective function needs to be defined and the PS algorithm searches for satisfactory configurations in the parameter space given a set of constraints on the objectives. The search is done using a surrogate model, which is fitted to the collected data, to approximate the objective function and to assess which regions of parameter space to explore by querying the HEP-stack. This yields parameter configurations where the objectives satisfy the constraints. We call this region the *satisfactory region*. Originally the library was created to give the user simple access to use the b-CASTOR [175] and Constraint Active Search (CAS) [174] algorithms for PS but, given the modular structure of it, many parameter scan algorithms can be implemented: e.g., `hep-aid` already includes a MCMC method using the MCMC-MH [193, 194], and MLScan [5], a NN based sampling algorithm for BSM phenomenology. The library is divided into two key modules, the `hep` module and the `search` module, illustrated in Figure 8.1.

The `hep` module facilitates the integration of the HEP software ecosystem into Python scripts. The `hep.stack` module integrates SPheno, HB, HS and MG in a sequential stack of software. Technically, this has four HEP-stacks, ranging from SPheno used independently to the full chain incorporating all four HEP programs, and the user can utilise any of these HEP-stacks depending on the phenomenological study. Each HEP-stack includes operational utilities, assisting in handling and managing data, mainly SUSY Les Houches Accord (SLHA) [119] files, and running the software externally. The HEP-stacks need to be initialised with a pre-defined configuration file. This configuration file contains information about the SPheno inputs that will undergo a parameter scan and the necessary directories for the HEP tools used in the scan.

[1]The two programs have recently been incorporated into the HiggsTools distribution [192], which would also be easily embedded in `hep-aid`.

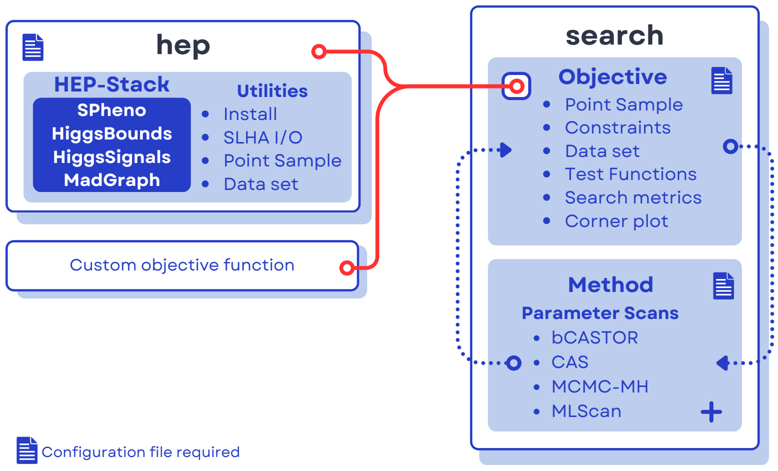
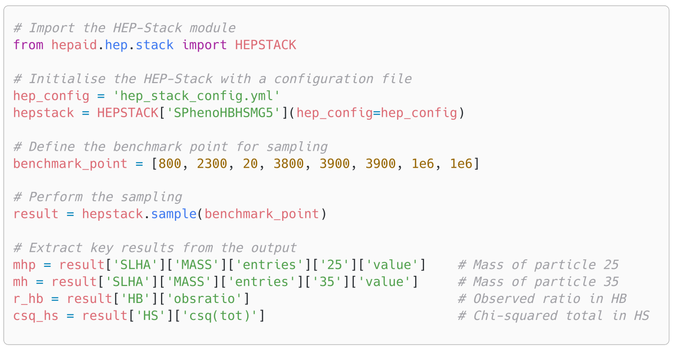


FIGURE 8.1: Diagram showing the general structure of the hep-aid library, which includes two main modules: **hep** and **search**. The search module contains the objective and methods submodules. The **objective** submodule allows users to plug in any custom objective function. The **hep** module serves as a specialised external objective function that integrates with HEP software and can be seamlessly used by the objective submodule for integration.



```
# Import the HEP-Stack module
from hepaid.hep.stack import HEPSTACK

# Initialise the HEP-Stack with a configuration file
hep_config = 'hep_stack_config.yml'
hepstack = HEPSTACK['SPhenoHBHSMG5'](hep_config=hep_config)

# Define the benchmark point for sampling
benchmark_point = [800, 2300, 20, 3800, 3900, 3900, 1e6, 1e6]

# Perform the sampling
result = hepstack.sample(benchmark_point)

# Extract key results from the output
mhp = result['SLHA']['MASS']['entries']['25']['value']    # Mass of particle 25
mh = result['SLHA']['MASS']['entries']['35']['value']     # Mass of particle 35
r_hb = result['HB']['obsratio']                           # Observed ratio in HB
csq_hs = result['HS']['csq(tot)']                         # Chi-squared total in HS
```

FIGURE 8.2: Example code demonstrating the initialisation of the HEP-stack with SPheno, HB, HS, and MG using a HEP-stack class specific to the $(B-L)$SSM. The corresponding **SPhenoHBHSMG5** object runs a parameter configuration defined by an array, returning results in a Python dictionary. Key outputs, such as the Higgs particle masses (**mhp**, **mh**), the HB ratio (**r_hb**), and the HS χ^2 (**csq_hs**), can be extracted for analysis.

The library offers a quick installation method integrated as a command-line feature, along with a function to create a template HEP-stack configuration file. Once the configuration file is defined and the HEP-stack is initiated, the **hep.stack** module enables the user to run a parameter space configuration defined as a simple array. This array corresponds to the input parameters for SPheno, as specified in the configuration file. The result contains all the input and output information of the HEP tools used in the HEP-stack, formatted as a Python dictionary.

The **search** module provides all the necessary components for performing a search on a multi-objective function, either by using the **hep** module or by defining a custom function. It also includes PS algorithms and tools to support experimentation and the development of new PS methods. Since we adhere to the AS approach, hep-aid also includes *surrogate models* to approximate the objective function under study. These features are implemented in three main sub modules, `search.objective`, `search.methods` and `search.models`. The `objective` module contains the `Objective` class, which is the objective function constructor it needs to be initialised with a configuration file stating the search space dimensions with their ranges, the objectives variables and the information of the constraints. The `Objective` class internally manages the sampling of the objective function, stores the dataset, performs data processing to export to the *surrogate models*, and saves the dataset to disk. The `methods` module contains all the PS methods available currently, namely, MCMC-MH [193, 194], b-CASTOR [175], CAS [174] and MLScan [5]. Each method is constructed by inheriting the `Method` base class which loads a specific configuration file and saves or loads checkpoints to continue the search. A set of metrics is recorded in each PS method, such as the total number of parameters sampled, and those that satisfy the constraints. Further metrics may be customised for different use cases. Finally, if the parameter scan algorithms follow an AS approach, they will use the surrogate models implemented in the `search.models` module currently containing Gaussian Processes (GPs) for the AS algorithms and a simple Multi-Layer Perceptron (MLP) for the MLScan method.

The workflow idea that `hep-aid` proposes is that the users define an objective function using the `Objective` class and define its configuration file. They can then run the preferred PS method which uses the initiated `Objective` object. For BSM phenomenological analyses using the `hep` module, the workflow needs to be more elaborated since the objective function needs to be constructed in a appropriate manner compatible with the `Objective` class, by retrieving the necessary values of masses, cross sections, Branching Ratios (BRs), etc.: see the code example in Figure 8.2.

8.2.1 Library Overview

Here, we describe in detail how to make use of `hep-aid`.

8.2.1.1 Installation

The library is publicly available in GitHub[2]. To install `hep-aid` one needs to clone the repository first, then proceed as illustrated in Figure 8.3.

```
git clone https://github.com/mjadiaz/hep-aid.git
pip install.
```

FIGURE 8.3: Commands to clone the repository and install the `hep-aid` package.

[2]https://github.com/mjadiaz/hep-aid.

8.2.1.2 Test Objective Functions

To provide a practical overview of how to use various functionalities of hep-aid, we utilise the test function introduced in chapter 7 equation (7.13) . This is a two-dimensional, double-objective function comprising both uni-modal and multi-modal objectives, with partially overlapping constraints for each objective that defines the satisfactory region. The ground truth satisfactory region is shown in Figure 7.4. The general pipeline for using a parameter scan with hep-aid is illustrated in the example code in Figure 8.4. When the search method is executed in the command line,

```python
from hepaid.search.objective.test import init_him_boo_fn
from hepaid.search.method import bCASTOR

# Initialise the test function
test_objective = init_him_boo_fn()

# Initialise the method algorithm
bcastor = bCASTOR(objective=test_objective)
# Run search
bcastor.run()
```

FIGURE 8.4: Quick start guide illustrating the following: initialising a two-dimensional double objective test function, selecting and starting the b-CASTOR method, then running the search.

a `progress` bar displays key metrics related to the search. In Figure 8.4, which demonstrates the b-CASTOR method, the progress bar will track the success rate, total points sampled, valid points count, and satisfactory points count. It also provides the current iteration and the parameter r, defining the radius of the neighbourhood around each parameter vector. Further details on this algorithm are provided in Section 8.4.2. The dataset generated from the search is stored in `test_objective.dataset`. This dataset can be visualised using a corner plot, a technique that displays pairwise correlations between multiple variables alongside their marginal distributions. In this context, the plot illustrates the various dimensions of the search space and potentially the objective dimensions. The corner plot can be generated using the utility module `hepaid.search.objective.plot`, as demonstrated in Figure 8.5.

Every ML-based active PS method which is used retains a copy of the fitted surrogate model in the `method.method` attribute. This model can be accessed after completing the search and will correspond to the model used in the latest iteration of the search loop. Figure 8.6 demonstrates how to use routines from the `.plot` module, e.g., `generate_meshgrid` and `reshape_model_output`, and to create filled contour plots that visualise the objective approximations generated by the surrogate model fitted to the current dataset. The hep-aid framework currently includes four main PS

```python
from hepaid.search.objective.plot import CornerPlot

# Define 'good' for satisfactory points, 'bad' for the rest
good = test_objective.satisfactory.prod(axis=1).astype(bool)
bad = ~good

# Export dataset as a DataFrame
df = test_objective.as_dataframe()

# Initialise CornerPlot for visualisation
cp = CornerPlot(
    figsize=(4, 4),
    parameters=test_objective.input_parameters,
    colormap='winter_r'
)

# Add traces for satisfactory and unsatisfactory points
cp.add_trace(df[good], r'$\in \mathcal{S}$', alpha=1)
cp.add_trace(df[bad], r'$\notin \mathcal{S}$', alpha=1)

# Update legend and clean up figure
cp.update_legend()
cp.clean()

# Save figure
cp.fig.savefig('plots/bcastor_run_result_himboo.png', dpi=300)
```

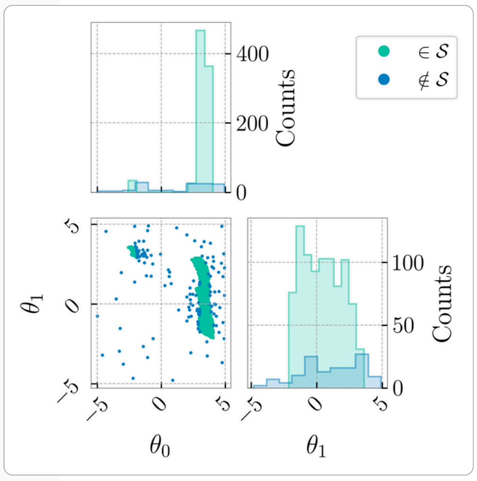


FIGURE 8.5: Code snippet that demonstrates the use of the CornerPlot utility to visualise data points classified as satisfactory or unsatisfactory. The dataset is filtered using the Objective.satisfactory attribute, and traces are added to distinguish between the two classifications.

```python
from hepaid.search.objective.plot import generate_meshgrid
from hepaid.search.objective.plot import reshape_model_output

# Generate the grid
dim_ranges = [(0, 1), (0, 1)]
steps_per_dim = [50, 50]
grid_points, meshgrids = generate_meshgrid(
    dim_ranges, steps_per_dim
    )

# Use the trained surrogate model
mean, sigma = bcastor.model.predict(grid_points)

# Reshape mean with the grid dimensions
reshaped_preds= reshape_model_output(
        model_output=mean[:,i],
        steps_per_dim=steps_per_dim
        ).detach().numpy()
    xx, yy = meshgrids

# Create plot with routines like contourf or imshow
```

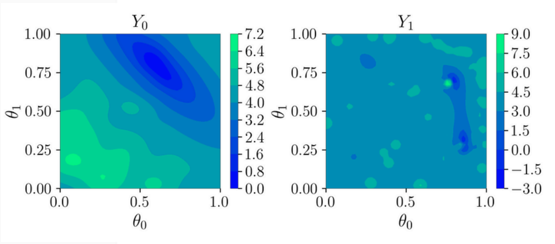


FIGURE 8.6: Example code for the application of the generate_meshgrid and reshape_model_output routines from the hepaid.search.objective.plot module, as well as the evaluation of the GP surrogate model b-CASTOR.model on the generated grid, illustrating the fitted model predictions based on the current dataset.

methods: MCMC-MH, b-CASTOR, CAS, and MLScan. The details of each algorithm are described in Section 8.4.2.

The flexible structure of the hep-aid library allows users to switch between methods seamlessly by following the procedure displayed in Figure 8.4. This flexibility facilitates straightforward performance comparisons between different algorithms, as demonstrated in Figure 8.7. The figure presents the efficiency, measured by the ratio of

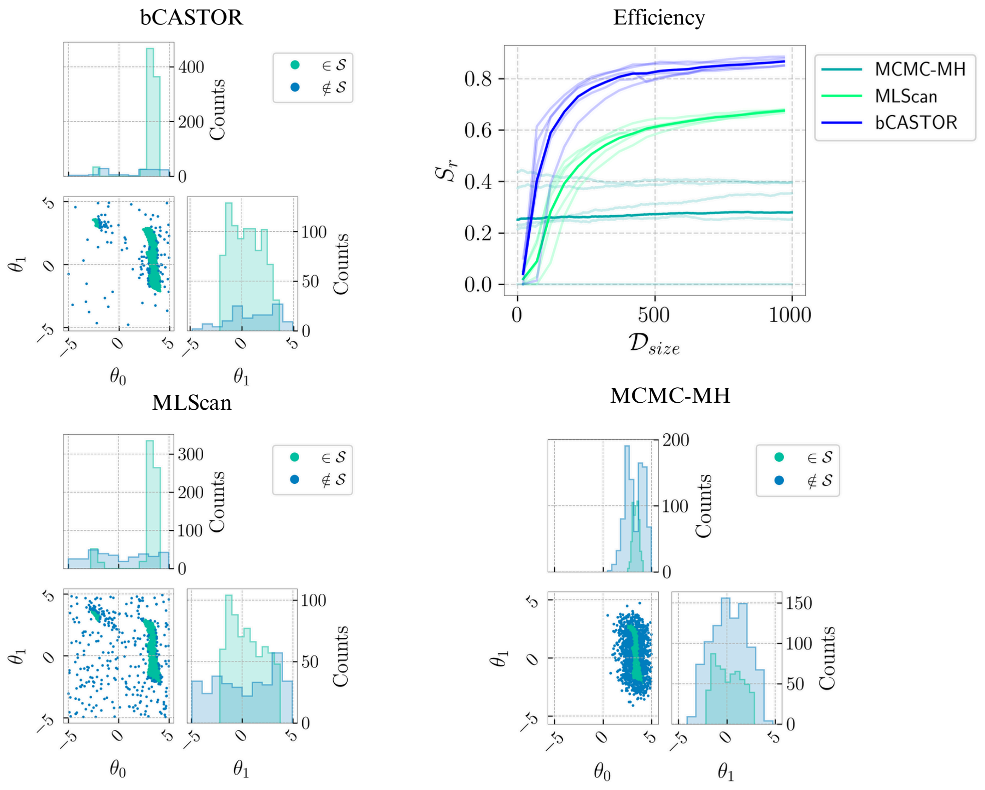


FIGURE 8.7: Performance comparison of b-CASTOR, MLScan, and MCMC-MH for a two-dimensional, double-objective test function. The mean is displayed for five runs for b-CASTOR as well as MLScan, and ten runs for MCMC-MH. The top-right plot displays the efficiency measured as the ratio of satisfactory to total points, $\mathcal{S}_r$, as a function of the dataset size, $\mathcal{D}_{\int\rangle\ddagger\rceil}$. The top-left plot shows the corner plot for b-CASTOR, while the bottom-left and bottom-right plots present the corner plots for MLScan and MCMC-MH, respectively.

satisfactory to total points $\mathcal{S}_r$, as a function of dataset size $\mathcal{D}_{size}$ for b-CASTOR, MLScan, and MCMC-MH on a two-dimensional, double-objective test function. The search for each algorithm is conducted over five independent runs for b-CASTOR and MLScan, and ten runs for MCMC-MH. Figure 8.7 highlights the superior sample efficiency of b-CASTOR compared to the other two methods; however, MLScan exhibits greater exploration within the parameter space, showing robustness for mode discovery. MCMC-MH demonstrates an $\mathcal{S}_r$ of zero in some runs due to suboptimal random starting points, highlighting a key weakness of this algorithm[3]. This comparison illustrates the importance of selecting a PS method based on user needs, as each algorithm has distinct strengths that can be used depending on the requirements of the specific use case.

[3]This analysis can be replicated with the code examples available in github.com/mjadiaz/hepaid_tutorials.

8.2.2 Searching for BSM Physics

To perform parameter searches in BSM phenomenology studies, the HEP-stack software must be installed separately. Therefore hep-aid provides a command-line utility for installing the specific HEP software used in the previous chapter and in [175]. This setup aims to facilitate reproducible phenomenological scans by sharing BSM spectrum files. By running the command `hepaid install-HEP-stack-cli` in the directory containing the SPheno and Universal FeynRules Output (UFO) [195] files, hep-aid will install the required software with the appropriate versions[4]. In this manual, we demonstrate how to conduct a parameter scan within the $(B-L)$SSM to identify Higgs mass values that can explain an experimental signal around 95 GeV in terms of a BSM Higgs boson, alongside the 125 GeV SM-like Higgs state. This signal is supported by multiple experimental analyses searching for new Higgs bosons, including a di-photon ($\gamma\gamma$) excess observed by CMS [11], a di-tau ($\tau^+\tau^-$) excess also reported by CMS [12], and a $b\bar{b}$ excess detected by LEP [13]. The search space in this case is defined by

$$\mathcal{X} = \left\{ x \in \mathbb{R}^8 : x = (M_0, M_{1/2}, \tan\beta, A_0, \mu, \mu', B_\mu, B_{\mu'}) \right\}. \tag{8.1}$$

For demonstration purposes, we simplify the objective space to include only the masses of the two lightest Higgs particles in the $(B-L)$SSM, i.e., the output space is defined by

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^2 : y = (m_{h'}, m_{h^{\text{SM}}}) \right\}. \tag{8.2}$$

Then the task for the search method is to find parameter space points that satisfy the constraints

$$\boldsymbol{\tau} = \begin{cases} m_{h^{\text{SM}}} & = 125 \pm \delta m \text{ GeV}, \\ m_{h'} & = 95 \pm \delta m \text{ GeV}, \end{cases} \tag{8.3}$$

where δm is a user defined mass window. In this case we will consider $\delta m = 5$ GeV. Figure 8.8 illustrates the use of the b-CASTOR algorithm to conduct a search within the $(B-L)$SSM. The objective function must be defined: in this example, we use the `hep_stack_fn` utility function, which enables the use of a HEP-stack object for evaluating a single parameter space point. The result is returned as a nested Python dictionary containing, in this instance, the input and output files for SPheno. To generate the required output from the objective function, we use the `create_simple_dict` utility function, which queries the result dictionary as shown in Figure 8.2 but in an automated manner. This function takes the list of keys from the `masses_spheno_config.yml` configuration file and returns only the specified keys and their values for the input and output parameters defined in the configuration file. The configuration format is designed to be compatible with the `Objective` object constructor by including the `key_chain` and `output_parameters` elements.

[4]The companion repository provides the UFO and SPheno files for the $(B-L)$SSM model, enabling the replication of the example study.

```python
from hepaid.search.objective import Objective
from hepaid.hep.stack import hep_stack_fn
from hepaid.hep.utils import create_simple_dict
from hepaid.search.method import bCASTOR

# Configuration file paths
stack_config_path = 'configs/hep_stack_config.yml'
objective_config_path = 'configs/masses_spheno_config.yml'
method_config_path = 'configs/bcastor.yml'

# Define HEP objective function
def calculate_higgs_masses(x):
    results = hep_stack_fn(x,stack_config_path)
    return create_simple_dict(objective_config_path, results)

# Create the Objective
hep_objective = Objective(
    function=calculate_higgs_masses,
    function_config=objective_config_path
)

# Initialise method and run search
method = bCASTOR(
    objective=hep_objective,
    hyper_parameters=method_config_path
)
method.run()
```

```yaml
# masses_spheno_config.yml
input_space:
  m0:
    lower: 100.
    upper: 1000.
    distribution: "uniform"
    key_chain: ["LHS", "MINPAR", "entries", "1", "value"]
  m12:
    lower: 1000.
    upper: 4500.
    distribution: "uniform"
    key_chain: ["LHS", "MINPAR", "entries", "2", "value"]
  TanBeta:
    lower: 1.
    upper: 60.
    distribution: "uniform"
    key_chain: ["LHS", "MINPAR", "entries", "3", "value"]
  Azero:
    lower: 100.
    upper: 4000.
    distribution: "uniform"
    key_chain: ["LHS", "MINPAR", "entries", "5", "value"]
  MuInput:
    lower: 1000.
    upper: 8000.
    distribution: "uniform"
    key_chain: ["LHS", "EXTPAR", "entries", "11", "value"]
  MuPInput:
    lower: 1000.
    upper: 8000.
    distribution: "uniform"
    key_chain: ["LHS", "EXTPAR", "entries", "12", "value"]
  BMuInput:
    lower: 1e3
    upper: 1e8
    distribution: "uniform"
    key_chain: ["LHS", "EXTPAR", "entries", "13", "value"]
  BMuPInput:
    lower: 1e3
    upper: 1e8
    distribution: "uniform"
    key_chain: ["LHS", "EXTPAR", "entries", "14", "value"]
output_parameters:
  Mh(1): ["SLHA", "MASS", "entries", "25", "value"]
  Mh(2): ["SLHA", "MASS", "entries", "35", "value"]
objectives:
  double_constraint:
    Mh(1): [["gt", 90.4], ["lt", 100.4]]
    Mh(2): [["gt", 120.0], ["lt", 130.4]]
```

FIGURE 8.8: Code example demonstrating the definition of the objective function, **calculate_higgs_masses**, and the initialisation of the Objective constructor. Here, **create_simple_dict** resolves the key chains in the nested **result** dictionary, retaining only the objective keys and their corresponding values. The search configuration file for HEP-stack is also provided, showing the search space configuration, the objectives and their constraints.

The results can be visualised using the **CornerPlot** class of hep-aid, as shown in Figure 8.9[5], which displays the distribution of points within and outside the $\mathcal{S}$ region. In this case, the two classes of points are not clearly separable, as the distributions of satisfactory and non-satisfactory points overlap, as shown in the marginal histograms in the corner plots. However, the number of non-satisfactory points is relatively small, given that b-CASTOR demonstrates nearly 95% efficiency in the top-right plot. The search was conducted with an initial dataset of 400 points. The configuration file for b-CASTOR used in this example is shown in Figure 3.

8.3 HEP Module

The HEP module in hep-aid, located in **hepaid.hep**, provides infrastructure for managing and executing HEP software. As mentioned, for phenomenological analysis, a collection of HEP tools, referred to as a HEP-stack, is typically required to run sequentially for a single parameter space point of the BSM scenario under study. In

[5]Note that the search does not consider $(\chi^2_{HS}, k_{\text{HB}}, \mu_{\gamma\gamma})$ as objectives, as was done in Chapter 7. The $\mathcal{S}$ region expands significantly when only the masses matching the desired values are taken into account.

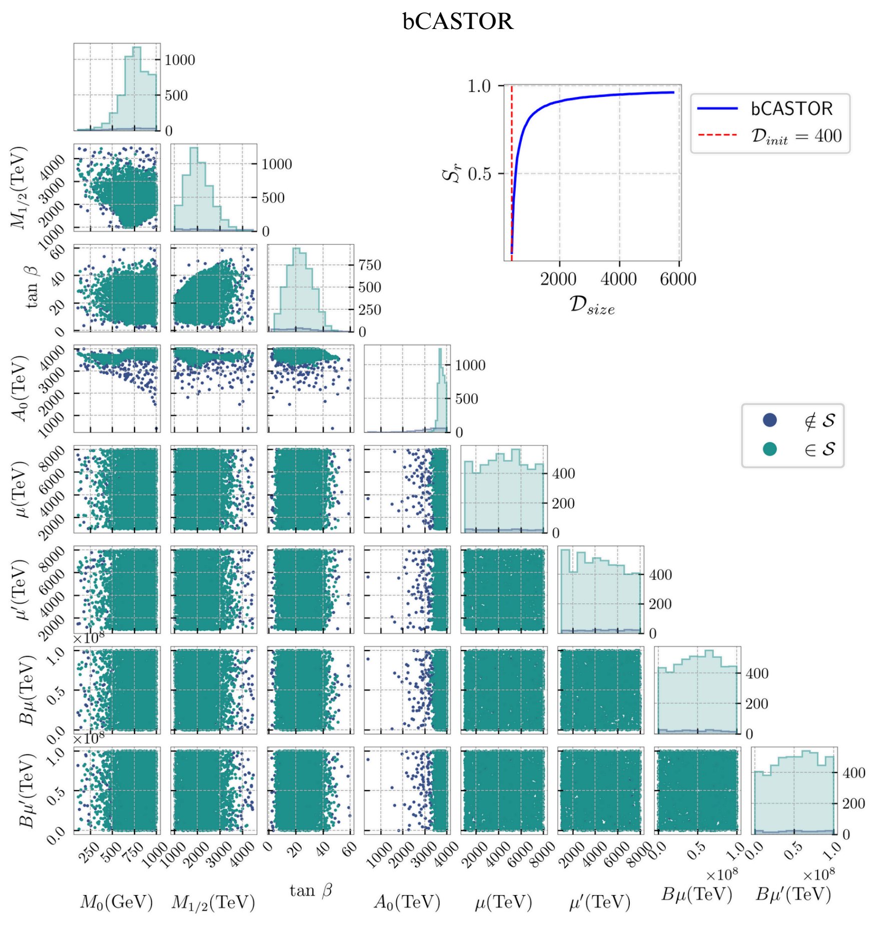


FIGURE 8.9: Corner plot visualisation of point distributions within and outside the $\mathcal{S}$ region. The top-right plot shows the b-CASTOR efficiency, with nearly 95% of points meeting satisfactory constraints. The red dashed line shows the initial number of points for the search.

fact, `hep-aid` aims to simplify the setup and execution of a HEP-stack by abstracting away the details of each tool initialisation and execution. It thus allows the user to perform HEP phenomenology analyses in a simple manner from a Python script or a Jupyter notebook.

8.3.1 HEP-stack

Every HEP-stack is implemented as a Python class and stored in the `HEPSTACK` dictionary. Each HEP-stack needs to be initialised with a pre-defined configuration file. This configuration file contains information about the input parameters that will undergo a parameter scan and the necessary directories for the HEP tools used in the scan. For instance, the HEP-stack composed by the SPheno-HB-HS-MG sequence can be accessed as shown in Figure 8.2. For this HEP-stack, a configuration file is shown in Figure 8.10. With the HEP-stack initialised, we can define a Benchmark Point (BP) within the search space specified in Eq. (7.18). We then execute the HEP-stack using

```yaml
# hep_stack_config.yml
hep_stack:
    name: 'SPhenoHBHSMG5'
    scan_dir: 'path/to/output'
    final_dataset: 'datasets'
    delete_on_exit: True
spheno:
    model: 'BLSSM'
    reference_slha: 'path/to/reference_slha'
    directory: 'path/to/SPheno'
higgsbounds:
    neutral_higgs: 6
    charged_higgs: 1
    directory: 'path/to/hb/build'
higgssignals:
    neutral_higgs: 6
    charged_higgs: 1
    directory: 'path/to/hs/build'
madgraph:
    directory: 'path/to/MG5_aMC'
    scripts:
        gghaa: 'path/to/mg5_script.txt'
```

```yaml
model:
  name: 'BLSSM'
  input:
    m0:
        block_index: 1
        block_name: 'MINPAR'
    m12:
        block_index: 2
        block_name: 'MINPAR'
    TanBeta:
        block_index: 3
        block_name: 'MINPAR'
    Azero:
        block_index: 5
        block_name: 'MINPAR'
    MuInput:
        block_index: 11
        block_name: 'EXTPAR'
    MuPInput:
        block_index: 12
        block_name: 'EXTPAR'
    BMuInput:
        block_index: 13
        block_name: 'EXTPAR'
    BMuPInput:
        block_index: 14
        block_name: 'EXTPAR'
```

FIGURE 8.10: Configuration file for the HEP-stack named `SPhenoHBHSMG5`. The file includes all relevant information for the HEP tools, and on the right, details about the parameters needed to conduct a search using SLHA block information.

the `.sample(x)` method, as shown in Figure 8.2. The result includes all the input and output information from the four HEP tools used in the HEP-stack, presented as a Python dictionary. To obtain the objective parameters within $\mathcal{Y}$, defined in Eq. (7.19), we can simply query the result dictionary using their corresponding chain of keys, as illustrated in Figure 8.2.

8.3.2 HEP Tools

The `hep-aid` package implements the `hepaid.hep.tools` module, which provides classes and methods for managing high-energy physics tools such as SPheno, MG, HB, and HS. This module offers a simplified interface for running computations with minimal boilerplate code. It also handles the automatic management of input and output files, including the creation of necessary directories, and parses tool-specific output formats into structured Python objects for further analysis.

The classes defined within this module corresponds to a specific tool. The structure of each tool class is defined by the `BaseTool` class which includes `run()` and `results()` methods. As an example, Figure 8.11 demonstrates how SPheno can be initialised and run in a straightforward manner. HB and HS are also implemented in `hep-aid`, as

```python
# Import the SPheno tool from the hepaid.hep.tools module
from hepaid.hep.tools import SPheno

# Define paths and model name
heptool_dir = "path/to/SPheno-4.0.4"
output_dir = "path/to/output/files"
model_name = "MODEL"

# Initialise the SPheno tool
spheno = SPheno(heptool_dir, output_dir, model_name)

# Run SPheno with a specified input SLHA file
spheno.run(input_slha_file)

# Retrieve the output SLHA file
output_slha_file = spheno.results
```

```python
# Import the HiggsSignals tool from hepaid.hep.tools
from hepaid.hep.tools import HiggsSignals

# Initialise the HiggsSignals
hs = HiggsSignals(
    heptool_dir='path/to/HS',
    output_dir='path/to/output/files',
    neutral_higgs=6,
    charged_higgs=1
)

# Run HiggsSignals and retrieve results
hs.run()
hs_result = hs.results
```

Figure 8.11: Using SPheno and `HiggsSignals`: The parameter `heptool_dir` corresponds to the path to SPheno's directory, `output_dir` corresponds to the directory where the output will be stored, and `model_name` is the name of the previously compiled model. The process is analogous for `HiggsSignals`, here `heptool_dir` refers to the directory containing the build.

mentioned[6]. The HB and HS tools can be used as shown in Figure 8.11, following the same structure as SPheno. Lastly, the MG HEP tool is implemented to run in script mode. In this case, the `run(mg5_script_path)` method takes the path to the script as input. The `results("path/to/process/output")` method finally takes the path of the MG output generated by the command `output example_process`.

8.3.3 Reading and Writing SLHA Files

The basis for input/ouput in the HEP module in `hep-aid` is the SLHA format [196, 119]. Libraries for manipulating SLHA files exist in literature already. The Pyslha [197] library is a well-established and widely used tool for SLHA file manipulation. Pyslha includes functions for calculating spectrum properties, making it a comprehensive solution for handling SLHA files in various contexts. Additionally, it offers capabilities for generating mass spectrum plots in various formats through the `slhaplot` script. Another notable library is [198], which efficiently manages both individual SLHA files and directories containing multiple files, with a particular emphasis on reading speed optimisation. This library allows for efficient data filtering through Targeted Data Extraction when specific blocks and entries are known. Furthermore, xSLHA (a Python parser for files written in the SLHA format) can process large files where spectra are separated by specific keywords.

However, in `hep-aid` implements its own SLHA, `module.lass`, which treats SLHA files like Python dictionaries, allowing users to access blocks and entries with familiar syntax. This simplifies data extraction and manipulation without the need for a specialised Application Programming Interface (API). The `SLHA` class retains file comments, preserving valuable metadata, explanations, and references essential for

[6]The two programs have recently been incorporated into the HiggsTools distribution [192], which would also be easily embedded in `hep-aid`.

reproducibility and context. It supports the `DECAY1L` and `HiggsTools` blocks, with a focus on SPheno input files, which include a unique `"on/off"` configurations. An example on how the `SLHA` module is used is displayed in Figure 8.12. The main objective of creating such a module is to support the construction of large spectrum datasets, by providing functionalities for exporting SLHA files as nested Python dictionaries, facilitating integration with JavaScript Object Notation (JSON) and enabling the storage of multiple SLHA files within a single zipped JSON file.

```python
# Importing the SLHA reader module from the hep-aid package
from hepaid.hep.read import SLHA

# Load the SLHA file containing particle physics parameters
slha_file = 'LesHouches.in.BLSSM'
slha_data = SLHA(slha_file)

# Print a summary of the SLHA data
print(slha_data)
# Output: SLHA object containing 8 parameter blocks

# Display the available parameter blocks in the SLHA file
parameter_blocks = slha_data.keys()
print(parameter_blocks)
# Output: ['MODSEL', 'SMINPUTS', 'MINPAR', 'EXTPAR',
#          'SPHENOINPUT', 'DECAYOPTIONS', 'YXIN', 'YVIN']

# Access and display the 'MINPAR' block, which contains input
parameters
minpar_block = slha_data['MINPAR']
print(minpar_block)
# Output:
# Block MINPAR   # INPUT PARAMETERS
# 1   6.28483500e+02   # M0 (Universal scalar mass)
# 2   2.45986100e+03   # M12 (Universal gaugino mass)
# 3   4.40308500e+01   # TANBETA (Ratio of Higgs vacuum
expectation values)

# Other blocks can be accessed similarly by their names
```

FIGURE 8.12: Using the SLHA class for the default input SLHA file of the $(B-L)$SSM. Blocks and entries exhibit dictionary-like behaviour and can be exported as a Python dictionary for efficient storage and communication with other modules.

Under the hood, every line representing data in a block is structured with a `BlockLineSLHA` class which stores the numerical entries, values, comments, and line category. Note that entries are defined as everything apart from the value of each line. The blocks are represented with a `BlockSLHA` class, which stores each line, the block header and additional information in this header. Lastly, the `SLHA` extracts automatically the information of an SLHA file and organises it with `BlockSLHA` and `BlockLineSLHA`. This modular design lets users customise the library behaviour or add support for new SLHA blocks, as needed. The `hepaid.hep.read` module also includes utilities for reading results from HB, HS, and single MG process generation, to extract the cross-section and the number of events. These routines are used in the `hepaid.hep.tools` module, where each HEP tool includes its results by calling the corresponding `hepaid.hep.read` routine. As an example, Figure 8.11 shows how

SPheno returns the resulting SLHA file by internally calling the `SLHA` class in `hepaid.hep.read`.

8.4 Search Module

In `hep-aid`, a Parameter Scan Algorithm is implemented as an AS process. AS, introduced in chapter 7 section 7.3, is an iterative search methodology that consists of three main components: the *objective function*, the *search policy*, and the *surrogate model*. At each iteration t, a dataset $\mathcal{D}_t$ is updated with new evaluations from the *objective function*. The *surrogate model* is then made to fit this dataset to refine its approximations and guide the *search policy* in proposing new candidate points within the satisfactory region. Therefore, technically `hep-aid` comprises three core components. In this section, we describe the main components, modules and additional utility functions that `hep-aid` implements to utilise, implement and develop parameter scan methods.

8.4.1 Objective

The interface between an objective function and a parameter scan method is managed by the `Objective` class. This class handles an external multi-objective function, where the objective function is treated as a black-box. By design, the multi-objective function must be defined as a Python function. This function takes the parameter configuration input $\mathbf{x}$ as an array and outputs a dictionary. The dictionary includes keys and values that correspond to the names of the input dimensions and the objectives, along with their respective values. The necessary parameters are defined in the configuration file to initialise the `Objective` class. This is illustrated in Figure 8.13 for a two-dimensional, single-objective Egg Box model function.

The `Objective` class provides a key feature in `hep-aid`: the ability to integrate any black-box function. This allows users to input any function, including external software, as long as the function accepts input $\mathbf{x}$ as an array and outputs a vector $\mathbf{y}$. This capability also serves as the mechanism by which the HEP module is integrated into the `Objective` class.

The `Objective` object stores the function in its `Objective.function` attribute and reads the configuration file to create a search space utility using `scikit-optimize` [199]. This search space facilitates the normalisation of the internal dataset, stored in `Objective.dataset`, which is essential for fitting *surrogate models*. The dataset is dynamically updated whenever the function is called through the `Objective.sample(x)` method, adding the new set $\mathbf{x}, \mathbf{y}$ to the internal dataset. The historical `Objective.dataset` is formatted as a Python dictionary. However, the

```python
from hepaid.search.objective.objective import Objective
import numpy as np

# Define a two-dimensional single objective function
def egg_box(x):
    f = (2 + np.cos(x[0] / 2) * np.cos(x[1] / 2)) ** 5
    return {'x1': x[0], 'x2': x[1], 'f': f}

# Initialise the Objective instance
test_objective = Objective(
            function=egg_box,
            function_config='egg_box_config.yml',
            cas=False
)
```

```yaml
# egg_box_config.yml
input_space:
  x1:
    lower: 0
    upper: 31.41
    distribution: uniform
  x2:
    lower: 0
    upper: 31.41
    distribution: uniform
output_parameters:  ['f']
objectives:
  double_constraint:
    f: [['gt', 80],['lt',160]]
```

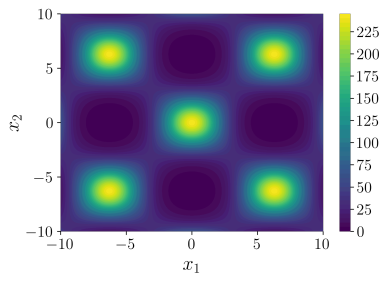


FIGURE 8.13: Definition of the Egg Box model as a function, which serves as the objective function to initialise the Objective class. The function accepts an array of input parameters and outputs a Python dictionary containing all the parameters, both input and output. These parameters and their ranges are defined in a configuration file, along with the objective dimensions and their constraints. Additionally, the contour plot of the egg_box function is displayed at the bottom.

Objective.x, Objective.X, and Objective.Y attributes hold the input data in the original space, the input data in the normalised space, and the output data in the original space, respectively. The method Objective.as_dataframe(satisfactory=True) return the dataset as a Pandas DataFrame [191]. The Objective.satisfactory attribute generates a boolean matrix, S_{ij}, with the same dimensions as Objective.Y. This matrix identifies which elements in Objective.Y satisfy the constraints defined in the configuration file. Consequently, taking the vertical product, $S_j = \prod_i S_{ij}$, of Objective.satisfactory produces a boolean array. This array labels as True the configurations in Objective.dataset that meet the specified constraints.

Lastly, the method Objective.save(path) saves the internal dataset as a compressed JSON file. Then, the method Objective.load(path, process=True) loads a previously saved JSON file, updating the internal dataset and all relevant information if the process argument is set to True. This allows the continuation of the search process with a parameter scan algorithm or for plotting purposes.

8.4.2 Parameter Space Sampling

The hepaid.search.methods module includes all the implemented PS algorithms. Currently, these methods are MCMC-MH, b-CASTOR, CAS, and MLScan. Due to the varying requirements of each parameter scan algorithm, hep-aid implements a base class called `Method`. This class is designed to be inherited by PS algorithms and encapsulates the essential functionalities needed for managing hyperparameters, tracking standard and custom metrics, handling checkpoints, and managing file directories. Every method that inherits from this base class will receive an `Objective` instance and a hyperparameter file as arguments. The file can be provided by the user as a string path or a `DictConfig` [200]. If neither is provided, the default configuration file will be used. The `Method` class also offers utility methods to save and load the state of the search process, including the state of the objective function, current metrics being tracked, and the hyperparameter configuration file. Each instance of the `Method` class initialises a `Metrics` object responsible for tracking standard metrics related to the search process. These metrics include the total number of points evaluated, the number of valid points, the number of satisfactory points, the success rate, and the current iteration counter. Additionally, users can extend the metrics functionality by defining custom metrics that can be updated during the search process.

There is a defined general structure for the hyperparameter configuration file, as illustrated in Figure 8.14. Each instance of the `Method` class will read the following hyperparameters. The `run_name` refers to the path to the directory where checkpoints and configuration files will be saved. The `parallel` parameter is defined as a boolean parameter to enable the parallel evaluation of specific processes during the search. This works in conjunction with the `n_workers` parameter, which dictates the number of parallel processes based on available resources. For example, this setup is used for evaluating the objective function of the batch $\boldsymbol{X}^*$ suggested by the policy, by using the function `batch_evaluation` located in `hepaid.search.objective.utils`. Except for MCMC-MH, search methods require an initial dataset, which is configured via the `initial_dataset` hyperparameter. The `initial_dataset.n_points` specifies the number of points to be generated, while `initial_dataset.generate` is a boolean indicator for the dataset generation. This is useful, for instance, when the user wishes to continue from previous checkpoints where the complete initial dataset is not needed. The initial dataset is generated by the function `generate_initial_dataset`. The `total_iteration` parameter sets the total number of iterations for the search loop. Additional parameters will depend on the specific algorithm and are explained below.

Hyperparameter	**Description**
`run_name`	Directory path for saving checkpoints and configuration files.
`parallel`	Boolean parameter to enable parallel evaluation of processes.
`n_workers`	Number of parallel processes based on available resources.
`initial_dataset`	Configures initial dataset required by most search methods.
`initial_dataset.n_points`	Number of points to generate in the initial dataset.
`initial_dataset.generate`	Specifies whether the initial dataset should be generated.
`total_iteration`	Sets the total number of iterations for the search loop.

TABLE 8.1: Hyperparameters for a PS method implemented by inheriting the `Method` class.

8.4.2.1 AS Methods

As mentioned, hep-aid implements two AS methods, CAS and the b-CASTOR algorithm introduced in chapter 7. Therefore, the additional hyperparameters relevant for CAS are located in the configuration file as `eci.num_samples`, which specifies the number of sample points for which an individual hypersphere is approximated, and the `resolution` block. In this context, `resolution.constant_resolution` allows for switching between a fixed radius or a linear decay in the radius. The `resolution.value` sets the value, typically ranging between 0 and 1, as the search space is normalised. The parameters `resolution.initial` and `resolution.final` are used when the linear decay mode is enabled. Additionally, `resolution.r_decay_steps` specifies the number of iteration steps during which the decay takes place.

Hyperparameter	**Description**
`eci.num_samples`	Number of sample points for approximating each hypersphere.
`resolution.constant_resolution`	Switches between fixed radius and linear radius decay.
`resolution.value`	Sets radius value, typically between 0 and 1 (normalised space).
`resolution.initial`	Initial radius value for linear decay.
`resolution.final`	Final radius value for linear decay.
`resolution.r_decay_steps`	Number of steps for radius decay.

Table 8.2: Hyperparameters used for CAS in hep-aid.

Then, the additional hyperparameters required by b-CASTOR include those from CAS related to the resolution parameter, as well as the following ones. For the batch sampling settings, we have `batch_sampling.tpe_trials`, which indicates the number of trials for optimising the ECI function using the TPE algorithm. Next, `batch_sampling.rank_samples` specifies the number of samples obtained from the historical data of the TPE optimisation using the stochastic prioritisation technique. This process determines the size of the batch $\boldsymbol{X}^*$. The `batch_sampling.alpha` parameter controls the degree of prioritisation in rank-based sampling; a higher value prioritises sample efficiency. The interaction between the `alpha` and `rank_samples` hyper-parameters determines the balance between exploration and exploitation in the algorithm. Empirically, setting `alpha=2` provides a good balance of efficiency and diversity within the satisfactory region. A higher number of `rank_samples`, i.e., a larger batch size, allows for more chances for exploration.

Hyperparameter	**Description**
`batch_sampling.tpe_trials`	Number of TPE trials for optimising the ECI function.
`batch_sampling.rank_samples`	Number of samples obtained through rank based sampling (batch $\boldsymbol{X}^*$ size).
`batch_sampling.alpha`	Prioritisation degree in rank-based sampling.

Table 8.3: Additional hyperparameters (with respect to the CAS ones) relevant to the b-CASTOR algorithm.

Figure 8.14 shows the complete configuration file for the b-CASTOR algorithm, which can also be used for the CAS algorithm. The file is displayed in `YAML` format.

```yaml
# configs/bcastor.yml
run_name: "blssm/run"
parallel: True
initial_dataset:
  generate: True,
  n_workers: 30
  n_points: 400
checkpoint:
  name: "checkpoint"
  n_step_save: 5
total_iterations: 250
```

```yaml
resolution:
  value: 1e-6
  constant_resolution: False
  r_decay_steps: 100
  initial: 1e-2
  final: 1e-6
batch_sampling:
  tpe_trials: 2500
  rank_samples: 30
  n_evaluation_workers: 30
  alpha: 2
eci:
  num_samples: 500
```

FIGURE 8.14: Configuration file for the b-CASTOR algorithm. The left side displays the general structure of a configuration file for a parameter scan implemented in `hep-aid`. The right side displays the parameters relevant to the CAS algorithm in the resolution block, where additional batch sampling blocks are required for the b-CASTOR algorithm.

The technical implementation for both the CAS and b-CASTOR algorithms is as follows. We used the ECI acquisition function implementation provided by the BoTorch library [201]. Additionally, we integrated GP models, available in `hepaid.search.models`, from the GPytorch library [146], a repository for scalable GP inference built on PyTorch [145]. We further utilise the TPE implementation available in Optuna [202], an open-source hyperparameter optimisation framework.

8.4.2.2 MCMC-MH

The MCMC-MH algorithm is introduced in chapter 5.1. The hyperparameters defined in the configuration file include `burn_in`, which defines the number of initial iterations to discard, allowing the chain to converge to the target distribution and helping to eliminate dependence on the starting value. The `initial_scale` parameter sets the initial step size for generating proposal states from the current state. The `adapt_frequency` parameter defines how often to adjust the scale of the proposal distribution during the sampling process, based on the observed acceptance rate. Lastly, the `target_acceptance_rate` specifies the desired acceptance rate for proposal moves, guiding the adaptation of the proposal distribution scale and influencing the amount of exploration of the parameter space while ensuring efficient convergence to the target distribution.

Hyperparameter	**Description**
`burn_in`	Number of initial iterations for burn-in.
`initial_scale`	Initial step size for generating proposal states.
`adapt_frequency`	Frequency of adjusting the scale on burn-in.
`target_acceptance_rate`	Desired acceptance rate for proposals.

TABLE 8.4: Hyperparameters relevant for the MCMC-MH algorithm.

For a PS, the target distribution is the likelihood constructed from the constraints in the objectives. In hep-aid, this likelihood can be provided as a function. If not, hep-aid will use a default likelihood constructed with sigmoid windows for each objective and its constraints as defined in chapter 7 in equations 7.10, 7.5.2 and 7.12.

8.4.2.3 MLScan

The Machine Learning Scan method [5], termed MLScan in hep-aid was designed for efficient sampling and exploration of parameter spaces using a Machine Learning surrogate model for the observables, fitting in the description of AS. It uses an MLP Neural Network as a *surrogate model.* The search policy in this case is performing Rejection Sampling (see section 5.1.1) over the likelihood, but this time the likelihood is evaluated using the predictions from the MLP model.

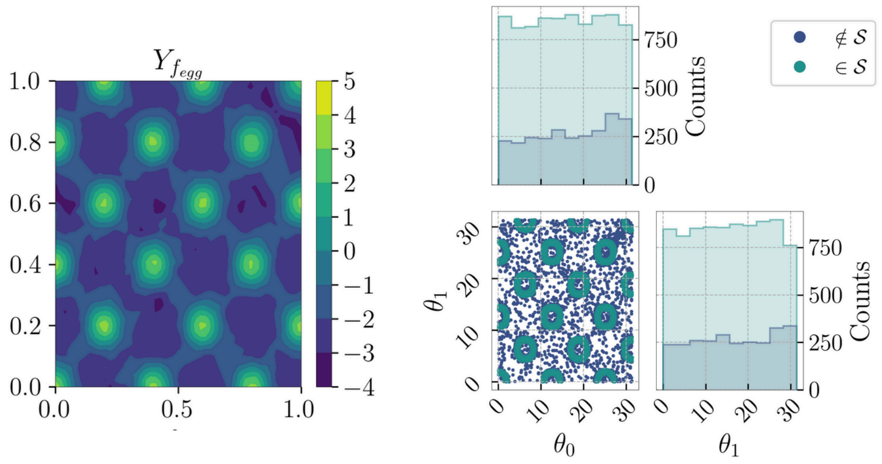


FIGURE 8.15: On the left, the predictions across the entire parameter space of the surrogate MLP in the final iteration of the search from Figure 8.16 are shown. On the right, the points identified by the MLScan method are displayed using the corner plot functionalities of hep-aid.

The implementation in hep-aid begins with an initial dataset, similar to previous methods. In each iteration, the MLP is trained using the currently available dataset, retaining the model parameters from previous iterations to perform incremental learning. The rejection sampling technique is used to generate the batch $\boldsymbol{X}^*$, with the size of this batch controlled by the hyperparameter `num_samples`. The scaling of the acceptance probability for the rejection sampling method is controlled by the parameter `m_factor`. By adjusting this factor, users can manage the balance between sample quality and the diversity of generated samples, thus affecting both the efficiency and effectiveness of the sampling process. The MLScan method also adds extra random samples to encourage exploration, with the number of samples controlled by the hyperparameter `extra_random_samples`.

```python
import torch
from hepaid.search.objective.test import init_egg_box_fn
from hepaid.search.method.mlscan import MLScan
from hepaid.search.method.eci import smooth_box_mask

# Define likelihood
def likelihood(x):
    if not isinstance(x, torch.Tensor):
        x = torch.tensor(x)
    f = x[:,1]
    lh = smooth_box_mask(f, 80., 160., 1e-3 )
    return  lh

# Initialise Eggs model
test_objective = init_egg_box_fn()

# Define the configuration file
hp = 'datasets/mlscan/run/eggs/hprms.yaml'

# Initialise method and run
mlscan= MLScan(
    objective=test_objective,
    likelihood=likelihood,
    hyper_parameters=hp
    )
mlscan.run()
```

```yaml
# hprms.yaml
num_samples: 100
checkpoint:
  n_step_save: 500
  name: chckpnt
run_name: mlscan/run
total_iterations: 100
parallel: false
n_workers: 4
initial_dataset:
  generate: True,
  n_points: 200
model:
  layer_sizes: [2, 60,60,60,1]
  dropout_prob: 0.0
  learning_rate: 0.01
  num_epochs: 500
  step_size: 5
  gamma: 0.999
  threshold: 1e-8
```

FIGURE 8.16: A code example demonstrating the utilisation of the MLScan method, a neural network-based sampling method introduced in [5], is provided. Users must define a likelihood function. Since the Egg Box model is already implemented in hep-aid, it can be readily used. Initialising the MLScan method requires a configuration file, as shown in the code snippet on the right.

For MLScan an additional hyper-parameter block, named `model_hyperparameters`, is required for configuring the MLP's architecture and training. The `layer_sizes` parameter receives a list specifying an input layer, an arbitrary number of hidden layers with their respective number of neurons, and the output layer neurons. The `dropout_prob` indicates the level of dropout regularisation in each layer. The `learning_rate` controls the magnitude of weight updates during training, while `num_epochs` defines the number of complete passes through the training dataset. Additionally, the `step_size` determines how frequently the learning rate is adjusted, and `gamma` specifies the decay rate for the learning rate, allowing it to decrease gradually. Finally, the `threshold` acts as a stopping criterion for training when the loss function fall below this value.

Finally, the `MLScan` class takes an argument for the likelihood function, which the user must define, as shown in Figure 8.16. In this example, `hep-aid` can be used to replicate the results from [5] using the Egg Box model test function, which is currently implemented in `hep-aid`. The results of running the MLScan search on the Egg Box model are displayed in Figure 8.15.

Hyperparameter	Description
`num_samples`	Batch size for rejection sampling.
`m_factor`	Scales acceptance probability in rejection sampling.
`extra_random_samples`	Number of additional random samples for exploration.
`model_hyperparameters`	Configures MLP architecture and training.
`layer_sizes`	Defines neuron counts for MLP layers.
`dropout_prob`	Dropout rate for regularisation.
`learning_rate`	Step size for weight updates.
`num_epochs`	Total training dataset passes.
`step_size`	Interval for adjusting learning rate.
`gamma`	Learning rate decay factor.
`threshold`	Loss threshold for stopping criterion.

TABLE 8.5: Hyperparameters relevant for the MLScan algorithm.

8.5 Conclusions

This chapter introduced `hep-aid`, a new Python library designed to facilitate PS algorithms for BSM phenomenology. The library provides a modular framework that integrates HEP software, simplifying the implementation and development of PS algorithms while offering essential functionalities for phenomenological analysis. Originally the library was created to give researchers simple access to use the b-CASTOR and CAS algorithms for parameter scans. However, its development lead to a modular structure allowing the implementation of further PS algorithms existing in the literature such as MLScan.

We demonstrated the utility of the library by performing multi-objective searches on test functions and comparing the performance of different PS algorithms. In this connection, b-CASTOR exhibited superior sample efficiency while achieving a comprehensive characterisation of the satisfactory parameter region. The experiments with the MLScan algorithm instead highlighted its robust exploration capabilities, due to the stochastic nature of its policy. However, for BSM phenomenological studies, a NN classifier needs to be implemented to enhance the sample efficiency of this algorithm. (All ML-based such approaches were also demonstrated to be superior to the more standard MCMC-HS one.) Additionally, we illustrated the application of the library in a real BSM case study, fitting the masses of the lightest Higgs particles in the $(B-L)$SSM to explain new physics signals in $\gamma\gamma$, $\tau^+\tau^-$, and/or $b\bar{b}$ final states [175].

Chapter 9

Summary, Conclusions and Reflections

Chapter 1 provided an overview of the theoretical framework established through decades of research in particle physics, culminating in the SM and the theoretical and observational challenges it faces. It explored how the SM can be extended to account for signals of new physics and address theoretical challenges, highlighting the vast model space that has emerged over the years. SUSY was introduced, with a particular focus on the $(B-L)$SSM model—a non-minimal realisation of SUSY—used as a benchmark for the development of parameter scan algorithms.

Chapter 2 introduced the computational challenges associated with BSM phenomenology, particularly the parameter space exploration problem, which is the primary research question addressed in this thesis.

Chapters 3 and 4 presented the necessary mathematical background for developing machine learning-based parameter scan algorithms. GPs and MLPs were introduced, along with methods for training them. The main sampling algorithms were reviewed, and decision-making was explained through two perspectives: optimisation and reinforcement learning.

In Chapter 7, the active search formulation for parameter scan methods is introduced, alongside the b-CASTOR algorithm—a novel multi-point, multi-objective active search method specifically designed for computationally expensive BSM scenarios. This method efficiently identifies the satisfactory region ($\mathcal{S}$) within the parameter space by utilising probabilistic surrogate models and the ECI acquisition function, which maximises the covered volume of $\mathcal{S}$ while maintaining high sample diversity and efficiency.

We demonstrated the effectiveness of b-CASTOR through two case studies: a double-objective 2D test function and a $(B-L)$SSM scenario. In the latter, the algorithm was applied to investigate three anomalies at approximately 95 GeV in the $\gamma\gamma$, $\tau\tau$, and $b\bar{b}$ channels. While b-CASTOR successfully identified solutions explaining the longest-established $\gamma\gamma$ anomaly, it was unable to concurrently satisfy all three channels, though it identified points meeting signal strengths for $\mu_{\gamma\gamma}$ and μ_{bb}.

The method is robust to the choice of the resolution parameter r, with gradual reductions enabling finer exploration of unexplored regions. Our experiments highlight the trade-offs between exploration and exploitation in b-CASTOR, influenced by ECI optimisation trials, sample prioritisation, and the number of Rank-based samples. Future advancements could focus on incorporating scalable probabilistic surrogate models and the development of performance metrics.

In chapter 7 we specifically compare our approach with MCMC-MH, as the latter is well-established in the community. Nevertheless, it is important to note that this comparison might not be entirely fair [39]. MCMC-MH operates under assumptions that we are testing *against*, such as requiring a long enough Markov Chain so that the MH algorithm samples the full posterior probability density function across the entire parameter space. This comparison serves as a proof-of-concept that a *search* algorithm could better suit our objectives of comprehensively characterising the $\mathcal{S}$ region in a sample-efficient manner. Exploring b-CASTOR performance against emerging approaches from the ML and AI community within HEP [186, 203, 46, 45] could yield valuable insights into potential enhancements and application scenarios. The first direct comparison of this kind was made on [204] and described in chapter 8 that introduced the `hep-aid` Python library.

The `hep-aid` library, presented in Chapter 8, is the result of extensive development starting from the initial experiments in this thesis. It has proven to be a valuable tool for PS development, particularly from an Active Search and ML perspective. Its key strengths are its modularity, support for PS algorithm development, and ease of use for implemented PS methods. The HEP module includes the SARAH family and original tools for SLHA file manipulation.

Future work on `hep-aid` will aim to improve its design to support the creation of new PS algorithms, including combining existing components. For example, using the rejection sampling feature from MLScan to sample from the ECI policy function, applied in CAS and b-CASTOR, could improve exploration capabilities—a direction already under investigation. Another area for development is integrating `hep-aid` with other PS libraries, especially those linked to a wide range of HEP software. The main goal of `hep-aid` remains to provide a versatile platform for developing and benchmarking PS algorithms, with a focus on ML-based approaches.

As highlighted in the introduction, the quest for a definitive BSM model requires navigating both the *model space* and the *parameter space* of each candidate. The development of the b-CASTOR PS algorithm and the `hep-aid` Python library represents the core contributions of this thesis. These tools are designed to push the boundaries of parameter space exploration, achieving greater sample efficiency and precision. By employing surrogate models, such as GPs or MLPs trained on parameter configuration datasets that thoroughly capture the satisfactory regions of a BSM, it becomes possible to evaluate critical metrics like the fine-tuning of a model or the integral of a likelihood over observables. These metrics can then feed into external discrete optimisation algorithms that explore the *model space*, enabling the discovery of not just optimal configurations, but the most compelling BSM scenarios. Workflows operating on the model space have been proposed in the literature [205, 171, 172]. However, they often avoid parameter space scanning due to its *high computational cost*–a limitation directly addressed by the methods presented in this thesis.

The PS methodology proposed in this thesis was applied to the $(B-L)$SSM model, a BSM scenario already known to offer solutions to the new physics signals under consideration. This work lays the foundation for future advancements in several key directions. First, efficiently identifying and sampling the complete satisfactory region of the parameter space for any BSM model, even when feasible benchmark points are unknown. The sample-efficient PS algorithms developed can be readily utilised through `hep-aid`, providing a practical and accessible tool for phenomenological testing. Second, incorporating a larger number of constraints to explore and test the high dimensionality in both search and objective spaces. Pushing the limits of the b-CASTOR algorithm could offer insights into more scalable PS algorithms, whether by employing advanced surrogate models or developing novel policies. Third, integrating kinematic observable information, whether through raw momentum data or predefined kinematic variables. In this work, we utilised the mass values from scalar searches, but emerging ML methodologies could enable the direct use of momentum data.

Appendix A

Bayes' theorem

Consider a dataset $\mathcal{D}$. The probability distribution for the parameters $\boldsymbol{\theta}$ of a model M, given $\mathcal{D}$, can be estimated using the so-called *Bayes' Theorem* [41],

$$P\left(\boldsymbol{\theta} \mid \mathcal{D}, M\right) = \frac{P\left(\mathcal{D} \mid \boldsymbol{\theta}, M\right) P\left(\boldsymbol{\theta} \mid M\right)}{P(\mathcal{D} \mid M)} \equiv \frac{\mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\mathcal{Z}} \tag{A.1}$$

Every component in this formula is crucial to understanding how Bayes' theorem provides a framework for updating our beliefs about model parameters given observed data. The term $P(\boldsymbol{\theta} \mid M)$, denoted as $\pi(\boldsymbol{\theta})$, is the prior. It encodes our prior assumptions or knowledge about the parameters $\boldsymbol{\theta}$ before observing the data. The term $P(\mathcal{D} \mid \boldsymbol{\theta}, M)$ is the likelihood, which represents the probability of the observed data $\mathcal{D}$ given the parameters θ and the model M. Importantly, the likelihood is interpreted as a function of the parameters:

$$\mathcal{L}(\boldsymbol{\theta}) \equiv \mathcal{L}(\boldsymbol{\theta}; D, M) \tag{A.2}$$

In this interpretation, the likelihood measures a *relative score* for different parameter values based on how well they explain the observed data. Unlike a probability distribution, the likelihood is not normalised, as shown by:

$$\int \mathcal{L}(\boldsymbol{\theta}) d\boldsymbol{\theta} \neq 1$$

The denominator, $P(\mathcal{D} \mid M)$, is known as the evidence or marginal likelihood of the model. It is commonly denoted as $\mathcal{Z} \equiv P(\mathcal{D} \mid M)$ and is computed by integrating out the model parameters:

$$\mathcal{Z} = \int \mathcal{L}(\boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{A.3}$$

Finally, the posterior $P(\boldsymbol{\theta} \mid \mathcal{D}, M)$ represents the probability distribution of the parameters $\boldsymbol{\theta}$ given the observed data $\mathcal{D}$ and the model M. It reflects our updated belief about the parameters after taking the data into account.

The posterior is often challenging to compute directly due to the intractability of high-dimensional integrals coming from the evidence, the likelihood may lack an analytical form and the parameter space might be vast, with significant contributions from the likelihood and prior confined to small regions.

Bibliography

[1] Kurt Riesselmann. The standard model of particle physics. https://www.symmetrymagazine.org/article/july-2015/standard-model, July 2015. Accessed: 2024-12-02.

[2] CMS Collaboration. A portrait of the higgs boson by the cms experiment ten years after the discovery. *Nature*, 607:60–68, 2022. URL https://arxiv.org/abs/2207.00043.

[3] G. Aad et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012. ISSN 0370-2693. URL https://www.sciencedirect.com/science/article/pii/S037026931200857X.

[4] David N. Spergel. The dark side of cosmology: Dark matter and dark energy. *Science*, 347(6226):1100–1102, 2015. URL https://www.science.org/doi/10.1126/science.aaa0980.

[5] Jie Ren, Lei Wu, Jin Min Yang, and Jun Zhao. Exploring supersymmetry with machine learning. *Nuclear Physics B*, 943:114613, 2019. ISSN 0550-3213. URL https://www.sciencedirect.com/science/article/pii/S0550321319300938.

[6] Stephen P. Martin. *A Supersymmetry Primer*, pages 1–98. World Scientific, July 1998. URL http://dx.doi.org/10.1142/9789812839657_0001.

[7] Florian Staub. Exploring new models in all detail with¡tt¿sarah¡/tt¿. *Advances in High Energy Physics*, 2015:1–126, 2015. ISSN 1687-7365. URL http://dx.doi.org/10.1155/2015/840780.

[8] S. Chatrchyan et al. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012. ISSN 0370-2693. URL https://www.sciencedirect.com/science/article/pii/S0370269312008581.

[9] Oliver Fischer, Bruce Mellado, Stefan Antusch, Emanuele Bagnaschi, Shankha Banerjee, Geoff Beck, Benedetta Belfatto, Matthew Bellis, Zurab Berezhiani, Monika Blanke, Bernat Capdevila, Kingman Cheung, Andreas Crivellin, Nishita

Desai, Bhupal Dev, Rohini Godbole, Tao Han, Philip Harris, Martin Hoferichter, Matthew Kirk, Suchita Kulkarni, Clemens Lange, Kati Lassila-Perini, Zhen Liu, Farvah Mahmoudi, Claudio Andrea Manzari, David Marzocca, Biswarup Mukhopadhyaya, Antonio Pich, Xifeng Ruan, Luc Schnell, Jesse Thaler, and Susanne Westhoff. Unveiling hidden physics at the LHC. *The European Physical Journal C*, 82(8), aug 2022. URL `https://doi.org/10.1140%2Fepjc%2Fs10052-022-10541-4`.

[10] P. S. Bhupal Dev, Amarjit Soni, and Fang Xu. Hints of natural supersymmetry in flavor anomalies? *Physical Review D*, 106(1), jul 2022. URL `https://doi.org/10.1103%2Fphysrevd.106.015014`.

[11] Search for a standard model-like Higgs boson in the mass range between 70 and 110 GeV in the diphoton final state in proton-proton collisions at $\sqrt{s} = 13$ TeV. 2023.

[12] Searches for additional Higgs bosons and vector leptoquarks in $\tau\tau$ final states in proton-proton collisions at $\sqrt{s} = 13$ TeV. Technical report, CERN, Geneva, 2022. URL `https://cds.cern.ch/record/2803739`.

[13] R. Barate et al. Search for the standard model Higgs boson at LEP. *Phys. Lett. B*, 565:61–75, 2003. .

[14] Alexander Belyaev, Rachid Benbrik, Mohammed Boukidi, Manimala Chakraborti, Stefano Moretti, and Souad Semlali. Explanation of the hints for a 95 gev higgs boson within a 2-higgs doublet model. 2023.

[15] Duarte Azevedo, Thomas Biekötter, and P. M. Ferreira. 2hdm interpretations of the cms diphoton excess at 95 gev. 2023.

[16] Pablo Escribano, Victor Martin Lozano, and Avelino Vicente. A scotogenic explanation for the 95 gev excesses. 2023.

[17] Saiyad Ashanujjaman, Sumit Banik, Guglielmo Coloretti, Andreas Crivellin, Bruce Mellado, and Anza-Tshilidzi Mulaudzi. $su(2)_L$ triplet scalar as the origin of the 95 gev excess? *Phys. Rev. D*, 108:L091704, Nov 2023. URL `https://link.aps.org/doi/10.1103/PhysRevD.108.L091704`.

[18] P. S. Bhupal Dev, Rabindra N. Mohapatra, and Yongchao Zhang. Explanation of the 95 GeV $\gamma\gamma$ and $b\bar{b}$ excesses in the Minimal Left-Right Symmetric Model. *Physics Letters B*, 849:138481, February 2024. ISSN 03702693. URL `http://arxiv.org/abs/2312.17733`. arXiv:2312.17733 [hep-ex, physics:hep-ph].

[19] Debasish Borah, Satyabrata Mahapatra, Partha Kumar Paul, and Narendra Sahu. Scotogenic $u(1)_{L_\mu - L_\tau}$ origin of $(g-2)_\mu$, $\mathbf{W}$-mass anomaly and 95GeV excess. February 2024. URL `http://arxiv.org/abs/2310.11953`. arXiv:2310.11953 [hep-ph].

[20] Srimoy Bhattacharya, Guglielmo Coloretti, Andreas Crivellin, Salah-Eddine Dahbi, Yaquan Fang, Mukesh Kumar, and Bruce Mellado. Growing Excesses of New Scalars at the Electroweak Scale. 6 2023.

[21] Junjie Cao, Xinglong Jia, and Jingwei Lian. Unified Interpretation of Muon g-2 anomaly, 95 GeV Diphoton, and $b\bar{b}$ Excesses in the General Next-to-Minimal Supersymmetric Standard Model. 2 2024.

[22] Junjie Cao, Xinglong Jia, Yuanfang Yue, Haijing Zhou, and Pengxuan Zhu. 96 GeV diphoton excess in seesaw extensions of the natural NMSSM. *Phys. Rev. D*, 101(5):055008, 2020. .

[23] Weichao Li, Haoxue Qiao, Kun Wang, and Jingya Zhu. Light dark matter confronted with the 95 GeV diphoton excess. 12 2023.

[24] Ting-Kuo Chen, Cheng-Wei Chiang, Sven Heinemeyer, and Georg Weiglein. 95 GeV Higgs boson in the Georgi-Machacek model. *Phys. Rev. D*, 109(7): 075043, 2024.

[25] Amine Ahriche. The 95 GeV Excess in the Georgi-Machacek Model: Single or Twin Peak Resonance. 12 2023.

[26] Giorgio Arcadi, Giorgio Busoni, David Cabo-Almeida, and Navneet Krishnan. Is there a (pseudo)scalar at 95 gev? 2023.

[27] Amine Ahriche, Mohamed Lamine Bellilet, Mohammed Omer Khojali, Mukesh Kumar, and Anza-Tshildzi Mulaudzi. The scale invariant scotogenic model: Cdf-ii w-boson mass and the 95 gev excesses. 2023.

[28] Junjie Cao, Xinglong Jia, Jingwei Lian, and Lei Meng. 95 gev diphoton and $b\bar{b}$ excesses in the general next-to-minimal supersymmetric standard model. 2023.

[29] Juhi Dutta, Jayita Lahiri, Cheng Li, Gudrid Moortgat-Pick, Sheikh Farah Tabira, and Julia Anabell Ziegler. Dark matter phenomenology in 2hdms in light of the 95 gev excess. 2023.

[30] J. A. Aguilar-Saavedra, H. B. Câmara, F. R. Joaquim, and J. F. Seabra. Confronting the 95 gev excesses within the $\mathrm{U}(1)^{'}$-extended next-to-minimal 2hdm. *Phys. Rev. D*, 108:075020, Oct 2023. URL https://link.aps.org/doi/10.1103/PhysRevD.108.075020.

[31] Thomas Biekötter, Sven Heinemeyer, and Georg Weiglein. The cms di-photon excess at 95 gev in view of the lhc run 2 results. *Physics Letters B*, 846:138217, November 2023. ISSN 0370-2693. URL http://dx.doi.org/10.1016/j.physletb.2023.138217.

[32] Xiao-Gang He, Zhong-Lv Huang, Ming-Wei Li, and Chia-Wei Liu. The sm expected branching ratio for $h \to \gamma\gamma$ and an excess for $h \to z\gamma$. 2024.

[33] Abdesslam Arhrib, Khiem Hong Phan, Van Que Tran, and Tzu-Chiang Yuan. When standard model higgs meets its lighter 95 gev higgs. 2024.

[34] Rachid Benbrik, Mohammed Boukidi, and Stefano Moretti. Superposition of CP-Even and CP-Odd Higgs Resonances: Explaining the 95 GeV Excesses within a Two-Higgs Doublet Model. 5 2024.

[35] Ulrich Ellwanger, Cyril Hugonie, Stephen F. King, and Stefano Moretti. NMSSM Explanation for Excesses in the Search for Neutralinos and Charginos and a 95 GeV Higgs Boson. 4 2024.

[36] Stefano Moretti and Shaaban Khalil. *Supersymmetry Beyond Minimality: From Theory to Experiment*. CRC Press, 2019. ISBN 978-0-367-87662-3.

[37] Ahmed Ali Abdelalim, Biswaranjan Das, Shaaban Khalil, and Stefano Moretti. Di-photon decay of a light Higgs state in the BLSSM. *Nucl. Phys. B*, 985:116013, 2022. .

[38] Oliver Brein. Adaptive scanning—a proposal how to scan theoretical predictions over a multi-dimensional parameter space efficiently. *Computer Physics Communications*, 170(1):42–48, July 2005. ISSN 0010-4655. URL `http://dx.doi.org/10.1016/j.cpc.2005.03.104`.

[39] David W. Hogg and Daniel Foreman-Mackey. Data analysis recipes: Using markov chain monte carlo*. *The Astrophysical Journal Supplement Series*, 236 (1):11, May 2018. ISSN 1538-4365. URL `http://dx.doi.org/10.3847/1538-4365/aab76e`.

[40] Joshua Albert, Csaba Balazs, Andrew Fowlie, Will Handley, Nicholas Hunt-Smith, Roberto Ruiz de Austri, and Martin White. A comparison of bayesian sampling algorithms for high-dimensional particle physics and cosmology applications, 2024.

[41] Joshua S. Speagle. A conceptual introduction to markov chain monte carlo methods, 2019.

[42] F. Feroz, M. P. Hobson, and M. Bridges. Multinest: an efficient and robust bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, October 2009. ISSN 1365-2966. URL `http://dx.doi.org/10.1111/j.1365-2966.2009.14548.x`.

[43] HEP ML Community. A Living Review of Machine Learning for Particle Physics. URL `https://iml-wg.github.io/HEPML-LivingReview/`.

[44] Rajneil Baruah, Subhadeep Mondal, Sunando Kumar Patra, and Satyajit Roy. Probing intractable beyond-standard-model parameter spaces armed with machine learning. *The European Physical Journal Special Topics*, July 2024.

ISSN 1951-6401. URL
`http://dx.doi.org/10.1140/epjs/s11734-024-01236-w`.

[45] A. Hammad, Myeonghun Park, Raymundo Ramos, and Pankaj Saha. Exploration of parameter spaces assisted by machine learning. *Computer Physics Communications*, 293:108902, December 2023. ISSN 0010-4655. URL
`http://dx.doi.org/10.1016/j.cpc.2023.108902`.

[46] Mark D. Goodsell and Ari Joury. Active learning bsm parameter spaces. *The European Physical Journal C*, 83(4), April 2023. ISSN 1434-6052. URL
`http://dx.doi.org/10.1140/epjc/s10052-023-11368-3`.

[47] Mark D. Goodsell and Ari Joury. Bsmart: Simple and fast parameter space scans. volume 297, page 109057, 2024. URL
`https://www.sciencedirect.com/science/article/pii/S0010465523004022`.

[48] Fernando Abreu de Souza, Miguel Crispim Romão, Nuno Filipe Castro, Mehraveh Nikjoo, and Werner Porod. Exploring parameter spaces with artificial intelligence and machine learning black-box optimization algorithms. *Phys. Rev. D*, 107:035004, Feb 2023. URL
`https://link.aps.org/doi/10.1103/PhysRevD.107.035004`.

[49] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. .

[50] Nikolaus Hansen. The cma evolution strategy: A tutorial. 2023.

[51] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization, 2022. URL
`https://arxiv.org/abs/2206.03301`.

[52] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X.

[53] Florian Staub, Thorsten Ohl, Werner Porod, and Christian Speckner. A Tool Box for Implementing Supersymmetric Models. *Comput. Phys. Commun.*, 183: 2165–2206, 2012. .

[54] Florian Staub. xbit: an easy to use scanning tool with machine learning abilities. 2019.

[55] Liangliang Shang and Yang Zhang. EasyScan_HEP: A tool for connecting programs to scan the parameter space of physics models. *Comput. Phys. Commun.*, 296:109027, 2024. .

[56] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012. .

[57] CMS Collaboration. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012. .

[58] Steven Weinberg. A model of leptons. *Phys. Rev. Lett.*, 19:1264–1266, Nov 1967. URL `https://link.aps.org/doi/10.1103/PhysRevLett.19.1264`.

[59] Abdus Salam and J. C. Ward. Weak and electromagnetic interactions. *Il Nuovo Cimento (1955-1965)*, 11(4):568–577, 1959. ISSN 1827-6121. URL `https://doi.org/10.1007/BF02726525`.

[60] Sheldon L. Glashow. The renormalizability of vector meson interactions. *Nuclear Physics*, 10:107–117, 1959.

[61] David Gross and Frank Wilczek. Ultraviolet behavior of non-abelian gauge theories. *Physical Review Letters*, 30:1343–1346, 1973. .

[62] H. David Politzer. Reliable perturbative results for strong interactions? *Physical Review Letters*, 30:1346–1349, 1973. .

[63] Franz et al. Gross. 50 years of quantum chromodynamics: Introduction and review. *The European Physical Journal C*, 83(12), December 2023. ISSN 1434-6052. URL `http://dx.doi.org/10.1140/epjc/s10052-023-11949-2`.

[64] François Englert and Robert Brout. Broken symmetry and the mass of gauge vector mesons. *Physical Review Letters*, 13(9):321–323, 1964. .

[65] Peter W. Higgs. Spontaneous symmetry breakdown without massless bosons. *Physical Review*, 145(4):1156–1163, 1966. .

[66] Gerald S. Guralnik, C. R. Hagen, and T. W. B. Kibble. Global conservation laws and massless particles. *Physical Review Letters*, 13(20):585–587, 1964. .

[67] Claude Bouchiat, John Iliopoulos, and Philippe Meyer. An anomaly-free version of weinberg's model. *Physics Letters B*, 38(7):519–523, 1972. URL `https://www.sciencedirect.com/science/article/pii/0370269372905321`.

[68] Gerard 't Hooft. Renormalizable lagrangians for massive yang-mills fields. *Nuclear Physics B*, 35:167–188, 1971. URL `https://www.sciencedirect.com/science/article/pii/0550321371901398`.

[69] Lochlainn O'Raifeartaigh and Norbert Straumann. Gauge theory: Historical origins and some modern developments. *Reviews of Modern Physics*, 72(1):1–23, 2000. URL `https://link.aps.org/doi/10.1103/RevModPhys.72.1`.

[70] ATLAS Collaboration. A detailed map of Higgs boson interactions by the ATLAS experiment ten years after the discovery. *Nature*, 607:52–59, 2022. URL https://www.nature.com/articles/s41586-022-04893-w.

[71] Elina Fuchs, Silja Thewes, and Georg Weiglein. Interference effects in bsm processes with a generalised narrow-width approximation. *Eur. Phys. J. C*, 75 (6):254, 2015. URL https://link.springer.com/article/10.1140/epjc/s10052-015-3472-z.

[72] M. Sajjad Athar, Steven W. Barwick, Thomas Brunner, Jun Cao, Mikhail Danilov, Kunio Inoue, Takaaki Kajita, Marek Kowalski, Manfred Lindner, Kenneth R. Long, Nathalie Palanque-Delabrouille, Werner Rodejohann, Heidi Schellman, Kate Scholberg, Seon-Hee Seo, Nigel J.T. Smith, Walter Winter, Geralyn P. Zeller, and Renata Zukanovich Funchal. Status and perspectives of neutrino physics. *Progress in Particle and Nuclear Physics*, 124:103947, May 2022. ISSN 0146-6410. URL http://dx.doi.org/10.1016/j.ppnp.2022.103947.

[73] Marco Cirelli, Alessandro Strumia, and Jure Zupan. Dark matter, 2024. URL https://arxiv.org/abs/2406.01705.

[74] Mustapha Ishak. What makes the universe accelerate? a review on what dark energy could be and how to test it. *Classical and Quantum Gravity*, 37(6): 063001, 2020. URL https://iopscience.iop.org/article/10.1088/1361-6382/ab6a7c.

[75] Mark D. Goodsell. Lectures on supersymmetry part 1. Available at: https://www.lpthe.jussieu.fr/ goodsell/SUSY/part1.pdf (Accessed: 08/03/2025), 2019-2020.

[76] Melissa van Beekveld, Sascha Caron, and Roberto Ruiz de Austri. The current status of fine-tuning in supersymmetry. *J. High Energ. Phys.*, 2020(1):147, 2020. URL https://arxiv.org/abs/1906.10706.

[77] Hyun Min Lee. Lectures on physics beyond the standard model. *Journal of the Korean Physical Society*, 78(11):985–1017, May 2021. ISSN 1976-8524. URL http://dx.doi.org/10.1007/s40042-021-00188-x.

[78] Benjamin Allanach. Beyond the standard model. *CERN Yellow Reports: School Proceedings*, pages Vol 5 (2017): Proceedings of the 2016 European School of High–Energy Physics, 2017. URL https://e-publishing.cern.ch/index.php/CYRSP/article/view/353.

[79] Fernando Quevedo and Andreas Schachner. Cambridge lectures on the standard model, 2024. URL https://arxiv.org/abs/2409.09211.

[80] C. P. Burgess. Quantum gravity in everyday life: General relativity as an effective field theory. *Living Reviews in Relativity*, 7(1):5, 2004. URL `https://link.springer.com/article/10.12942/lrr-2004-5`.

[81] Claus Kiefer. Quantum gravity: General introduction and recent developments. *Annalen der Physik*, 15(1-2):129–148, 2005. URL `https://arxiv.org/abs/gr-qc/0508120`.

[82] Andreas Crivellin and Bruce Mellado. Anomalies in particle physics and their implications for the standard model. *Nature Reviews Physics*, 6:294–309, 2024. URL `https://www.nature.com/articles/s42254-024-00703-6`.

[83] ATLAS C. Arcangeletti. Measurement of higgs boson production and search for new resonances in final states with photons and z bosons, with the atlas detector, 2023. URL `https://indico.cern.ch/event/1281604/`.

[84] S. F. Novaes. Standard model: An introduction, 2000.

[85] G. C. Branco, P. M. Ferreira, L. Lavoura, M. N. Rebelo, Marc Sher, and Joao P. Silva. Theory and phenomenology of two-Higgs-doublet models. *Phys. Rept.*, 516:1–102, 2012.

[86] Heather E. Logan and Vikram Rentala. All the generalized georgi-machacek models. *Physical Review D*, 92(7), October 2015. ISSN 1550-2368. URL `http://dx.doi.org/10.1103/PhysRevD.92.075011`.

[87] Michael Duerr, Pavel Fileviez Pérez, and Juri Smirnov. Scalar dark matter: direct vs. indirect detection. *Journal of High Energy Physics*, 2016(6), June 2016. ISSN 1029-8479. URL `http://dx.doi.org/10.1007/JHEP06(2016)152`.

[88] Giorgio Arcadi, Abdelhak Djouadi, and Martti Raidal. Dark matter through the higgs portal. *Physics Reports*, 842:1–180, February 2020. ISSN 0370-1573. URL `http://dx.doi.org/10.1016/j.physrep.2019.11.003`.

[89] André de Gouvêa. Neutrino mass models. *Annual Review of Nuclear and Particle Science*, 66:197–217, 2016. URL `https://www.annualreviews.org/doi/10.1146/annurev-nucl-102115-044600`.

[90] Igor P. Ivanov. Building and testing models with extended higgs sectors. *Progress in Particle and Nuclear Physics*, 95:160–208, July 2017. ISSN 0146-6410. URL `http://dx.doi.org/10.1016/j.ppnp.2017.03.001`.

[91] S.A.R. Ellis, R.M. Godbole, S. Gopalakrishna, and J.D. Wells. Survey of vector-like fermion extensions of the standard model and their phenomenological implications. *Journal of High Energy Physics*, 2014(9):130, 2014. .

[92] Nicolas Bizot and Michele Frigerio. Fermionic extensions of the standard model in light of the higgs couplings. *Journal of High Energy Physics*, 2016(1):36, 2016. .

[93] Thomas G. Rizzo. *Z' Phenomenology and the LHC*, page 537–575. WORLD SCIENTIFIC, January 2008. ISBN 9789812819260. URL `http://dx.doi.org/10.1142/9789812819260_0009`.

[94] Daniel Hayden, Raymond Brock, and Christopher Willis. Z prime: A story, 2013.

[95] Ilja Doršner, Emina Džaferović-Mašić, and Shaikh Saad. Parameter space exploration of the minimal SU(5) unification. *Phys. Rev. D*, 104(1):015023, 2021. .

[96] Frank F. Deppisch, Tomás E. Gonzalo, and Lukas Graf. Surveying the so(10) model landscape: The left-right symmetric case. *Physical Review D*, 96(5), September 2017. ISSN 2470-0029. URL `http://dx.doi.org/10.1103/PhysRevD.96.055003`.

[97] H. Kawase and N. Maekawa. Flavor structure of e6 gut models. *Progress of Theoretical Physics*, 123(6):941–955, June 2010. ISSN 1347-4081. URL `http://dx.doi.org/10.1143/PTP.123.941`.

[98] Yu. A. Golfand and E. P. Likhtman. Extension of the algebra of poincaré group generators and violation of p invariance. *JETP Letters*, 13:323–326, 1971. URL `https://jetpletters.ru/ps/1584/article_24309.shtml`. Translated from *ZhETF Pis'ma*, vol. 13, pp. 452–455.

[99] J. Wess and B. Zumino. Supergauge transformations in four dimensions. *Nuclear Physics B*, 70:39–50, 1974. URL `https://cds.cern.ch/record/201649`.

[100] Shaaban Khalil and Stefano Moretti. *Supersymmetry Beyond Minimality: From Theory to Experiment*. CRC Press, Boca Raton, 2017. ISBN 978-1-4987-5673-0.

[101] David Bailin and Alexander Love. *Supersymmetric Gauge Field Theory and String Theory*. Taylor & Francis, London and New York, 1994. ISBN 9780750302600.

[102] Julius Wess and Jonathan Bagger. *Supersymmetry and Supergravity*. Princeton University Press, Princeton, NJ, 2nd edition, 1992. ISBN 9780691025308.

[103] Christian Saemann. Introduction to supersymmetry, 2009. Lecture notes, Trinity College Dublin. Available at `https://christiansaemann.de/wp-content/uploads/2022/07/LecturesOnSUSY.pdf`.

[104] A DJOUADI. The anatomy of electroweak symmetry breaking tome ii: The higgs bosons in the minimal supersymmetric model. *Physics Reports*, 459(1–6):

1–241, April 2008. ISSN 0370-1573. URL http://dx.doi.org/10.1016/j.physrep.2007.10.005.

[105] Jihn E. Kim and H.P. Nilles. The -problem and the strong cp-problem. *Physics Letters B*, 138(1):150–154, 1984. ISSN 0370-2693. URL https://www.sciencedirect.com/science/article/pii/0370269384918902.

[106] Ann E. Nelson and Tuhin S. Roy. Generalized supersoft supersymmetry breaking and a solution to the μ problem. *Phys. Rev. Lett.*, 114:201802, May 2015. URL https://link.aps.org/doi/10.1103/PhysRevLett.114.201802.

[107] Radovan Dermisek and John F. Gunion. The nmssm solution to the fine-tuning problem, precision electroweak constraints, and the largest lep higgs event excess. *Physical Review D*, 76(9):095006, 2007.

[108] Keith A. Ulmer. Supersymmetry: Experimental status, 2016. URL https://arxiv.org/abs/1601.03774.

[109] Howard Baer, Vernon Barger, Shadman Salam, Dibyashree Sengupta, and Xerxes Tata. The lhc higgsino discovery plane for present and future susy searches. *arXiv preprint arXiv:2007.09252*, 2020.

[110] Marcela Carena, Howard E. Haber, Ian Low, Nausheen R. Shah, and Carlos E. M. Wagner. Alignment limit of the NMSSM Higgs sector. *Phys. Rev. D*, 93 (3):035013, 2016. .

[111] Radovan Dermisek and John F. Gunion. The nmssm solution to the fine-tuning problem, precision electroweak constraints, and the largest lep higgs event excess. *Physical Review D*, 76(9):095006, 2007.

[112] S. Khalil and A. Masiero. Radiative b-l symmetry breaking in supersymmetric models. *Physics Letters B*, 665(5):374–377, 2008. ISSN 0370-2693. URL https://www.sciencedirect.com/science/article/pii/S0370269308007703.

[113] Shaaban Khalil. Higgs Bosons in B-L Supersymmetric Standard Model. *LHEP*, 2023:454, 2023. .

[114] Florian Staub. Sarah 4: A tool for (not only susy) model builders. *Computer Physics Communications*, 185(6):1773–1790, June 2014. ISSN 0010-4655. URL http://dx.doi.org/10.1016/j.cpc.2014.02.018.

[115] F. Staub. Sarah, 2012. URL https://arxiv.org/abs/0806.0538.

[116] Adam Alloul, Neil D. Christensen, Céline Degrande, Claude Duhr, and Benjamin Fuks. Feynrules 2.0— a complete toolbox for tree-level phenomenology. *Computer Physics Communications*, 185(8):2250–2300, August 2014. ISSN 0010-4655. URL http://dx.doi.org/10.1016/j.cpc.2014.04.012.

[117] A. Semenov. Lanhep - a package for automatic generation of feynman rules from the lagrangian. updated version 3.2, 2014.

[118] Céline Degrande et al. Ufo - the universal feynrules output. *Comput. Phys. Commun.*, 183(6):1201–1214, 2012.

[119] B. C. Allanach et al. Susy les houches accord 2. *Comput. Phys. Commun.*, 180 (1):8–25, 2009. .

[120] W. Porod and F. Staub. Spheno 3.1: extensions including flavour, cp-phases and models beyond the mssm. *Computer Physics Communications*, 183(11): 2458–2469, November 2012. ISSN 0010-4655. URL http://dx.doi.org/10.1016/j.cpc.2012.05.021.

[121] Werner Porod. Spheno, a program for calculating supersymmetric spectra, susy particle decays and susy particle production at e+e- colliders. *Computer Physics Communications*, 153:275–315, 2003. URL https://api.semanticscholar.org/CorpusID:7905927.

[122] Alexander Belyaev, Neil D. Christensen, and Alexander Pukhov. Calchep 3.4 for collider physics within and beyond the standard model. *Computer Physics Communications*, 184(7):1729–1769, July 2013. ISSN 0010-4655. URL http://dx.doi.org/10.1016/j.cpc.2013.01.014.

[123] Johan Alwall, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. Madgraph 5: going beyond. *Journal of High Energy Physics*, 2011(6), June 2011. ISSN 1029-8479. URL http://dx.doi.org/10.1007/JHEP06(2011)128.

[124] Johan Alwall et al. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *J. High Energy Phys.*, 2014(7):79, 2014.

[125] Christian Bierlich, Smita Chakraborty, Nishita Desai, Leif Gellersen, Ilkka Helenius, Philip Ilten, Leif Lönnblad, Stephen Mrenna, Stefan Prestel, Christian T. Preuss, Torbjörn Sjöstrand, Peter Skands, Marius Utheim, and Rob Verheyen. A comprehensive guide to the physics and usage of PYTHIA 8.3. *SciPost Phys. Codebases*, 8, 2022. .

[126] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057, 2014. .

[127] David Eriksson, Johan Rathsman, and Oscar Stål. 2HDMC: Two-Higgs-Doublet Model Calculator. *Comput. Phys. Commun.*, 181:189–205, 2010. .

[128] Julien Baglio, Ramona Gröber, Margarete Mühlleitner, Dao Thi Nhung, Heidi Rzehak, Michael Spira, Juraj Streicher, and Kathrin Walz. NMSSMCALC: A Program Package for the Calculation of Loop-Corrected Higgs Boson Masses and Decay Widths in the (Complex) NMSSM. *Comput. Phys. Commun.*, 185: 3372–3391, 2014. .

[129] P. Bechtle, O. Brein, S. Heinemeyer, G. Weiglein, and K.E. Williams. Higgsbounds: Confronting arbitrary higgs sectors with exclusion bounds from lep and the tevatron. *Computer Physics Communications*, 181(1):138–167, January 2010. ISSN 0010-4655. URL `http://dx.doi.org/10.1016/j.cpc.2009.09.003`.

[130] Philip Bechtle, Sven Heinemeyer, Oscar Stål, Tim Stefaniak, and Georg Weiglein. Higgssignals: Confronting arbitrary higgs sectors with measurements at the tevatron and the lhc. *The European Physical Journal C*, 74(2), February 2014. ISSN 1434-6052. URL `http://dx.doi.org/10.1140/epjc/s10052-013-2711-4`.

[131] Henning Bahl, Thomas Biekötter, Sven Heinemeyer, Cheng Li, Steven Paasch, Georg Weiglein, and Jonas Wittbrodt. Higgstools: Bsm scalar phenomenology with new versions of higgsbounds and higgssignals. *Computer Physics Communications*, 291:108803, October 2023. ISSN 0010-4655. URL `http://dx.doi.org/10.1016/j.cpc.2023.108803`.

[132] S. Navas and others (Particle Data Group). Review of particle physics. *Phys. Rev. D*, 110:030001, 2024. URL `https://pdg.lbl.gov/`.

[133] Richard E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, NJ, 1961. ISBN 9780691079011.

[134] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

[135] Dehua Peng, Zhipeng Gui, and Huayi Wu. Interpreting the curse of dimensionality from distance concentration and manifold effect, 2023.

[136] Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation, 2021.

[137] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020.

[138] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. . https://distill.pub/2019/visual-exploration-gaussian-processes.

[139] Grigorios A Pavliotis. Stochastic processes and applications. *Texts in applied mathematics*, 60, 2014.

[140] Joram Soch et al. The book of statistical proofs, January 2025. URL https://doi.org/10.5281/zenodo.14646799.

[141] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[142] Chenyi Lyu, Xingchi Liu, and Lyudmila Mihaylova. *Review of Recent Advances in Gaussian Process Regression Methods*, page 226–237. Springer Nature Switzerland, 2024. ISBN 9783031555688. URL http://dx.doi.org/10.1007/978-3-031-55568-8_19.

[143] Tomasz Szandała. *Review and comparison of commonly used activation functions for deep neural networks*. Springer, 2021.

[144] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023. URL http://udlbook.com.

[145] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. 2019.

[146] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[147] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[148] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics amp; Epidemiology*, 1(1):161–187, January 2017. ISSN 2470-9379. . URL http://dx.doi.org/10.1080/24709360.2017.1396742.

[149] Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, page 211–214. IEEE, June 2019. URL http://dx.doi.org/10.1109/AIKE.2019.00044.

[150] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[151] Alejandro Segura and Angie Catalina Parra. A practical guide to statistical techniques in particle physics, 2024.

[152] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006. ISBN 978-0387310732. URL https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/.

[153] Benyamin Ghojogh, Hadi Nekoei, Aydin Ghojogh, Fakhri Karray, and Mark Crowley. Sampling algorithms, from survey sampling to monte carlo methods: Tutorial and literature review. *arXiv preprint arXiv:2011.00901*, 2020.

[154] Daniel Foreman-Mackey, David W. Hogg, Dustin Lang, and Jonathan Goodman. emcee: The mcmc hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306–312, March 2013. ISSN 1538-3873. URL http://dx.doi.org/10.1086/670067.

[155] Michael Betancourt and Mark Girolami. *Hamiltonian Monte Carlo for Hierarchical Models*, page 79–101. Chapman and Hall/CRC, May 2015. ISBN 9781482235128. URL http://dx.doi.org/10.1201/b18502-5.

[156] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[157] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[158] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

[159] Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces, 2021.

[160] Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. Diversity-guided multi-objective bayesian optimization with batch evaluations. *Advances in Neural Information Processing Systems*, 33:17708–17720, 2020.

[161] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34: 2187–2200, 2021.

[162] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, dec 1997. ISSN 0098-3500. URL `https://doi.org/10.1145/279232.279236`.

[163] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. Bradford Books, Cambridge, MA, 2 edition, 2018.

[164] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley-Interscience, Newy York, 2014.

[165] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. URL `https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf`.

[166] Congye Wang, Wilson Chen, Heishiro Kanagawa, Chris Oates, et al. Reinforcement learning for adaptive mcmc. *arXiv preprint arXiv:2405.13574*, 2024.

[167] Theophane Weber, Nicolas Heess, Ali Eslami, John Schulman, David Wingate, and David Silver. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2015.

[168] Hadi S. Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. Hyp-rl : Hyperparameter optimization by reinforcement learning, 2019.

[169] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL `https://arxiv.org/abs/1509.02971`.

[170] Joshua Achiam. Spinning up in deep reinforcement learning. 2018.

[171] T. R. Harvey and A. Lukas. Particle physics model building with reinforcement learning, 2021.

[172] George N. Wojcik, Shu Tian Eu, and Lisa L. Everett. Graph reinforcement learning for exploring bsm model spaces, 2024. URL https://arxiv.org/abs/2407.07203.

[173] James Halverson, Brent Nelson, and Fabian Ruehle. Branes with brains: exploring string vacua with deep reinforcement learning. *Journal of High Energy Physics*, 2019(6):3, 2019. ISSN 1029-8479. URL https://doi.org/10.1007/JHEP06(2019)003.

[174] Gustavo Malkomes, Bolong Cheng, Eric H Lee, and Mike Mccourt. Beyond the pareto efficient frontier: Constraint active search for multiobjective experimental design. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7423–7434. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/malkomes21a.html.

[175] Mauricio A. Diaz, Giorgio Cerro, Srinandan Dasmahapatra, and Stefano Moretti. Bayesian active search on parameter space: a 95 gev spin-0 resonance in the $(b-l)$ssm. 2024. URL https://arxiv.org/abs/2404.18653.

[176] Mauricio A. Diaz. Code repository for 2404.18653. 2025. URL https://github.com/mjadiaz/blssm-bcastor. Accessed: 2025-01-07.

[177] F. Feroz, M. P. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon. Not. Roy. Astron. Soc.*, 398:1601–1614, 2009. .

[178] Jonas Wessén and Eliel Camargo-Molina. A diversity-enhanced genetic algorithm for efficient exploration of parameter spaces. 12 2024.

[179] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. 2016.

[180] I. M. Sobol. Distribution of points in a cube and approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7: 784–802, 1967.

[181] David Luengo, Luca Martino, Mónica Bugallo, Víctor Elvira, and Simo Särkkä. A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020(1), May 2020. ISSN 1687-6180. URL http://dx.doi.org/10.1186/s13634-020-00675-6.

[182] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*,

7(1):110–120, 1997. ISSN 10505164. URL http://www.jstor.org/stable/2245134.

[183] Thomas Bäck. Title Pages. In *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 02 1996. ISBN 9780195099713. .

[184] Ben O'Leary, Werner Porod, and Florian Staub. Mass spectrum of the minimal susy b-l model. *Journal of High Energy Physics*, 2012(5), May 2012. ISSN 1029-8479. URL http://dx.doi.org/10.1007/JHEP05(2012)042.

[185] Ahmed Ali Abdelalim, Biswaranjan Das, Shaaban Khalil, and Stefano Moretti. Di-photon decay of a light higgs state in the blssm. *Nuclear Physics B*, 985: 116013, 2022. ISSN 0550-3213. URL https://www.sciencedirect.com/science/article/pii/S0550321322003649.

[186] Fernando Abreu de Souza, Miguel Crispim Romão, Nuno Filipe Castro, Mehraveh Nikjoo, and Werner Porod. Exploring parameter spaces with artificial intelligence and machine learning black-box optimization algorithms. *Phys. Rev. D*, 107(3):035004, 2023. .

[187] University of Southampton. Iridis research computing facility. https://www.southampton.ac.uk/research/facilities/iridis-research-computing-facility. Accessed: 20 January 2025.

[188] B.S. Kronheim, M.P. Kuchera, H.B. Prosper, and A. Karbo. Bayesian neural networks for fast susy predictions. *Physics Letters B*, 813:136041, February 2021. ISSN 0370-2693. URL http://dx.doi.org/10.1016/j.physletb.2020.136041.

[189] F. Staub. SARAH. 6 2008.

[190] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, page 843–850, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

[191] The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL https://doi.org/10.5281/zenodo.3509134.

[192] Henning Bahl, Thomas Biekötter, Sven Heinemeyer, Cheng Li, Steven Paasch, Georg Weiglein, and Jonas Wittbrodt. HiggsTools: BSM scalar phenomenology with new versions of HiggsBounds and HiggsSignals. *Comput. Phys. Commun.*, 291:108803, 2023. .

[193] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21: 1087–1092, 1953. .

[194] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57:97–109, 1970. .

[195] Luc Darmé, Céline Degrande, Claude Duhr, Benjamin Fuks, Mark Goodsell, Gudrun Heinrich, Valentin Hirschi, Stefan Höche, Marius Höfer, Joshua Isaacson, Olivier Mattelaer, Thorsten Ohl, Davide Pagani, Jürgen Reuter, Peter Richardson, Steffen Schumann, Hua-Sheng Shao, Frank Siegert, and Marco Zaro. UFO 2.0 – The Universal Feynman Output format. *Eur. Phys. J. C*, 83(7):631, 2023. .

[196] Peter Z. Skands et al. SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators. *JHEP*, 07:036, 2004. .

[197] Andy Buckley. Pyslha: a pythonic interface to susy les houches accord data. 2015. URL `https://arxiv.org/abs/1305.4194`.

[198] Florian Staub. xslha: An les houches accord reader for python and mathematica. *Computer Physics Communications*, 241:132–138, August 2019. ISSN 0010-4655. . URL `http://dx.doi.org/10.1016/j.cpc.2019.03.013`.

[199] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize, October 2021. URL `https://doi.org/10.5281/zenodo.5565057`.

[200] Omry Yadan, Jasha Sommer-Simpson, and Olivier Delalleau. omegaconf, 2019. URL `https://github.com/omry/omegaconf`.

[201] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL `http://arxiv.org/abs/1910.06403`.

[202] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[203] Jorge Crispim Romão and Miguel Crispim Romão. Combining evolutionary strategies and novelty detection to go beyond the alignment limit of the z_3 3hdm. 2024.

[204] Mauricio A. Diaz, Srinandan Dasmahapatra, and Stefano Moretti. hep-aid: A python library for sample efficient parameter scans in beyond the standard model phenomenology, 2024.

[205] Wolfgang Waltenberger, André Lessa, and Sabine Kraml. Artificial proto-modelling: building precursors of a next standard model from simplified

model results. *Journal of High Energy Physics*, 2021(3), March 2021. ISSN 1029-8479. URL `http://dx.doi.org/10.1007/JHEP03(2021)207`.