Towards algorithmic theory construction in physics

Hampus Möller

Fysikum
Master thesis 45 HE Credits
Physics
Master program in physics (120 Credits)
Autumn term 2023
Supervisor: Jens Jasche

Towards algorithmic theory construction in physics

Hampus Möller

Abstract

Keywords: Symbolic regression, machine learning, symmetry, minimum description length, entropy.

This master thesis explores the challenge of algorithmic hypothesis generation and its connection to potential inductive biases. In the scientific method, hypotheses are formulated and tested against data for validity. The process of hypothesis generation is, however, not explicitly formulated. A structured approach to hypothesis generation would allow for a near algorithmic process that could scale far beyond the current capabilities of science.

The thesis explored the concepts of entropy, symmetry and minimum description length for use as inductive biases. Two algorithms were implemented and evaluated: one for symmetry finding and one for symbolic regression. The theoretical results show a strong connection between entropy and minimum description length with a weaker connection to symmetry. Both implementations indicate potential paths exist to partial or full automation of the hypothesis generation process. In general, there do not seem to exist fundamental issues in automating the process besides the challenge of its implementation.

Thus, the thesis demonstrates a clear path forward towards partial or complete automation of the scientific method for physical research.

Contents

1 Introduction

If you were to search an infinite space for an object you only know approximately what it looks like, where would you start?

Compared to our ancestors, we are, by most metrics, living in a state of prosperity. Central to our ability to create prosperity is our ability to change and harness the world around us. We build buildings to keep us warm and safe, roads to travel and machines to perform mechanical work. We grow our food on farms and synthesize life-saving medications using chemistry. We use physics to build computers, siphon energy from sunlight and split the atom, for better and for worse.

Key to all the examples above is that effective utilization of the world comes from understanding cause and effect. A well-understood system can be manipulated to yield the intended outcome, whether that be increased output, new possibilities or higher efficiency. Historically, this knowledge has been used to construct machines such as mills and cars, which perform our manual labour. More recently, it has been used to create digital computers, which help us with information processing to the degree that computers outperform humans in many tasks.

One task which has not been augmented by technology is the process of scientific inference. The measurement devices and calculations have improved, but the general process of scientific discovery has not seen significant changes since the formalization of the scientific method.

Where our previous inventions fall short, the rise of artificial intelligence and machine learning may allow us to augment the scientific process. The changes could potentially increase the output of scientists or make science more available by removing the significant overhead knowledge required. It could also change the nature of research to be more concerned with meta-science, in which researchers decide which phenomena need to be analyzed and how, while machines perform the practical steps. If the process could be formulated as an algorithm, it could potentially allow for an exponential growth of knowledge, ultimately maybe even surpassing the limits of human understanding.

The idea, as outlined above, is not unique to this thesis. Several groups are attempting to formulate their own algorithms, for instance, using classic machine learning [1], language models [2] and more specialized approaches [3]. Currently, there is no guarantee that any algorithm is neither efficient nor optimal.

The problem may be approached top-down, where a set of successful heuristics defines the algorithm. It may also be constructed bottom-up, attempting to use fundamental relationships of nature to find the solutions. If the structure of nature can be identified, an optimal strategy should be possible to realize.

In the thesis, we aim to explore the possibility of enhancing the process of understanding itself using artificial intelligence. This is done in a bottom-up

approach where we inspect the foundational components of science and identify the potential paths to automate the process, starting with the scientific method.

1.1 Historical examples of inference

Obvious, given the success of science, is that we historically have been able to infer successful hypotheses repeatedly, where successful refers to their practical applicability. From the many important discoveries of the last millennia, we will highlight three examples from James Maxwell, Albert Einstein and Wolfgang Pauli.

Maxwell's equations form a unified view of electromagnetism and constitute an essential contribution to modern science. The equations encode the electromagnetic framework in 20 equations, typically written as four differential equations [4]. Most of the individual constituent components of Maxwell's equations were already discovered. What Maxwell introduced was the hypothetical notion of the electromagnetic field [4], providing a causal explanation of electromagnetism and its action on physically separated objects. With the field theory, he derives the necessary displacement current and can formulate the unified equations of electromagnetism [4]. From his unified equations, he also predicts the speed of electromagnetic waves, which coincides with the measured speed of light, supporting the idea that light is electromagnetic in nature [4].

Based on Maxwell's equations, Einstein discovered special relativity. Using the equations as a basis, he attempted to find spatial transformations that preserved the form of the equations [5]. Einstein indicated that he had many attempts at reconciling Galilean transformations of light before suggesting the Lorentz transform [6]. In his discovery, Einstein relied primarily on Maxwell's theoretical framework of electromagnetism and less on the experimental evidence [5]. While Lorentz independently derived the Lorentz transform before Einstein did, he did not understand the physical implications of time dilation [6].

Finally, Pauli suggested the introduction of a new particle, now known as the neutrino, to ameliorate the difference in theoretical predictions and experimental evidence in the β-decay [7]. Specifically, experiments indicated that energy, momentum and spin were not conserved in the reaction. Pauli's contribution was a single particle which corrected all of the missing conservation laws [7], but was considered a "desperate remedy" [8, p. 1]. The hypothesis was consistent with the theory, but the particle was almost undetectable and not well received by all contemporaries [8]. The basic idea was simple yet grew into a more nuanced theory over time, strengthened by an increasing amount of indirect evidence [8].

In all the listed cases, they seem to share two general properties:

1. Successful hypotheses rely on well-established principles or observations.
2. Successful hypotheses offer a causal explanation of the observations.

The first property indicates that hypotheses are based on previous knowledge and experience, hinting at the presence of a structure. The second is that a good hypothesis should not only predict but also explain the observations beyond the particular observed case. The above accounts rely on available published knowledge as well as the testament of the scientists. Such recollections can potentially be biased since it may be hard to recall the process of discovery.

1.2 The foundation of understanding

> "The most incomprehensible thing about the world is that it is comprehensible."
>
> Albert Einstein, [9]

Unifying all the previously listed discoveries is that they are scientific in nature, following the scientific method. Hypothesis generation is the first stage of the scientific method, and all other steps follow from the given hypothesis [10, p. 104]. Arguably, the understanding of a process is the hypothesis generation itself. The central issue is that there exists no explicit method to generate a hypothesis. Analysis of the process indicates it relies on experience and previous observations [10, p. 104] or, in less informative terms, "tremendous insight" [11, p. 178].

Contrary to such observations, authors such as A.C. Benjamin [12] hold that there seems to be more structure to hypothesis generation than typically attributed to it. The search space of possible hypothesis one can consider is likely infinite [13], as such, if there was no logical structure in hypothesis generation then it is a "miracle" that we would be able to find a correct hypothesis [13, p. 3229]. For these reasons, we believe that there *by necessity* exists a "logic of discovery" and that it should be possible to formalize the process of hypothesis generation.

If an algorithmic approach to hypothesis generation can be formulated, even if highly inefficient, it would allow for a scalable approach to science limited only by the computational resources and data at hand. Due to the fundamental nature of the question, such a technology could potentially allow for exponential growth in the understanding of the world. This would likely lead to informational imbalance and major competitive advantages in most fields, including academia, economy and defence. Failure to implement such a technology, if available to competitors, would almost certainly spell the end of the enterprise given sufficient time.

1.3 The scientific method

At the core of modern science is the scientific method, a conceptual framework for generating knowledge and understanding from observations using theoretical

models. Central to the method is the notion of falsifiability: a scientific theory should produce predictions that can be directly or indirectly falsified by empirical observations [14]. In simple terms, the scientific method can be summarized as the process in figure 1.

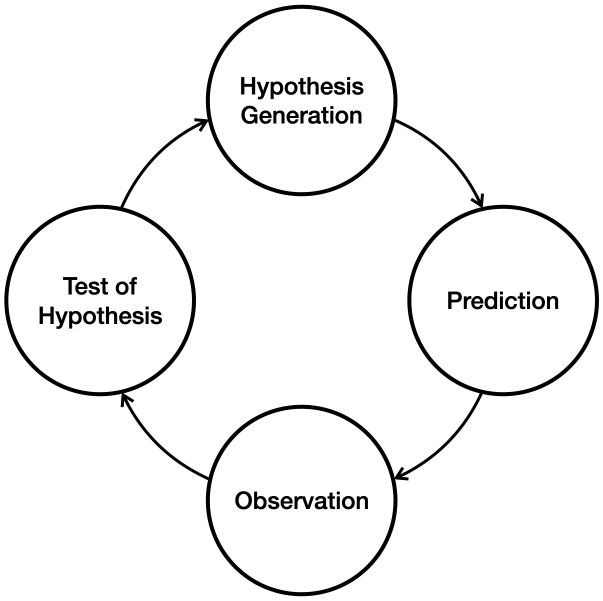

Figure 1: Conceptual algorithmic loop of the scientific process, as outlined by the figure in [15, Ch. 1.1].

In applying the process, one starts with a hypothesis about some phenomena. For example, "this coin always lands heads-up" is a valid hypothesis on the bias of a particular coin. A hypothesis which is direct in its prediction and has a low level of abstraction.

To test the hypothesis, one observes data which is testable against the predictions of a hypothesis. In the case of the coin hypothesis, the scientist would perform an experiment by flipping the coin repeatedly. Depending on the hypothesis, this might be done using observational studies or experiments.

The explanatory power of a hypothesis is typically tested on the basis of significance. In simple terms, significant observations can be defined as a set of observations that significantly deviate from the hypothesis predictions [16, Ch. 10]. If the hypothesis fails to reach the set level of significance, typically evaluated using probability theory, they are discarded as *falsified* and the final stage of the loop is completed.

In our example, the coin is observed to land tails-up at least once, and the hypothesis is thus falsified. To repeat the process, we should generate a new hypothesis but immediately encounter an issue. Unlike the other parts of the

process, there is no structured approach from which a hypothesis is generated [10, Ch. 4].

Fundamentally, the scientific method can only falsify incorrect theories. It does not concern itself with statements on what is true. This imbalance is closely related to the challenge of hypothesis generation. Even though the process can not tell the researcher what is true, knowledge of what is false can guide further developments, as indicated by the example of Pauli in section 1.1.

In John Skilling's paper "Classic maximum entropy" [17], he outlines the relation between special cases and general theories. If a phenomenon can be expressed in terms of a general theory, then the theory will apply to all cases [17], any special case will be covered by the general theory but may have a simpler form. If a viable solution can be identified for a sufficient number of special cases, one might have enough knowledge to infer the general theory, potentially fully constraining it [17].

His reasoning hints at a process in which properties of a theory can be distilled in a step-wise fashion by the relative success of constituent components. If the underlying theories of nature have a finite complexity, the process should be viable, given that the essential elements can be identified.

1.4 Epicycles of planetary orbits

Historical models of the world have been discarded in favour of new models for different reasons. One good example is that of epicycles, a historical description of planetary orbits [18]. The orbits were described by the sum of rotating vectors with frequencies of multiples of the base rotation [19]. This construct is conceptually equivalent to a Fourier series [20]:

$$f(t) = \sum_{n=0}^{N} c_n e^{i2\pi nt/P}. \tag{1}$$

Where t is time, c_n are coefficient of index n and P is the period. Fourier series are known to trace a periodic curve to arbitrary precision, given a sufficient number of coefficients [19]. The modern description using the elliptical orbits of Kepler superseded the epicycle model, called the Keplarian view [18]. A Kepler orbit can be expressed in the eccentric anomaly as [18]:

$$\mathbf{r} = \begin{pmatrix} a\cos E \\ b\sin E \end{pmatrix}. \tag{2}$$

Where a, b are the semi-major and semi-minor axes of the orbits, respectively. E denotes the eccentric anomaly, defined as the angle from the semi-major axis from the ellipse's centre. By setting x and y, in the typical sense of orthogonal

coordinates, to correspond to the real and imaginary components of $f(t)$ the ellipse can be described as a Fourier series in the eccentric anomaly:

$$f(t) = \frac{a-b}{2}e^{-iE} + \frac{a+b}{2}e^{iE}. \tag{3}$$

To transfer the central description to a focal point, a constant $c = \sqrt{a^2 - b^2}$ can be subtracted from the x component. Thus, the shape of the ellipse is fully described by two parameters, a, b or a, e. This description holds for describing an object orbiting a focal point in the ellipse, as is approximately the case for the sun and the Earth. The historical epicycle model used the geocentric perspective instead of the heliocentric one [18]. In the geocentric frame, the elliptical orbit of the moon can easily be parameterized, but Mars' orbit will require new parameters to explain the shape:

$$\bar{r}_{mars} = \frac{a_e - b_e}{2}e^{-iE_e} + \frac{a_e + b_e}{2}e^{iE_e} - \frac{a_m - b_m}{2}e^{-iE_m} - \frac{a_m + b_m}{2}e^{iE_m}. \tag{4}$$

Where the index e, m denotes Earth and Mars. However, this equation does not encapsulate the different time dependencies of Mars and Earth. Time dependence is encoded in E through Kepler's equation [21]:

$$\frac{2\pi}{P_o}(t - T_p) = E - e\sin E. \tag{5}$$

Where E is the eccentric anomaly of each celestial body, P_o is the orbital period, e is the eccentricity, T_p is the time at periapsis and the left-hand side is commonly referred to as the mean anomaly. Kepler's equation forms a non-trivial relationship between the eccentric anomaly of Earth and Mars, which complicates the description of the orbits. Thus, the shape of Mars' orbit requires the formalization of the trajectory of each body and raises the number of parameters to eight. Both models and their parameters are summarized in table 1. They constitute a simplified description of the orbit and do not include parameters that explain each orbit's orientation and relative position, which would be required in a full analysis.

Keplerian	Epicycle
$\mathbf{r}(t) = \begin{pmatrix} a\cos E(t) \\ b\sin E(t) \end{pmatrix}$	$f(t) = \sum_{n=0}^{N} c_n e^{inE(t)}$
a, b, P_o, T_p	P_o, T_p, $\{c_n\}$

Table 1: Summary of the base model and the corresponding number of parameters. a is the body's semi-major axis, b is the semi-minor axis, P_o is the orbital period and T_p is the time at periapsis. For the epicycle model, the additional $\{c_n\}$ is the set of selected coefficients, fitting the orbit up to order N.

To visualize this in an epicycle model, one may simulate the orbits of a simplified system and identify the Fourier series coefficients. Assume a system of three bodies: Mars, Earth and the Moon. Earth and Mars orbit around the same focal point at the origin, a hypothetical sun, and the moon orbits Earth in its focal point. Let all orbits have their periapses aligned along the x-axis and be initialized at periapsis ($T = 0$), see figure 2. The parameters used were: $a_e, a_{mo}, a_{ma} = 1.496E11, 2.279E11, 3.844E8$ $[m]$ and $e_e, e_{mo}, e_{ma} = 0.0167086, 0.0934, 0$, based on data from NASA Planetary Fact Sheets [22]. To maintain continuity in the analysis, the orbital period was simplified to 30, 360 and 720 days for the moon, Earth and Mars, respectively.

Performing a Fourier analysis in the angular domain yields the results in figure 3, using an angular period of $2\pi n_{rot}$ where n_{rot} is the number of local orbits per total orbit. Both approaches use the same eccentric anomaly as the trajectory parameter but differ in the number of coefficients with large magnitudes. Explaining the orbit of Mars in the geocentric frame requires more coefficients than in the heliocentric frame. The inverse is true of the moon's orbit in the heliocentric frame since it is orbiting the Earth.

As shown in figure 3, the correct central body description leads to significantly fewer relevant terms since most components are more than ten orders of magnitude smaller than the principal components. Given the eccentric anomaly model, the number of parameters in the heliocentric epicyclical view of orbits is not significantly different from that of the Keplerian model. It includes the constant term, which can be inferred from the first-order terms, as shown in equation 3.

We classify the heliocentric model as less complex than the geocentric model since it has fewer relevant terms in the Fourier series. However, the lunar orbit can be considered complex from the perspective of heliocentrism. The data is unchanged, only observed in a different reference frame. As such, the data should be regarded as *factual* but not *objective*. Since all models explain the data well, none of them can be discarded on the grounds of falsification. Rather, favouring the Keplerian view on the movement of celestial bodies is *not* data-driven but due to other considerations.

1.5 Theory considerations beyond data

Historically, certain epistemological principles have been introduced and used in the formation and valuation of theories. Theories in science are not only vetted on their ability to explain data but also on their complexity. Most theories in science have a simple written form, even ones explaining the movements of vast amounts of particles. If the laws of nature are complicated, it seems unlikely that science could formulate them in short equations. Among the most well-known of the principles is Occam's razor, typically quoted as [23]:

> "Entities should not be multiplied beyond necessity"

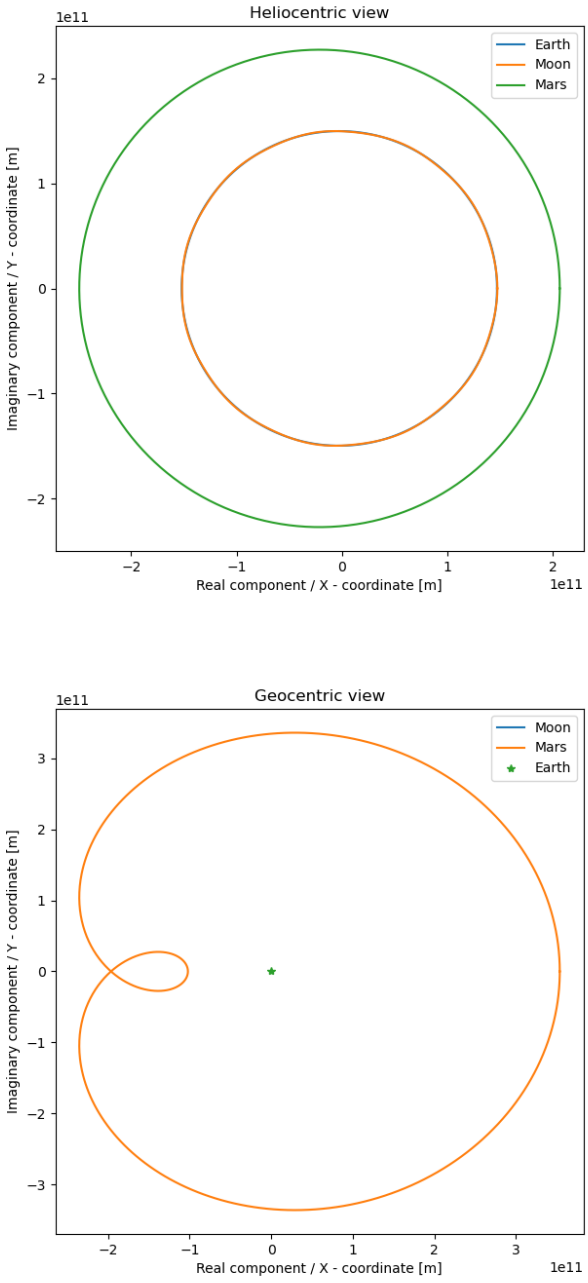

Figure 2: Simplified elliptical orbits of the bodies in heliocentric (top) and geocentric (bottom) view. The moon's orbit is present as very small fluctuations around the Earth's orbit.

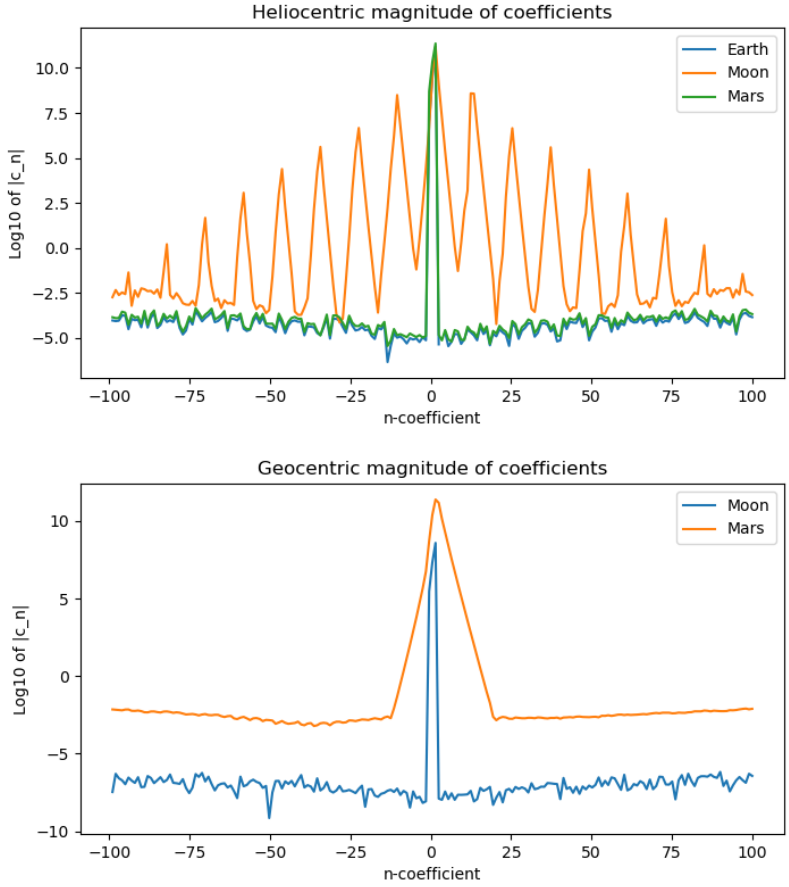

Figure 3: Fourier coefficients in the heliocentric (top) and geocentric (bottom) view expressed in the eccentric anomaly of each body.

In summary, the principle deals with the excess complexity of theories, postulating that simple explanations are preferable over more complicated ones, given sufficient accuracy. The wording *necessity* refers to the ability to explain the data. Thus, one should use any *sufficiently* complicated theory capable of explaining the data.

Given the example in chapter 1.4, the more complicated description, with regards to the number of parameters, of the orbits should be discarded according to Occham's razor as it introduces unnecessary complexity and detracts from the interpretability of the results.

1.5.1 Compression in physics

The application of Occam's razor can be seen as a removal of excess information in a theory. Thus, a shorter but *adequately* good explanation is preferred or, in a sense, a more compressed representation is favoured. Many branches of physics rely on differential equations to describe nature, one example being mechanics. Newton's law of acceleration is [24]:

$$\mathbf{F} = m\mathbf{a} = m\ddot{\mathbf{x}}.$$

Where $\mathbf{F}$ is the force acting on a particle and $\mathbf{a} = \ddot{\mathbf{x}}$ is the acceleration. The equation relates the change in velocity with the force acting upon the particle. To exemplify, for a constant force field in 1D, the evolution of the position can be solved via integration:

$$ma = C_F \rightarrow x = \frac{C_F}{2m}t^2 + v_0 t + x_0, \tag{6}$$

Where C_F is the constant force field, t is time from $t = 0$, v_0 is $\dot{x}$ at $t = 0$ and x_0 is x at $t = 0$. This equation describes the full trajectory of the particle given initial conditions and the force acting on the particle.

To reproduce the trajectory, one needs to store the initial conditions and the force descriptor. The force descriptor combined with the theory describes a set of potential paths the particle might take, and the initial conditions are used to select the specific path in the set.

Assume one has a particle with a trajectory with positions $\boldsymbol{y}$ at times $\boldsymbol{t}$. Given that Newtonian mechanics applies to the system in question, and one would be able to identify a description of the force and the initial conditions, the positions could be described using this information. Since the system description and $\boldsymbol{t}$:s entirely determine the $\boldsymbol{y}$:s, the user may ignore storing the $\boldsymbol{y}$. The data is then compressed into a representation containing the time data, the initial conditions and the force description, see figure 4.

If the description is minimal, then it is expected that all regularity of the data is encoded in the compressor/decompressor. What remains are the parts of data that define each trajectory. In typical use of physics, the decompression element is the differential equations describing the evolution of the state and the minimal description is the initial state. The compressor component of physics is rarely discussed but corresponds to the process of finding the initial conditions fitting the data.

In terms of Occam's razor, we prefer a theory that compresses the representation to a high degree but still offers an accurate reconstruction of the data. One may also consider that Occam's razor can be interpreted in a meta-compression setting: a single theory explaining two seemingly disconnected phenomena is preferable to two separate theories.

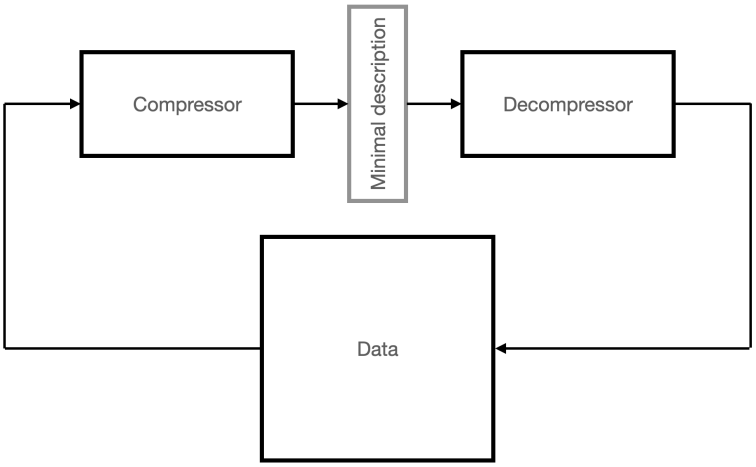

Figure 4: In a fully understood system the scientist should be able to take a general data set "Data" from the system and use the knowledge of the system to compress it to a "minimal description". The minimal description can be used with knowledge of the system to reconstruct the initial data set.

1.6 Machine learning

The above properties form a set of interesting considerations for theories, and in light of the size of the search space of hypotheses, any guidance could prove very helpful. To connect to the algorithmic component of the thesis, a brief introduction to machine learning and its application is needed.

Machine learning is fundamentally an automated approach to finding regularity in data [25]. In simple terms, machine learning seeks to minimize a measure referred to as the loss function, or empirical risk, across a data set by optimizing a set of model parameters [26, Ch. 2]. For a given problem, the learning process may generally modify two quantities and their inherent parameters: the loss function and the hypothesis class. The hypothesis, or model, is the structure the model will use to perform its task [26, Ch. 2], for instance, predicting a function output or labelling data. The empirical risk across a sample is calculated as [26, Ch. 3]:

$$L_E(H, \ell) = \frac{1}{n} \sum_{i=1}^{n} \ell(H(\boldsymbol{x}_i), y_i), \tag{7}$$

where L_E is the empirical risk, H is the hypothesis or predictor, evaluated, n is number of samples, $\boldsymbol{x}_i$ are the variables used to predict the output y_i and ℓ is the loss function. As should be clear, this framework is not significantly different from that of hypothesis testing. It should be noted that any application of machine learning has other practical considerations, for instance, an optimizer

for parametric models.

The No Free Lunch (NFL) theorem is closely related to the question of the structure of the hypothesis generation. In summary, the theorem states that no optimization or search algorithm can be efficient for all classes of problems, where efficient refers to any algorithm which performs better than a random search [27]. As a corollary, to have an efficient search, the algorithm needs to exploit the structure inherent to the problem at hand [27]. In the scientific framework, it is typically called an inductive bias or prior information.

The NFL theorem is fundamental and relates to hypothesis generation. The success of science and research indicates that humans have a structured approach to hypothesis generation. By creating an algorithm in this framework capable of encapsulating the structure discussed above, it should be possible to prune and explore the space of hypotheses efficiently.

1.7 Outline

This thesis will use a bottom-up approach to explore a possible path to the algorithmic approach of hypothesis generation. As explained in section 1.6, any efficient search requires a prior, or more specifically, an inductive bias. We will focus on three potential inductive biases through symmetry (section 3), minimum description length (section 2.4) and entropy (section 2.3), investigating the concepts, their relationship and connection to theory generation.

This theoretical investigation of the necessary concepts for algorithmic hypothesis generation culminates in the implementation of two algorithms. The algorithms will encapsulate the search for symmetries and automatic hypothesis generation for physics research.

2 Background

The introduction covers the basics of the most central concepts and their relevance. In order to formulate the criteria relevant to theory finding, we need to quantify prediction accuracy and theory complexity. Accuracy refers to the ability to model the data correctly, and complexity refers to a measure of how complicated a theory is.

2.1 Knowledge and probability

Central to evaluating hypotheses is probability theory, a logical approach to inference in a state of limited knowledge [15, Ch. 1]. At the core of the Bayesian perspective on probability (but not invalid or irrelevant in the frequentist) is Bayes theorem, a method to update one's knowledge about an object [16, Ch. 2]:

$$p(H|X) = \frac{p(X|H)p(H)}{\int p(X|H)p(H)dH}, \tag{8}$$

where $p(.)$ denotes a probability density, X is a random variable, H is a model, $p(H|X)$ is the posterior density, $p(X|H)$ is the likelihood function, $p(H)$ is the prior probability density. The denominator is the normalization factor, the evidence $p(X) = \int p(X|H)p(H)dH$, which is the likelihood marginalized across all models. In simple terms, the prior density is used to encode prior knowledge of the problem and is updated as data is observed, forming the posterior distribution.

A hypothesis is fundamentally a proposed theory about a phenomenon, leading to some testable prediction. Hypotheses with no free parameters are called simple hypotheses, and ones with parameters are called composite hypotheses [16, Ch. 10]. In the frequentist paradigm, a hypothesis is true or false, using a null to represent the assumed truth [15, Ch. 7]. The null hypothesis is the base prediction, yielding the baseline for evaluating an experiment [28, Ch. 7], for instance, how big a typical variation is and which measure to use.

Hypothesis testing, as described in the introduction, is commonly performed in the framework of probability theory. For hypothesis testing done on the basis of significance, one assigns a level of significance, a probability of observing an event in the critical region w given the truth of the null hypothesis H_0 [16, Ch. 10]:

$$P(X \in w|H_0) = \alpha. \tag{9}$$

Where the critical region is the region in which an observation would reject the null hypothesis. For example, if one has a null hypothesis that the data follows a standard Gaussian distribution $f(x) = 1/\sqrt{2\pi}e^{-x^2/2}$ [16, Ch. 4],

assume the acceptance region is a double-sided interval $[-2, 2]$ corresponding to a significance of $\approx 4.2\%$.

Another important significance test is that of likelihood ratio, typically the most powerful test of simple hypotheses [16, Ch. 10]. The likelihood is defined as [16, Ch. 10]:

$$L(X|\boldsymbol{\theta}) = p(X|\boldsymbol{\theta}) = \prod_i p(x_i|\boldsymbol{\theta}), \tag{10}$$

where $p(x_i|\boldsymbol{\theta})$ is the probability density of event x_i given the parameters $\boldsymbol{\theta}$ and $p(X|\boldsymbol{\theta})$ is the likelihood in probability density notation. The likelihood function is a function of the data conditional on $\boldsymbol{\theta}$, the hypothesis parameters. The likelihood ratio is related to the Bayesian posterior distribution ratio, which has the name Bayes factor and encompasses both the observed and prior information [15, Ch. 11]:

$$BF = \frac{p(\boldsymbol{\theta}_1|X)}{p(\boldsymbol{\theta}_2|X)} = \frac{p(X|\boldsymbol{\theta}_1)}{p(X|\boldsymbol{\theta}_2)} \frac{p(\boldsymbol{\theta}_1)}{p(\boldsymbol{\theta}_2)}, \tag{11}$$

where p denotes the probability density over the parameters $\boldsymbol{\theta}_i$. The evidence, $p(X)$, from equation 8 is marginalized over $\boldsymbol{\theta}$. As such, it becomes model-independent and cancels in the ratio. The Bayes factor also includes a penalization of high-dimensional models, referred to as Occam's factor [15, Ch. 3]. It arises naturally in Bayesian model selection and is related to the ratio of the prior and parametric volumes of models [15, Ch. 3].

This parsimony (preference for smaller models) is inherent to the structure and indicates structural limits on the complexity of models. This seems closely related to the structural search of section 1.6, which requires prior knowledge to guide an efficient search, limiting the complexity of the search. The above chapter introduced the probabilistic framework, forming the basis of hypothesis testing and comparison. The concept of complexity is, however, best formalized in computer science.

2.2 Computability

In computer science, algorithms are structured processes used to perform computational work using computers. Central to any algorithm used in practice is computability.

Computability of a function refers to its ability to be calculated or processed on a computer [29, Ch. 1.7] and is a fundamental property of a function. Computers are conceptually functions which operate on information and produce an output and can be thought of as function approximators. The conceptual framework of computing is evaluated using Turing machines, which are deterministic machines operating on a tape of data using a finite set of rules of states [29, Ch. 1.7].

To compute a function effectively, the computer needs to represent the problem in a way that it can calculate. The function needs to have the right input and output domain, accept all inputs and *halt*, meaning the calculation stops at some point. A computational function with these properties is called recursive and allows a computer to approximate a function to arbitrary precision [29, Ch. 1.7]. As proven by Gödel, there exist functions for which no Turing machine may halt in finite time, commonly referred to as the halting problem [29, Ch. 1.7] and is an example of uncomputable functions.

The generality of Turing machines makes the notion of computability not only a statement on whether a specific implementation may calculate a function but also a statement on whether any computer can approximate it. Thus, the importance of the statement on computability should be evident in the context of algorithmic approaches.

2.3 Complexity and entropy

Complexity vaguely refers to a measure of how complex an object is but is not a well-defined metric. One of the simplest and most natural measures of complexity is simply the number of inherent parts, for instance, the number of particles in a system or the number of players on a football field. Closely related is dimensionality; the number of characteristic "axes" along with an object varies. One instance of this is complex numbers with two dimensions: one real and one imaginary. Another is the number of parameters in the parameter space of a function. While these measures are useful, a more formalized, general description of complexity exists. Named after A. N. Kolmogorov [29, Ch. 1.9], Kolmogorov complexity is a measure of absolute information and is one of the key concepts of Algorithmic Information Theory (AIT).

Conceptually, the Kolmogorov complexity $C(x)$ is the length of the theoretically shortest program in bits, which generates the output x [29, Ch. 1.8]. The Kolmogorov complexity is an uncomputable quantity, meaning that it is impossible to determine if the representation one has is the minimal one [30]. The quantity also connects to randomness through the Martin-Löf definition, where a string is random if the length of a string is close to its Kolmogorov complexity [29, Ch. 2.4].

The prefix complexity is an extension of the Kolmogorov complexity. It is prefix-code (or rather, a prefix-free code), a code in which none of the compressed representations is an initial substring of any other representation [29, Ch. 3]. In terms of the conditional Kolmogorov complexity, the prefix complexity is given as [29, Ch. 3, p. 202]:

$$K(x|y) = C_{\psi_0}(x|y). \tag{12}$$

Where $K(x|y)$ is the prefix complexity of the string x given knowledge of y, the Kolmogorov complexity of x, and C_{ψ_0} is the Kolmogorov complexity con-

ditional on a theoretically optimal set of prefixes ψ_0. This measure has several advantages over the Kolmogorov complexity in that it is, among other things, subadditive ($K(x|y) \leq K(x) + K(y) + O(1)$) [29, Ch. 3.1]. Unlike Kolmogorov complexity, the prefix complexity allows for unambiguous recovery of the complete representation [29, Ch. 1.11]. The two complexities are asymptotically equal up to a constant [29, Ch. 3]:

$$C(x) = K(x|C(x)) + O(1). \tag{13}$$

Where $C(x)$ is the Kolmogorov complexity of x, $K(x|C(x))$ is the prefix complexity of x given knowledge of the length of the minimal compressed representation and $O(1)$ is a constant *ordo* 1, meaning it is independent of the object x.

2.3.1 Entropy

The second important measure of complexity is entropy, often referred to as a measure of "disorder" [31] or information [32]. There are several related definitions of entropy, the most important from the perspective of complexity being Shannon entropy. His definition relies on the probability of observed, or observable, events i as [32]:

$$H_s(X) = K_H \sum_{i=0}^{N} p(x_i) \log \frac{1}{p(x_i)}.$$

Where K_H is a unit-dependent constant and p is the probability mass distribution of X with $x_i \in X$, N represents the number of draws or observations. For Shannon entropy, the theoretically maximum entropy is reached when all observable events x_i, x_j have uniform probability $p(x_i) = p(x_j), \forall i, j$ with value $K \ln N$ [33]. In this state, there is maximum uncertainty of the outcomes and the entropy is analogous to Boltzmann's entropy [34]:

$$S_b = k_B \ln \Omega \tag{14}$$

Where k_B is Boltzmann's constant, Ω is the number of valid permutations of the system. The analogy refers to both being equal up to an additive constant, dependent on the base of the logarithm. The conditional Shannon entropy is [29, Ch. 1.11]:

$$H_s(X|Y) = -\sum_{i=1,j=1}^{M,N} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(y_j)} \tag{15}$$

Where $p(x_i, y_j)$ is the joint probability distribution of state x_i, y_j, M is the i:th highest numbered state and N is the j:th highest numbered state. The

conditional entropy is a measure of information given knowledge of a part of the system. It can be used to form descriptions of systems through algorithmic entropy, sometimes also referred to as physical entropy, defined as [29, Ch. 8.6]:

$$S_a = K(x) + H_s(y|x), \tag{16}$$

where $K(x)$ is the prefix complexity of x, a descriptor of the system, and $H(y|x)$ is the Shannon entropy of the state y conditioned on x. Conceptually the first term corresponds to the most compact description of the knowledge of the system as of now. The second one corresponds to a lack of knowledge about the state, given what is known in x [35]. This is a combined information statement on both the system and the state of the system.

2.4 Compression of data

Compression, as outlined in the introduction, has a clear connection to the understanding of phenomena. Fundamentally, compression relies on regularity in data to create smaller representations of it [29, Ch. 1.8]. An example of a practical compression scheme is Huffman codes, where the algorithm attempts to code recurring sequences with short code representations and rare sequences using longer representations [36].

Compressibility indicates that the system contains redundant information which is not necessary for reconstruction. In contrast, an incompressible object does not contain redundant information and thus requires the full length to be described. In terms of prefix complexity, incompressibility means that the object's current representation size is equal to or smaller than its prefix complexity [29, Ch. 3.3]. As mentioned in section 2.3, if the data can not be compressed, it is random.

Most binary strings of a finite length can be considered to be incompressible in the Martin-Löf sense. Given a binary string, the relative frequency of the strings that can be compressed effectively is $f_{compr} \leq 2^{-k}$, where k is the number of bits that have been compressed [37, Ch. 1.2], commonly referred to as k-compressibility. The conclusion from above is that randomness is the norm, while regularity and compressibility are rare properties.

Reflecting on the introductory discussion on compression in physics, section 1.5.1, there exists a correspondence between the algorithmic and the physical concepts of compression. If one can express the structure of physical theories in terms of information theory, then the complexity of a physical theory can be formalized.

2.4.1 Minimum description length

The information-theoretic compression framework for theories is formalized in the concept of Minimum Description Length (MDL). A theory or model of

physics can be used as a basis for the regularity in data. Thus, it should be able to be used to compress it.

The minimum description length (MDL) is a concept in information processing, similar to Occam's razor in epistemology [29, Ch. 5.4]. There are several common but distinct definitions of MDL with several conceptual frameworks denoted the *ideal* MDL. Use of the ideal MDL principle is the selection of the hypothesis H which minimizes the quantity [29, Ch. 5.4]:

$$H_{iMDL} = \arg\min_{H} : K(H) + K(D|H), \tag{17}$$

where $K(H)$ is the prefix complexity of the hypothesis and $K(D|H)$ is the prefix complexity of the observed data D, given knowledge of the hypothesis H. The quantity is measured in bits and is an objective measure of the information contained in the system. The prefix complexity is uncomputable, but upper semicomputable [29, Ch. 3.4], meaning it can not be calculated exactly but an approximate upper bound can be estimated. The above limitation directly shows the main issues with applying ideal MDL.

In practice, the MDL approach requires the use of approximations to be useful [37, Ch. 1.5]. For instance, within a hypothesis class one may have a program that allows for the calculation of any polynomial, given a set of parameters. Since the complexity of any polynomial can be expressed in terms of its parameters and the program complexity, the difference in the model complexity is the difference in parameter complexity, allowing for simple comparison.

To quantify a solution one needs to describe the model complexity as well as the deviations between the data and the model prediction, commonly referred to as a two-part code, or crude MDL [29, pp. 382–384]. As an equation, the principle can be expressed as [37, Ch. 1.5]:

$$H_{cMDL} = \arg\min_{H} : L(H) + L(D|H), \tag{18}$$

where H is the hypothesis, or model, in our hypothesis class, H_{MDL} is the optimal model, $L(H)$ is the description length of the model and $L(D|H)$ is the description length of the data, given the model. Whereas the ideal MDL specifies the quantities through the prefix complexity, the crude MDL requires an intelligent formulation of the description lengths.

For probabilistic hypotheses, such as a polynomial with an added Gaussian noise term, the description length of the data given a hypothesis can be effectively described by the negative log-likelihood [37, pp. 15–16].

$$L_d(D|H) = -\log_2 p(D|H), \tag{19}$$

where L_d is the description length in bits (or nats if the natural logarithm is used), p is the likelihood of the data D given model H. How to describe the model is less obvious in the two-part MDL framework since the model length is implementation-dependent. In practical terms, one may describe the model using a lexicon of functions, variables, constants and parameters, as in [38]. With such a set of operators and operands, the complexity per component can be expressed as:

$$L_d(H) = d \log_2 N, \tag{20}$$

where d is the number of components and N is the number of entries in the set. Thus, one needs a sequence of length d, each with the data cost $\log_2 N$. Parameters come with the additional requirement that their value needs to be specified. One approach [37, p. 137] is using a discretized domain, as is typical in computing. Each parameter can then be described by a precision d, equal to the informational content in the correct logarithm (for example, a Float32 is 32 bits/in $\log_2$).

In the MDL framework, there exists a natural "prior" complexity, which is the effective description size of the free data [37, Chap 14.2]. If no preference or regularity is given, then each point is a draw of the uniform distribution over the allowable values [37, Ch. 14.2]. In practice, this can also be conceptualized as the $n \log_2 m$, with n data points, each of m complexity [37, Chap 14.2]. For instance, using 32 bits, one may represent 2^{32} states. If these states are mapped uniformly over the interval $[a, b]$, then each step is $\frac{b-a}{2^{32}}$, allowing a stepwise representation from a to b. This concept quantifies the base complexity of data and limits how complex a model can be used for a given amount of data [37, Ch. 14.2].

As mentioned in section 2.3, probability theory and algorithmic information theory share a close connection through Shannon's entropy, asymptotically approaching the average prefix complexity over strings all the strings x of length $l(x) = n$ [29, Ch. 2.8]:

$$H_s(x)|_{l(x)=n} \sim \sum_{l(x)=n} p(x)K(x), \tag{21}$$

where $p(x)$ is the probability mass function over x and $K(x)$ is the prefix complexity. The algorithmic equivalent to the uniform prior on binary strings is the universal distribution $m(x)$. The coding theorem [29, p. 273] states:

$$\log \frac{1}{m(x)} = \log \frac{1}{Q_U(x)} = K(x), \tag{22}$$

for all x up to an additive constant, where $K(x)$ is the prefix complexity and $Q_U(x)$ is the universal a priori probability [29, p. 272]:

$$Q_U(x) = \sum_{U(P_c)=x} 2^{-l(P_c)}. \tag{23}$$

The U subscript denotes a reference prefix machine, the conceptual computer running the program, and $l(P_c)$ is the length of a program P_c, in bits, which calculates the output x running on U. In other words, the sum is the sum over all programs P_c which will output the string x, and the contribution for each program which will output x is $2^{-l(p)}$. More concisely, the universal distribution is the distribution over all outputs x of a computer running a random program P_c [39]. As can be noted from the coding theorem, if the string has a short Kolmogorov complexity, it also has a high probability of being output.

With a prior and a connection to probability theory, there exist similarities with the Bayesian approach. Under certain conditions, MDL and Bayesian hypothesis selection will approach a similar hypothesis [29, Ch. 5.4]. The maximum a posteriori hypothesis of the Bayesian approach is generated through:

$$\begin{aligned} \arg\max_H : p(H|D) &= \arg\max_H : \frac{\Pi_{i=1}^N p(d_i|H)}{p(D)} p(H) \\ &\rightarrow \arg\min_H : -\log p(H) - \sum_{i=1}^N \log p(d_i|H). \end{aligned} \tag{24}$$

in the $N \rightarrow \infty$ limit, where d_i are the individual data points observed in D. Comparing this to the ideal MDL description of the data, as in equation 17, there is a strong structural similarity between both forms, but one is measured in probability and one in complexity. Under a set of non-trivial constraints, the probability can be expressed in terms of complexity via the coding theorem [29, Ch. 5.4]. Most importantly, the likelihood can be expressed in terms of complexity as [29, Ch. 5.4]:

$$-\log_2 p(D|H) = K(D|H), \tag{25}$$

up to a constant $K(p(.|H)) + O(1)$, where $p(.|H)$ is the general probability distribution given a hypothesis H and K is the prefix complexity. Following theorem 5.4.1 of [29, Ch. 5.4, pp. 389-390], the log-ratio of likelihoods of the MDL and Bayesian approach has an upper limit related to the prefix complexity of the probability distribution and prior distribution, proportional to $K(p(.|H)) + K(p(H)) + O(1)$. In simple terms, if the posterior and prior distributions have low prefix complexities, both approaches should favour a similar hypothesis [29, Ch. 5.4], at least in the large n limit where the likelihood will dominate the selection.

2.5 Conceptual summary

This chapter briefly introduced the most essential and general concepts, primarily focusing on complexity. Probability theory is a coherent framework for expressing and updating knowledge of a system. Second, the concept of algorithmic complexity was introduced through Kolmogorov and prefix complexity, absolute measures of information. Both concepts connect through entropy, a measure of information using probabilities.

Using the formalized framework of algorithmic information theory, the application of compression in physics is done through MDL. A brief exploration of the literature shows how the MDL relates to the Bayesian model selection under some non-trivial conditions.

3 Algorithmic symmetry searching

As outlined in the previous section, the complexity of theories is an important and logically coherent part of theory formation. Another important consideration is symmetry, which is important since any physical law can be expressed in terms of their symmetries [40]. Symmetries also have a correspondence with conserved quantities, such as energy conservation, through Noether's theorem [41, p. 101]. Both properties could indicate that symmetries are, or could form, an important part of an inductive bias.

In the following section, we will explore symmetries, their properties and their connection to other properties. It also aims to give a background on key concepts required to implement an algorithmic approach to symmetry finding, which we implement in a machine learning algorithm.

3.1 Symmetries and groups

A symmetry is, simply explained, an operation upon a system that leaves the observed property invariant [42, Ch. 1]. Ubiquitous symmetries include mirror symmetry (left-right), rotational symmetry of spheres and circles, partial rotations in cubes, pentagons, and so forth. Symmetry typically refers to the invariance of a feature space (limbs, corners of geometrical figures, etc.) with regard to a geometrical transformation (translation, rotation, etc.), but is not limited to geometry.

As an example of the description above, imagine a free particle with three state variables (and one initial condition). The position is described by the function $x(x_0, v, m, t)$ where v is velocity, m is mass, t is time and x_0 is initial position. The subspace of the state space preserving position and energy are described by:

$$dx_0 + vdt + tdv = 0 \text{ for position and } \frac{v^2 dm}{2} + mvdv = 0 \text{ for energy,} \tag{26}$$

where the dx notation indicates an infinitesimal increment of the quantities. The observed position is invariant under any transformation of m, meaning they are symmetrical states in position. For energy to be preserved under transformations of m, a change in v needs to balance it to maintain symmetry.

In more mathematically rigorous terms, a symmetry is described in terms of group theory. A group is a set of elements with specific mathematical properties. The set needs a composition (some operation) which combines elements of the set, commonly denoted ab as the composition of a and b, both members of the set [42, pp. 6–10]. For all elements of the set, the composition should fulfil closure and associativity, have an identity element and have an inverse [42, pp. 6–10].

Expanding on the definition above, a group is a set of objects which together follow a set of mathematical properties. Closure refers to the property that the composition of two group elements is part of the group [42, pp. 6–10]. Associativity relates to the group member order in the composition, $ab = ba$ [42, pp. 6–10]. The identity element is an element which is the identity $eb = b = be$ and the inverse of elements $b^{-1}b = e$ [42, pp. 6–10].

Let a, b be states of a system. The symmetry transformation can then be defined in terms of the states as [42, pp. 80–81]:

$$S(a) = b \equiv a, \tag{27}$$

where $S()$ is a symmetry transformation of the states and $a \equiv b$ is the equivalence relation. In this case, the states $a \equiv b$ both leave the observed property of the system invariant and equivalence in this sense refers to the preservation of the observed property. If a and b are part of the input space of a function with the property that $f(a) = f(b)$, then the symmetry transformations $S : a \mapsto b$ form a group.

As an alternative description of the symmetry description of a system, De Haro et al. [43] provide an insightful definition where a physical system is divided into a state space $\mathcal{S}$, a quantity space $\mathcal{Q}$ and the dynamics space $\mathcal{D}$. A symmetry is then defined as an isomorphic map on the state space $\mathcal{S} \mapsto \mathcal{S}$ such that the observed important quantities $\mathcal{Q}$ are invariant [43]. Notably, the authors make a distinction between *bare theory* and *realizations*, where a realization of a theory as a given model may not adequately capture the full symmetry of the theory. An example of two isomorphic representations of one theory is the Heisenberg and Schrödinger pictures of a quantum system [43]. A representation of a theory does not need to be isomorphic with the theory and may be unable to capture or express a symmetry or may only allow trivial symmetries [43].

The importance of symmetries has been known for a long time, as exemplified by Curie's symmetry principle. Curie's original principle postulated that an asymmetry gives rise to phenomena [44]. J. Rosen expanded upon the base premise and asserts that the degree of symmetry in an effectively isolated system will not decrease as it evolves [42, pp. 146–150]. This principle is analogous to the entropy increase of processes and the author concludes that there exists a correspondence between symmetry and entropy such that a high entropy implies a high degree of symmetry [44].

3.1.1 Manifolds

Manifolds are simply explained lower dimensional objects which exist as a subspace in a higher dimensional space, forming a geometrical object. The existence and properties of manifolds are relevant for symmetry searching as well as for certain mechanical systems.

More precisely, manifolds are sets of points with a corresponding continuous, injective map from the manifold to the full space they reside in [41, Ch. 3.11]. The map is valid on a local neighbourhood around each point, meaning a small, open subset around each point [41, Ch. 3.11]. Each point on the manifold, in its local coordinates, is mapped into an equal or higher-dimensional point in the full space, and the set of points forms an object residing in the space. See figure 5 for two examples.

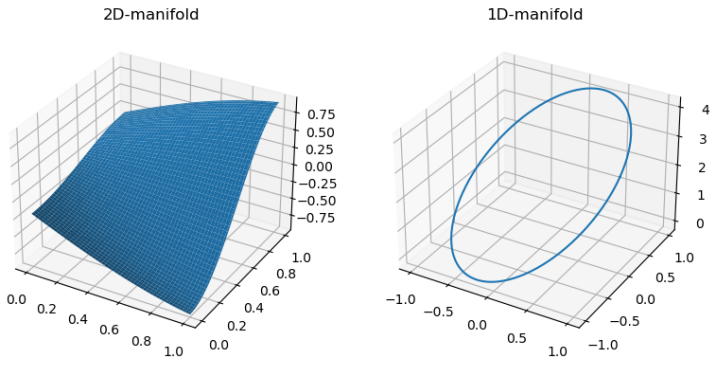

Figure 5: Examples of 1D and 2D manifolds embedded in 3D space. Units and axes are arbitrary.

A straight line is among the simplest manifolds and can be a manifold in 2D space. By setting a local coordinate $t \in R^1$ and local origin O_t, at some point in space $x, y \in R^2$ we may specify a map as $M : R^1 \mapsto R^2$:

$$M(t) = \boldsymbol{m}_{O_t} + \boldsymbol{k}_{dt} t, \tag{28}$$

where $\boldsymbol{m}_{O_t}$ is the position of the origin in manifold coordinates and $\boldsymbol{k}_{dt}$ are the coefficients relating dx/dt and dy/dt.

The manifold may require and is allowed to have a finite amount of maps to fully map all points in M in an injective manner [41, pp. 93–94]. Differentiable manifolds are ones in which the isomorphic mapping to and from the manifold is differentiable [45, pp. 20–22]. In practical terms, a hypersurface is an instance of a manifold where each point on the surface can be expressed in coordinates on the surface, mapping the connection to R^n via a function. A trivial manifold in R^n is R^n itself through the identity map [41, pp. 93–94].

Manifolds relate to symmetries as they geometrically describe the subspace of states which preserve the observable quantity. In the case of a free particle in 1D with a constant gravitational potential energy state is:

$$E_{tot} = gx + \frac{m\dot{x}^2}{2}. \tag{29}$$

Given a state $\{x, \dot{x}\}$, the parameters m, g can form a manifold, the subset of those parameters which leave the sum invariant. One can leave the parameters invariant and map the states that preserve the property and form a manifold.

3.2 Noether's theorem and the Lagrangian

Closely related to both manifolds and symmetries is Noether's theorem, which describes how conserved quantities and symmetries are connected. According to Noether's theorem, for each continuous symmetry of a Lagrangian system, there is a corresponding conserved quantity [41, p. 101]. The theorem deals with the action invariance due to infinitesimal changes.

The Lagrangian describes a mechanical system as $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) = T - U$ where $\mathbf{q}$ is the canonical position, $\dot{\mathbf{q}}$ is the canonical velocities t is the time, T is the kinetic energy and U is the potential energy. The system follows Hamilton's principle of least action, where the solution forms the extremal of [45, pp. 59–61]:

$$S(\mathbf{q}) = \int_{t_0}^{t_1} \mathcal{L} dt, \tag{30}$$

which is commonly referred to as the action of the system. This equation is solved using variational calculus, a method of finding minima or maxima using infinitesimal variations. Around an extremal point, the first derivative is by definition zero, and the first order Taylor expansions need to cancel [41, Ch. 4.1].

The process, as outlined by J. Schwichtenberg [41, Ch. 4.4], uses a small perturbation around the unknown optimum path $q(t) = a(t) + \epsilon(t)$. Expanding the Lagrangian in the perturbation and using the linearity of the integral, we set the first-order perturbation integral of the system equal to zero for all ϵ. By using the invariance of the endpoints of the action and rearranging, the ϵ can be factorized, giving the first order integral as [41, Ch. 4.4]:

$$0 = \int_{t_0}^{t_1} \epsilon(t) \left(\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} \right) dt. \tag{31}$$

Since the integral needs to be zero for all ϵ, the second factor in the integral needs to be zero, forming the Euler-Lagrange equations [45, pp. 59–61].

Noether's theorem, based on the above concepts, deals with the infinitesimal changes to the quantities, leaving the action unchanged. The action is invariant under the addition of a total time derivative to the Lagrangian $\mathcal{L}' = \mathcal{L} + dG/Dt$, yielding [41, pp. 101–105]:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\delta\mathbf{q} + G\right) = 0. \tag{32}$$

Thus, one can deduce that $\partial\mathcal{L}/\partial\dot{\mathbf{q}} + G$ is a constant quantity in time, which is the essence of Noether's theorem. Expanding on this, the classical formulation of Noether's theorem is given as [45, pp. 88–91]:

> *"If the system (M, L) admits the one-parameter group of diffeomorphisms h^s: $M \to M$, $s \in \mathbb{R}$, then the lagrangian (sic) system of equations corresponding to L has a first integral I: $TM \to \mathbb{R}$. In local coordinates q on M the integral I is written in the form*
>
> $$I(\mathbf{q}, \dot{\mathbf{q}}) = \frac{\partial L}{\partial \dot{\mathbf{q}}} \left.\frac{dh^s(\mathbf{q})}{ds}\right|_{s=0}."$$ (33)

Where M is a smooth manifold, L is the Lagrangian acting on the tangent bundle TM which is the manifold containing all tangent vectors of M, h^s is a diffeomorphic mapping $M \to M$ using the one parameter s such that $L(h_*\mathbf{v}) = L(\mathbf{v})$, or in other words keeps the Lagrangian invariant [45, pp. 88–91].

3.2.1 Lie algebra and groups

Noether's theorem is an application of what is known as the Lie algebra, which is a group of continuous symmetries formed by the generator X_L such that [46]:

$$X_L = \left.\frac{dh(s)}{ds}\right|_{s=0}, \tag{34}$$

where $h(s)$ are the admissible transformations of the parameter s. The transformations can be generated from the generator via incremental steps s/N repeatedly, with the limit of infinitesimal steps [46]:

$$h(s) = \lim_{N\to\infty}\left(I + \frac{s}{N}X_L\right)^N = e^{sX_L}. \tag{35}$$

The Lie algebra of the Lie group, G_L, is then the generators X_L such that $e^{sX_L} \in G_L$ for real-valued scalars s [46]. This is described by h^s in Noether's theorem but with the positional dependence of $\mathbf{q}$ explicitly included.

A Lie group can be described as both a group and a differentiable manifold such that the group operator maps onto the manifold itself [41, Ch. 3.4]. The self-mapping is differentiable in itself, commonly referred to as a diffeomorphism [41, Ch. 3.4].

3.3 Hamiltonian dynamics

The action integral, equation 30, was previously solved in the Lagrangian formalism but can also be solved in the Hamiltonian formalism. The Hamiltonian and Lagrangian formulations are equivalent, but the Hamiltonian consists of a set of $2n$ first-order equations instead of n second-order equations [45, Ch. 3].. The canonical formulation of Hamilton's equations is given as [45, p. 236]:

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}, \ \frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}. \tag{36}$$

Where $\mathbf{p}, \mathbf{q}$ are canonical coordinates such that $\mathbf{p} = md\mathbf{q}/dt$ and $\mathcal{H}$ is the Hamiltonian, $\mathcal{H} = T + U$, which is the kinetic and potential energy respectively. In the Hamiltonian view, each system is formulated as a manifold in the position and momentum space, commonly referred to as the phase space, with a set of important properties [45, Ch. 7]. Given a set of initial conditions, each position and momenta form a bijective mapping onto the time domain [45, p. 236].

If the Hamiltonian lacks an explicit time dependence, then this manifold corresponds to the energy-symmetrical states in phase space. This energy-preserving manifold is the object that will be the main target of the algorithm.

3.4 Deep learning

To learn the representation, the algorithm uses deep learning, a sub-field of machine learning utilizing deep neural networks as the objects to be optimized. Neural networks can be explained as a set of non-linear functions in a structure in which the data is processed by these functions and passed between each other in a structured manner using weighted channels [47, Ch. 6]. These networks can, for some classes and under some constraints, form universal function approximators [48][49].

Of special interest for the algorithm are autoencoders, deep learning networks which compress input data into a small latent space and then attempt to decompress the data into the original form [47, pp. 499–523]. The loss function attempts to minimize [47, pp. 499–523]:

$$\ell(\mathbf{x}, \mathbf{y}) = \ell(\mathbf{x}, g(f(\mathbf{x}))). \tag{37}$$

Where $\mathbf{x}$ is the input data, $f(\mathbf{x}) = \mathbf{z}$ is the action of the compressor and $g(\mathbf{z}) = \mathbf{y}$ is the action of the decompressor, the structure being graphically represented in figure 6, note the similarity to that of figure 4.

As expanded upon in section 2.4, compressing the data requires extracting the regularity. Using a latent space smaller than the input, the regular features will be stored in the compressor and decompressor, while the latent space stores the unique components. The compressor and decompressor do however need to

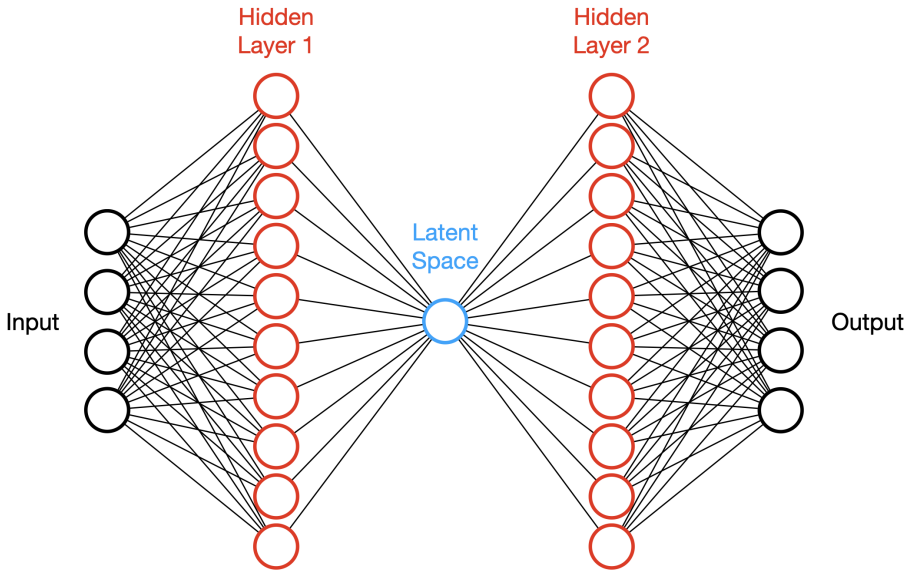

Figure 6: Visual structure of an autoencoder using fully connected, single layer, encoder and decoders. The data is passed along the lines between the nodes, first from each input node to each node in the hidden layer. The blue latent space is smaller than the input and output spaces, forcing regularity to be encoded in the hidden layers in order to accurately reconstruct the input, using the data in the latent space. Red nodes have non-linear activation functions.

have limited capacity to avoid effectively creating a lookup table of the data using the latent space as index [47, pp. 500–501].

3.4.1 Loss functions

As described in section 1.6, loss functions are scalar quantities summarizing a measure of success or deviation from the reference solution. The loss function is some function of the prediction and the correct outcome with a real, positive valued output, specified over the hypothesis and label (or output) spaces [26, pp. 26–27]. For use in the algorithm, the mean square error (MSE) is based on the square loss function as [26, Ch. 9.2]:

$$L_{MSE}(H) = \frac{1}{N}\sum_{i}^{N}(H(\boldsymbol{x}_i) - \boldsymbol{y}_i)^2, \tag{38}$$

where H is the predictor using variables $\boldsymbol{x}_i$ to make a prediction, compared to reference values $\boldsymbol{y}_i$ for a sequence of N values. Used with the autoencoder:

$$L_{MSE}(f, g) = \frac{1}{N} \sum_{i}^{N} (g(f(\boldsymbol{x}_i)) - \boldsymbol{x}_i)^2, \tag{39}$$

where f is the compressor, g is the decompressor and $\boldsymbol{x}_i$ is the input. The MSE also shares a close connection with the log-likelihood for a normally distributed probabilistic model, see section 4.6.3.

In deep learning, the choice of the loss function is related to the choice of output units [47, pp. 172–187]. In many cases, the loss function need not be strictly positive or zero but should have a large enough and nicely behaved gradient. [47, pp. 172–187]

When the autoencoder has learned to reconstruct data effectively, it has identified some regularity and can use that to compress it, see section 2.4. This regularity is related to the manifold mentioned in section 3.3. Thus, learning the regularity is closely connected to learning the symmetry of the data set.

3.5 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a type of search algorithm for high-dimensional spaces. To quantify the regularity, the algorithm aims to identify an approximate distribution of regularity-preserving data transformation. If the transformed data can be effectively reconstructed by the autoencoder, then the regularity is present in the transformed data as well. To analyze this property of the reconstruction, the algorithm needs a method to find the transformation parameters which preserve the manifold.

Suppose a measure of deviation from the symmetry can be expressed as a posterior distribution. Then MCMC is a numerical approach for efficiently sampling the posterior distribution of high dimensional models [15, p. 312]. In simple terms, the method "walks" through the parameter space of a problem, sampling points in regions with a relative frequency proportional to the PDF of the posterior [15, p. 314].

MCMC is an extension of Monte Carlo methods. In Monte Carlo, one uses random sampling to approximately solve problems. To demonstrate the basic principle, the mean of a continuous random variable can be written as [16, Chap. 2.4]:

$$\langle X \rangle = \int_X X p(X) dX, \tag{40}$$

where X is the random variable and $p(X)$ is the probability density function of the variable. Similarly, if one can express the posterior probability density $p(X|D, I)$ as a function of data D and prior I, a function $f(X)$ of the parameter space has the expectation value [15, Ch. 12.1]:

$$\langle f(X)\rangle = \int_X f(X)p(X|D,I)dX, \tag{41}$$

where $f(X)$ is some function on X. In the basic Monte Carlo framework, one may uniformly pick points across the parameter space of X. According to the weak law of large numbers, the above integral will be approximated by [15, Ch. 12.1]:

$$\langle f(X)\rangle = \liminf_n \frac{1}{n}\sum_{i=1}^{n} f(x_i)p(x_i|D,I), \tag{42}$$

where x_i are uniform draws from X. In this regime, it is likely that many samples are taken in regions of low probability, meaning most of the calculations won't provide significant contributions to the average. In MCMC, the samples are drawn from the target distribution through a random walk with the relative number of samples approximating the probability distribution [15, Ch. 12.2]. In the Metropolis-Hastings algorithm, the samples in the space of X are generated using a transition kernel, a conditional probability distribution $p(x_{i+1}|x_i)$ [15, Ch. 12.2], where x_{i+1} is the new position from x_i. The next step in the parameter space is generated using a proposal distribution, often symmetric, which is the accepted or rejected based upon the Metropolis ratio [15, Ch. 12.2]:

$$r = \frac{p(y|D,I)}{p(x_i|D,I)}\frac{q(x_i|y)}{q(y|x_i)}, \tag{43}$$

where y is the proposed position for x_{i+1} and $q(y|x_i)$ is the probability of the proposal distribution for point y given current position x_i. If r is greater than one, the new position is accepted, and if it is less than one, the new position has a r probability of being accepted [15, Ch. 12.2]. Note that if the proposal distribution is symmetric, for instance, Gaussian or uniform, the second factor is one. Then, only the ratio of posterior probabilities remains.

One weakness of the algorithm is that a considerable amount of time can be spent in low-probability sections of the domain, and the sampler can also be stuck in one of the probability modes [15, Ch. 12.4]. One way of avoiding this is parallel tempering in which multiple independent chains are initialized with different parameters (commonly denoted β_n) [15, Ch. 12.5]. The chains sample a rescaled posterior distribution and exchange positions with the neighbouring chain $n, n+1$ given a swap probability [15, Ch. 12.5]:

$$r_c = min\left\{1, \frac{p(x_{i,n+1}|D,\beta_n,I)p(x_{i,n}|D,\beta_{n+1},I)}{p(x_{i,n}|D,\beta_n,I)p(x_{i,n+1}|D,\beta_{n+1},I)}\right\}, \tag{44}$$

where $p(x_{i,n}|D,\beta_n,I)$ is the rescaled posterior distribution $\propto$ $p(x_i|D,I)e^{\beta_n \ln(p(D|X_i,I))}$, n indexes the chain number and i the position in the

chain. β is $1/\tau$ where τ is the so-called "temperature" of the chain. The lowest temperature corresponds to the original distribution, while the higher temperatures have flatter distributions. These chains will have a higher acceptance probability in regions of low probability of the original distribution. As can be seen in equation 44, the neighbouring chains (in temperature) can swap positions with some probability and will always swap if the joint probability of the swapped state is higher than the joint probability of the original state.

3.6 Symmetry search algorithm

Using the concepts and methods introduced above, we follow the approach of Yoh-ichi Mototake [50] to perform the symmetry search. His approach was partially implemented and evaluated for a single reference case.

Starting from a particle description, the phase space describes the canonical position and velocity from which the energy state of the particle can be calculated. Given a Hamiltonian system, the position in phase space at time t_i determines the position at t_{i+1}, in other words, the state evolution is governed by Hamilton's equations.

If the Hamiltonian lacks explicit time dependence, the system has a symmetry in total system energy: $\mathcal{H}(t_i) = \mathcal{H}(t_j) \forall i, j$, and the system forms a manifold in the phase space. The manifold representation can be learned by training a neural network to reconstruct the state from a compressed representation [50]. If the state to be reconstructed rests on the manifold, it will be reconstructed with a low error through the network. States outside the manifold will be "pulled" towards it [50]. See figure 7 for a visual representation.

Using the strategy outlined in [50], the training data is transformed and passed through the network to infer the knowledge stored in the network. Transformed data invariant under the reconstruction follows the same conservation laws as the original data. By using the reconstruction error as a measure, the approximate allowable transformations can be sampled efficiently using MCMC.

3.6.1 Manifold learning

An autoencoder is used to learn the dynamics of the system given two consecutive points in phase space [50]. The data used is on the form $\boldsymbol{x}_i = (q_i, p_i, q_i', p_i')^T$ where q_i, p_i is the phase space position and momentum at time t_i and q_i', p_i' is at time $t_i + \Delta t$ [50]. The accuracy of the reconstruction process is described by the mean square error [50]:

$$E(\mathbf{x}|\boldsymbol{\gamma}) = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - f_{AE}(\boldsymbol{x}_i|\boldsymbol{\phi}))^2. \tag{45}$$

Where N is the number of samples and $f_{AE}(\boldsymbol{x}_i|\boldsymbol{\phi})$ is the operation of the autoencoder on the data $\boldsymbol{x}_i$ after training given the hyperparameters $\boldsymbol{\phi}$. This

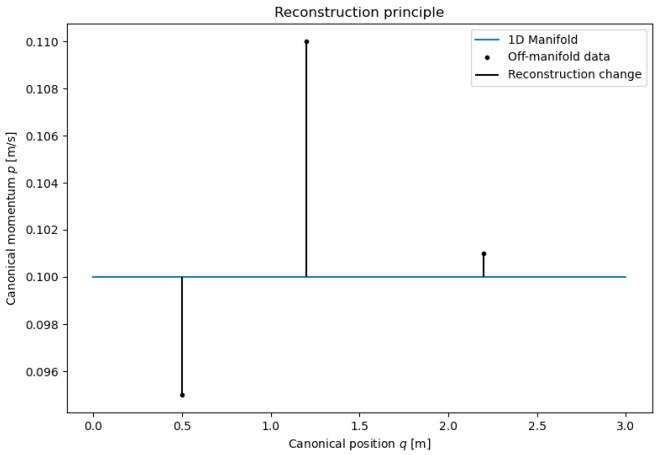

Figure 7: Principle of operation. The learned representation in 1D fails to reconstruct the state outside the manifold and "pulls" the data points towards the manifold. Inspired by figure 1 in [50].

hyperparameter reliance is implicit when referring to f or E below.

3.6.2 Network probing

To infer the conservation laws stored in the network, one aims to find the set of transformations which allow for efficient reconstruction [50]. In other words, the algorithm aims to find transformation parameters $\boldsymbol{a}$ with the transformation $\boldsymbol{x}'_i = X(\boldsymbol{x}_i|\boldsymbol{a})$ such that:

$$f_{AE}(\boldsymbol{x}'_i) - \boldsymbol{x}'_i \approx 0, \forall i. \tag{46}$$

Where f is the action of the autoencoder, X is some transformation of data which minimizes the reconstruction error of the neural network and $\boldsymbol{x}_i$ is the i:th data point and $\boldsymbol{x}'_i$ is the transformed i:th data point.

The transformations, $X(\cdot|\boldsymbol{a})$ is realized as a linear transformation on q, p plus a constant in each [50]. The transform is performed identically on the q, p and q', p' states. Thus, each transformation requires $(2d)^2+2d$ parameters where d is the dimensionality of the data [50]. In the case of 1D motion, the transformation becomes:

$$X(\boldsymbol{x}_i|\boldsymbol{a}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} q \\ p \\ q' \\ p' \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{01} \\ a_{02} \end{pmatrix}. \tag{47}$$

Where $\otimes$ denotes the Kronecker product and $\boldsymbol{a} = (a_{11}, a_{12}, ..., a_{02})$. This generalizes trivially to higher dimensions, with the linear transform being a square matrix of $4d^2$ parameters and the constant vector containing $2d$ parameters:

$$X(\boldsymbol{x}_i|\boldsymbol{a}_{4d^2+2d}) = \mathbf{I}_2 \otimes \mathbf{A}_{(2d)^2}\boldsymbol{x}_i + \boldsymbol{J}_{2,1} \otimes \mathbf{I}_2 \cdot \boldsymbol{B}_{2d}. \tag{48}$$

Where $J_{2,1}$ is a column vector of ones, I_2 is the identity matrix of shape 2×2, $\mathbf{A}_{(2d)^2}$ is a square matrix of the first $4d^2$ parameters and $\boldsymbol{B}_{2d}$ is a column vector of the remaining $2d$ parameters. The transformed data $X(\boldsymbol{x}_i|\boldsymbol{a})$ is processed by the autoencoder and a new error is calculated $E(X(\mathbf{x}|\boldsymbol{a})) = \frac{1}{N}\sum_{i=1}^{N}[X(\boldsymbol{x}_i|\boldsymbol{a}) - f_{AE}(X(\boldsymbol{x}_i|\boldsymbol{a}))]^2$ [50]. The size of this reconstruction error is a measure of the deviation from the underlying manifold learned from the data.

3.6.3 Analysis of the transformation space

Transforming the data requires $4d^2 + 2d$ parameters and quickly becomes a high-dimensional problem. To sample the parameter space, the paper uses REMCMC. Refer to chapter 3.5 for more details. The target (or posterior) distribution is modelled on the reconstruction error as [50]:

$$p(\boldsymbol{a}_m|\mathbf{x}) = \frac{1}{Z_m} e^{-\frac{N}{2\sigma_m^2}E(X(\mathbf{x}|\boldsymbol{a}_m))} \cdot p(\boldsymbol{a}_m). \tag{49}$$

Where N is the number of samples, $\mathbf{x}$ is the full set of training data, $\boldsymbol{a}_m$ is the m:th chain of proposed transformation parameters, Z_m is a chain dependent normalization constant, σ_m is the chain specific standard deviation and $p(\boldsymbol{a}_m)$ constitute prior knowledge on the distribution. In this application, the prior limits the solutions to an acceptable space and in out application was set to uniform for transformations such that the determinant of the transformation matrix $0.8 \leq \left|\mathbf{A}_{(2d)^2}\right| \leq 1.2$, and zero otherwise.

3.6.4 Inferring the symbolic relationship

The sampled set of parameters are sampling results from the distributions of the transformations which preserve [50]:

$$\begin{aligned} S_i = \Bigg\{ & q_t, p_t, q_{t+\Delta t}, p_{t+\Delta t} \Bigg| \ \mathcal{H}(q_t, p_t) = E_i, \\ & p_{t+\Delta t} = p_t - \frac{\partial \mathcal{H}(q_t, p_t)}{\partial q_t}, \ q_{t+\Delta t} = q_t + \frac{\partial \mathcal{H}(q_t, p_t)}{\partial p_t} \Bigg\}. \end{aligned} \tag{50}$$

In other words, the transformations to be sampled are the transformations which preserve the energy state of the particle. This set of transformations forms a differentiable manifold and infinitesimal transformations can be modelled as

the tangent space around the identity transformation [50]. Using the implicit functional form of the invariant transformations [50]:

$$\begin{aligned} f_1(\boldsymbol{a}(\boldsymbol{\theta})) &= 0, \\ f_2(\boldsymbol{a}(\boldsymbol{\theta})) &= 0, \\ &\vdots \\ f_{d-d_\theta}(\boldsymbol{a}(\boldsymbol{\theta})) &= 0. \end{aligned}$$

Where f_i is an unknown function and $(\boldsymbol{a}(\boldsymbol{\theta}))$ are the transformation parameters conditional on $\boldsymbol{\theta}$, a d_θ dimensional vector. The number of equations is determined by the difference between the dimensionality of the transform and the embedded manifold, $d - d_\theta$. If the Jacobian of this system of equations is non-singular at the identity transformation, the variables can be explained using a subset of $\boldsymbol{a}$ denoted $\boldsymbol{b}$ of length d_θ [50]. These relationships can be regressed as implicit equations [50]:

$$h_i(c_i, \boldsymbol{b}) = 0, \tag{51}$$

where i range from 1 to $d - d_\theta$ and $c_i \in \{\boldsymbol{a}\} - \{\boldsymbol{b}\}$ with $\{\boldsymbol{b}\} \subset \{\boldsymbol{a}\}$. In other words, we view the c_i as a function of the $\boldsymbol{b}$. The solution to the set of derivatives of the above implicit h_i equations: $\partial h_i(...)/\partial b_l = 0$, at the identity of the initial transformation $\mathbf{A}_{4d^2} = \mathbf{I}_{2d}$, gives a tangent vector for each b_l and defines the transformations as [50]:

$$\begin{pmatrix} \delta\mathbf{q_l} \\ \delta\mathbf{p_l} \end{pmatrix} = \epsilon \left. \frac{A(b_l)}{\partial b_l} \right|_{\mathbf{A}_{4d^2}=\mathbf{I}} \begin{pmatrix} \mathbf{q} \\ \mathbf{p}, \end{pmatrix} \tag{52}$$

with $\frac{\mathbf{A}(b_l)}{\partial b_l}$ corresponding to the matrix form of the solution on the set of differential equations $\partial h_i(...)/\partial b_l$. Thus, the method consists of identifying the Lie algebra. To find these relationships, the equations are generated as [50]:

$$h_k(c_k, b_1, ...b_{d_\theta}) = \sum_{s_0=0}^{d_b} \sum_{s_1=0}^{d_b} ... \sum_{s_{d_\theta}=0}^{d_b} \gamma_{s_0 s_1 ... s_{d_\theta}} \beta_{s_0 s_1 ... s_{d_\theta}} c_i^{s_0} b_1^{s_1} ... b_{d_\theta}^{s_{d_\theta}}. \tag{53}$$

Where γ is a selection factor, β is the scale factor of the polynomial, d_θ is the selected dimensionality of the manifold, d_b is the selected highest polynomial order, c_i is the explained variable and b_l are the explaining factors. These equations can be regressed using orthogonal distance regression [50], determining γ and d_θ in the process.

3.6.5 Implementation and results

We implemented the data generation, neural network and sampling of transformation parameters listed above using Python, TensorFlow (`tf`)[51] and TensorFlow Probability (`tfp`). The Hamiltonian description for the linear motion gives the differential equations:

$$\left\{\mathcal{H}(\mathbf{q},\mathbf{p}) = \frac{\mathbf{p^2}}{2m}\right\} \rightarrow \frac{\partial \mathbf{p}}{\partial t} = 0 \text{ and } \frac{\partial \mathbf{q}}{\partial t} = \frac{\mathbf{p}}{m}, \tag{54}$$

where $\mathbf{q}$ is the canonical position, $\mathbf{p}$ is the canonical momenta and m is particle mass. The equations can be trivially solved by hand and have no time scaling issue. We generated the training data based on the reference case from [50], with positions uniformly distributed $q = U(0, 1)$ in m, momentum fixed $p = 0.1$kgm/s and mass set to 1 kg.

We created an autoencoder using `tf.keras.Sequential`, basic structure based on Tensorflow introduction to autoencoders [52], trained using the generated data. Encoder and decoder using fully connected layers through `keras.layers.dense`. Parameters used for the autoencoder are listed in table 2 and see figure 6 for a visual representation.

Parameter	Linear motion
Network structure	4-10-1-10-4 [1)]
Activation function	RELU
Data points	1000 [1)]
Optimizer	ADAM
Loss function	MSE
Batch size	30
Epochs	400

Table 2: Neural network parameters. [1)] [50], appendix H, table III.

The transformation parameters were sampled with `tfp.mcmc.ReplicaExchangeMC` using a `tfp.mcmc.RandomWalkMetropolis` inner kernel with a normally distributed proposal function. The standard deviation was based on the method in [50] but our implementation was rescaled by a factor $\frac{1}{2}$:

$$\sigma_p = \begin{cases} \frac{C}{2} \text{ or,} \\ \frac{C}{2(N\sigma_l^{-2})^d} \text{ if } N\sigma_l^{-2} < 1, \end{cases} \tag{55}$$

where C is a constant and N is number of samples. σ_l is the l:th chain scaling factor [50]:

$$\sigma_l^{-2} = \sigma_{min}^{-2} \gamma^{l-1-L}, \tag{56}$$

where γ and σ_{min} are constants, l is the chain index and L is the total number of chains. The function parameters are listed in table 3.

Parameter	Burn-in	Samples	Chains	C	γ	d	σ_{min}
Value	50 000	50 000	8	0.03	1.9	0.7	5.4E-05

Table 3: Parameters used in REMCMC sampling algorithm.

Examples of the resulting sampling density for eight chains of the REMCMC can be seen in figure 8 with calculated autocorrelations in figure 9. The Gelman-Rubin test for 8 runs suggest convergence of a_{11}, a_{21}, a_{01} with a_{12} approaching convergence.

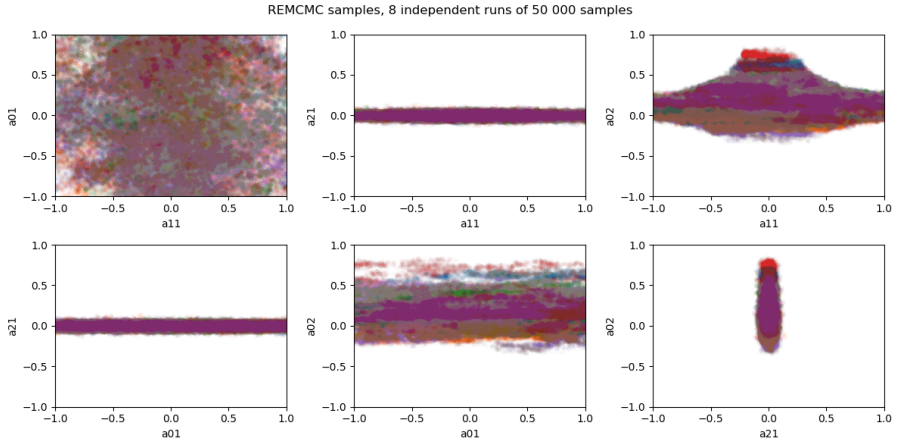

Figure 8: Accepted samples of transformation parameters using 50,000 samples, 50,000 burn-in samples with 30 parallel RE-chains and eight independent runs. Relations show the approximate distribution of allowable transformations. Three of the graphs specifically show that a_{21} is heavily constrained.

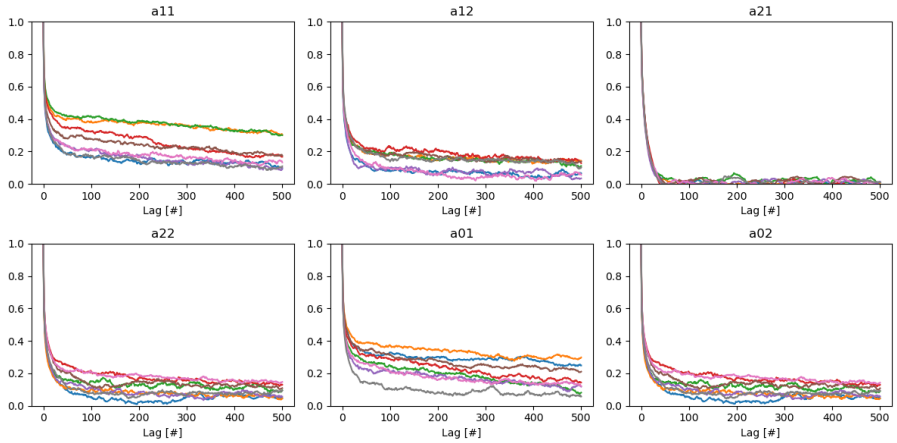

Figure 9: Autcorrelation factors for REMCMC chains using a max lag of 500 samples. Autocorrelations suggest chains have not fully converged.

After sampling, we could not regress the resulting data using Scipy `ODR`, which defaults to the zero solution. Our sampling data does resemble the reported results from the paper but generally have a larger spread. The important relationships in the transformation space 8 are the straight lines which indicate the invariance of a_{01} and a_{11}. The lines correspond to approximately conserved momentum which also preserves the energy state.

3.7 Discussion

In this chapter, we discussed and implemented an algorithm for inferring symmetries present in data of particle dynamics. To learn the underlying manifold, an autoencoder was trained to reconstruct the data of two successive points in time. The learned relationship was then probed by measuring the reconstruction error of the autoencoder for transformed states. By sampling the transformation parameters using REMCMC, a distribution of approximate symmetry transformations was acquired.

The algorithm is an interesting application of symmetry finding on realistic data by sampling the transformation parameters. While the MCMC chains of our implementation did not reach convergence according to the Gelman-Rubin tests, half the parameters are converged, with one more approaching convergence. The paper did implement a few limitations of the search space by inspection, which is likely to affect the chain convergence positively. Such limitations are also expected to improve the ODR regression step.

The resulting graphs in figure 8 show that the sampling parameter a_{21}, the second component of the linear translation is heavily constrained. This is the expected result as the autoencoder would learn the constant velocity motion

and should reject any transformations increasing the velocity. One limitation of the algorithm is that it requires a large number of observations to learn the manifold description of data. As such, it may prevent application in several fields where data is scarce.

In our implementation, we only applied the algorithm to a simple case, but the approach is possible to use on more advanced problems, as shown in the original paper [50]. A similar method was also used in AI Poincaré [53] called walk-pull, using the learned dynamics, MCMC sampling and principal component analysis to look for symmetries. There are several papers suggesting different methods for inspecting the symmetries in data [3][54][55][50][56], primarily using neural networks.

A notable non-neural approach is checking if data is symmetric under transformations of the Lie group (G_L, X_L), where G_L are the set of symmetry transformations and $\times$ is the group operator. Data with inherent symmetry are invariant to transformation - reverse transformation [57]. If Γ_ϵ is a symmetry transformation of the data, parameterized by ϵ, then [57]:

$$(\boldsymbol{x}, f(\boldsymbol{x})) \to \Gamma_\epsilon(\boldsymbol{x}, f(\boldsymbol{x})) = (\boldsymbol{x}', f'(\boldsymbol{x}')), \tag{57}$$

where the resultant new state has transformed parameters $\boldsymbol{x}'$ mapped to the transformed output $f'(\boldsymbol{x}')$. This can be transformed back into the original state with the inverse transform [57]:

$$\Gamma_{-\epsilon}(\boldsymbol{x}', f'(\boldsymbol{x}')) = \Gamma_{-\epsilon} \times \Gamma_\epsilon(x, f(x)) = (x, f(x)), \tag{58}$$

where $\Gamma_{-\epsilon}$ denotes the inverse transformation. To use this property in practice, one would transform the input data according to some symmetry transformation, generating $(\boldsymbol{x}', f'(\boldsymbol{x}'))$, and fit it using a method of the user's choice [57]. The function and data are then inversely transformed back to $(x, f(x))$, and if the function still approximates the data well, then the symmetry is inherent in the data [57].

There seems to be no consensus on how symmetries should be identified in the general case. Still, the principles above seem to be a pathway towards a general symmetry-finding approach, primarily for continuous symmetries.

4 Algorithmic hypothesis generation

A hypothesis generation algorithm is a machine learning algorithm which encompasses the full (or a vast) hypothesis space and has an efficient search mechanism to explore the potential solutions. As discussed in section 1.6, the search strategy is the fundamental focus for such an algorithm due to the NFL theorem. The algorithm also needs to solve the challenge of human readability to be classified as an acceptable generator. The subfield of generating human-interpretable equations is typically called symbolic regression and is growing in interest.

4.1 Symbolic regression

Symbolic regression is the machine learning framework for generating symbolic expressions from data. Symbolic expressions have several advantages over numerical models, most importantly interpretability [58]. To exemplify, a simple symbolic regressor could be constructed as a program which accepts data, x_i and y_i, and tests polynomial functions in ascending order:

$$\begin{aligned} y_i &= \theta_0, \\ y_i &= \theta_1 x_i + \theta_0, \\ &\vdots \end{aligned}$$

The best function, according to some metric, would then be returned to the user and used as a solution. This algorithm is likely inefficient for many cases and incorporates an implicit search strategy in limiting the hypotheses to polynomials and searching low-order polynomials first. Several more details need to be specified before the regressor can be implemented, including the optimisation process, multivariate search strategy, loss function and data structure.

As discussed in section 1.6, the NFL theorem requires the regressor to have an intelligent strategy for generating hypotheses since the search space is vast. The core function of symbolic regressors is the ability to intelligently select a solution and present it in an easily understandable form. As such, the algorithm also needs to include methods for evaluating properties such as complexity, if it is an important consideration.

The full problem of symbolic regression is likely NP-hard [58], supporting the problem's perceived difficulty. One successfully applied symbolic regressor [3] uses up to 19 operations and variables which gives a high branching factor for each selection. However, the number of valid combinations is significantly lower than the full search space [38].

The area is actively researched, and several approaches exist, including symmetry/conservation law search using learned relations [59] [53], tree searches [60]

Classical machine learning	Neural networks (NN)	Hybrid
Genetic algorithms	Language models	ML on NN
Monte Carlo tree search	Sequence-to-sequence	ML with NN support

Table 4: Rough characterization of symbolic regression methods with examples.

[38], transformers [2], genetic programming [61] as well as hybrid approaches [3]. The general approaches can broadly be split up into three domains. See table 4 for a summary.

Purely neural algorithms may provide a human interpretable sequence as an output, given a set of data. However, it may be difficult to understand the cause for selecting the model. This is primarily due to the "black box" nature of neural networks, making it hard to examine and explain the process [62]. In classical machine learning, a lack of understanding of causality or interpretability is usually of little concern due to the more explicit nature of their process. For hybrid approaches, the interpretability of the process depends on the implementation. See table 5 for examples of common algorithms.

Paper/Algorithm	Method	Note	Ref.
AI Feynman	Hybrid	Subproblems	[3]
AI Feynman 2.0	Hybrid	Subproblems	[63]
SymPhysLearn	MCTS		[60]
Exhaustive SR	Exhaustive search		[38]
SymbolicGPT	Transformer		[2]
PySR	Genetic alg.	Directed evo.	[64]
Nested MCTS	MCTS		[1] [65]

Table 5: Hypothesis generation for a selection of algorithms for symbolic regression.

In symbolic regression, deep learning is used to perform various tasks in the process, such as interpolation [3] and symmetry finding [59] [50]. They can also be used for end-to-end regression by using language models [66] or as representations for objects to provide non-symbolic answers.

4.1.1 Benchmarks

Symbolic regressors use several benchmarks to evaluate their performance. Some are specific to the tested algorithm/paper, but two often cited benchmarks are the Nguyen equations [67] as well as the Feynman benchmark [68] from the AI Feynman paper [3]. Both contain univariate and multivariate equations which are used to create reference data to perform symbolic regression on. The Nguyen equations have specified bounds on the amount of data, while the Feynman paper used a growing data set.

4.2 Bracket-free notation

A symbolic regressor outputs an equation as the solution to the problem. An equation may also be used as an intermediate state in the algorithm. In both cases, the notation must be unambiguous for the output to be meaningful. Typical equations, written in a human-readable form, can end up in ambiguous states without brackets. For example:

$$x/2 - x. \tag{59}$$

Without brackets, it is not obvious whether the equation is $-x/2$ or $x/(2-x)$, as is evident by several similar trending posts on social media. One solution to this is to use prefix or postfix notations, commonly referred to as Polish Notation and Reverse Polish Notation [69], where the operator is placed before or after the operands. In late operator prefix [69]:

$$x + y \equiv +\text{xy and } x + x/y \equiv +\text{x/xy}. \tag{60}$$

Reading it from left to right, first, the + requires two operands, the first being x and the second y. The second equation starts with an operator + with x as one operand and / as the second. Since / is an operator, it looks to the next two operands, with x as the numerator and y as the denominator. The tree can also be written in early operator postfix notation [69]:

$$x + y \equiv \text{xy}+ \text{ and } x + x/y \equiv \text{xxy/}+. \tag{61}$$

The postfix notation is easier to read backwards. The first equation starts with +, giving the implicit state $\_ + \_$ where the $\_$ is undetermined, requiring two operands to close the expression. By convention, the second operand (reading backwards) is y, and the first is x. This forms the complete expression $x + y$.

For the second equation, the initial step is the same. The + operator once again requires two operands, but the second operand after + is an operator, /. This operator forms a subexpression and we look for two operands for the implicit state $\_/\_$. The first following operand becomes the second operand in the expression. y comes first followed by x, forming x/y. This subexpression is now evaluated and can be entered as an operand into the first giving $\_ + x/y$. Repeating the process for the final x, one arrives at $x + x/y$. In both cases, knowledge of the number of operands per operator (two in this case) allows for unambiguous reading of the equation. [69] The need for this is clear when using operators with one operand:

$$e^x + x \equiv x \exp x + .$$

In this case, one needs to know that exp only requires one operand since assuming two operands will yield a nonsensical state. This extends to any operation and number of operands, such as an integral, which requires four operands: lower limit, upper limit, expression and integration variable. Like in the case of non-commuting operators (i.e. -, /, int), the order of operands is fixed and determines the function of the operands.

4.3 Composite loss metrics

As outlined in section 1.6, any machine learning algorithm requires a metric to rank the solutions coherently. In the symmetry searching algorithm, the MSE was used to quantify the manifold-preserving qualities of the transformation parameters. While the MSE captures the deviation from the target, it does not quantify any loss based on complexity. To extend the loss function to include secondary factors, one can introduce a regularisation term, penalising unwanted attributes of a solution.

Of the many choices which exist, we decided to focus on the Bayesian Information Criteria (BIC) and MDL metrics due to their ability to balance the complexity of a solution with accuracy. While the MDL has been covered earlier, the BIC is an interesting candidate loss function, sharing a close connection to the posterior probability. Fundamentally, the BIC is a combined loss metric based on the log-likelihood combined with a parameter-based regularisation, generated from Bayesian modelling considerations [70]:

$$BIC = k \ln n - 2 \ln L(D|\boldsymbol{\theta}), \tag{62}$$

where k is the number of parameters in the model, n is the number of observations in D and L is the likelihood function. The BIC comes from the maximisation of relative Kullback-Leibler (KL) information/divergence between the model evidence distribution $p(X|H)$ relative to the underlying data evidence [70]. The evidence is:

$$p(D|H) = \int p(D|H, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{63}$$

where $p(D|H, \boldsymbol{\theta})$ is the likelihood and $p(\boldsymbol{\theta})$ is the model prior on the parameters, see section 2.1. The evidence is the normalising constant in Bayes theorem, see section 2.1. This means that the most favourable model should predict a data distribution as similar to the underlying data distribution as possible.

Following the process of Stoica et al. [70], this is done by maximising the model evidence averaged across the true distribution as $\int p_R(D) \log p(D|H) dD$, where R denotes the real data distribution, $p(D|H)$ denotes model evidence distribution and D is the data [70]. In order to get the evidence of the data, the likelihood and prior of the model need to be integrated. Under a few constraints,

the likelihood can be expanded around the maximum likelihood parameters as $\hat{\theta}$ [70]:

$$p(D|H,\boldsymbol{\theta}) \approx p(D|H,\hat{\boldsymbol{\theta}})e^{-\frac{1}{2}(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})^T\hat{\mathbf{J}}(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta})}. \tag{64}$$

Where $\hat{\mathbf{J}}$ is the Fisher information matrix [70]:

$$\hat{\mathbf{J}} = -\frac{\partial^2 \log p(D|H,\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}, \tag{65}$$

evaluated at the maximum likelihood parameters $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. This expansion allows for the marginalisation over $\boldsymbol{\theta}$ to calculate the evidence for a given model in equation 63. Having solved the model evidence, the true data evidence needs to be assessed to maximise KL information. This evidence is, however, unavailable, but the KL information can be estimated using $\log p(D|H)$ as an approximation [70].

The approximation yields equation 62 with two additional terms proportional $O(k)$ and $O(1)$, corresponding to an integration constant and the prior [70]. In the asymptotic $n \to \infty$ limit, both contributions will become insignificant [70], and the BIC reduces to equation 62. The BIC is closely related to the maximum a posteriori solution and should predict the same solution in large n [70].

4.4 Monte Carlo Tree Search

Beyond the basic structure of how to write and rank the equations, an algorithm needs a method to search the equation space. One approach to this is the Monte Carlo Tree Search (MCTS), a method for finding optimal or near-optimal solutions to problems [71]. The method relies on the UCT metric for searching the tree [72]:

$$UCT = Q + \sqrt{\frac{2\log n}{N}}. \tag{66}$$

Where Q is the score of the explored node normalised to $[0,1]$, typically the mean of the children node scores, and N, n are the number of visits to the (potential) child node and parent node, respectively. The $\sqrt{2}$ can be rescaled to rebalance exploration and depth but will be fixed for the applications in the thesis. The method relies on building a tree of states and progresses through node selection, expansion, rollout and finally backpropagation [72].

The node selection traverses the tree using the UCT metric, selecting the top-scoring node in each case. Upon finding a node with unexplored options, one option is selected at random, and the tree expands with one new node. From the expanded node, a rollout is performed, a random sequence of actions until

a final state is reached. The result of that playout is backpropagated through the tree, updating the Q, n and N for each node until the first root node, see figure 10.

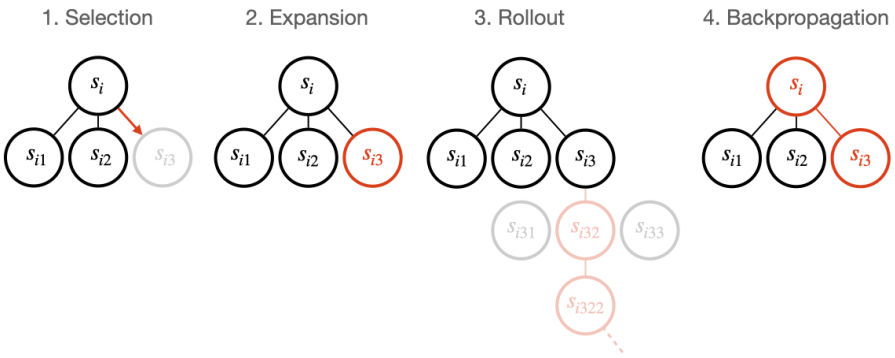

Figure 10: Visualization of the general MCTS process based on [72], with a branching factor of 3. Changes marked in red, s_i denotes initial state. 1 - Selects s_{i3} since an unexplored node is available from s_i. 2 - Creates new node s_{i3} of selected state and expands tree. 3 - Simulates succesive random states: $s_{i3} \rightarrow s_{i32} \rightarrow s_{i322} \rightarrow ...$, until end state. 4 - Results of simulation updates s_{i3} and then s_i and process restarts.

Extensions of the basic MCTS include Option Monte Carlo Tree Search (O-MCTS), in which the tree search uses options that expand the tree according to a rule [72]. If efficient rules for exploration can be specified, then the addition of options should avoid unnecessary searches and states. MCTS has previously been investigated for symbolic regression in the base form and in the form of nested search [1] [65].

4.5 Active learning

Active learning is a technique and concept used in machine learning in which the most informative data points are specifically selected or generated and used for training [73]. The technique is useful in cases where information is expensive or limited. The application is problem-specific and requires the selection of several properties [74], including a sampling and query strategy.

Depending on the context, the request might be for new data or the correct labels for already available samples. The query strategy determines the selection strategy for requests and determines which data property is important. Previous strategies for labelling tasks include query by committee, expected model change, and Fisher information ratio [74]. The method can, at its core, be summarised as an optimisation problem:

$$x_{new} = \arg\max_{x} f(x|M, \mathbf{x}_{old}), \tag{67}$$

where the system attempts to find a new parameter selection x based on available data $\mathbf{x}_{old}$. The query is performed based on some function f on the properties M of the system that offers a maximal benefit. The quantity offers large creative freedom to the user and can be, for instance, entropy or variance. Active learning has received significant attention in research, having explored options such as recommendation systems and deep learning [73].

4.6 Hypothesis generator

The above concepts can together be used to form a symbolic regressor. Starting with the hypothesis generator it uses an MCTS-inspired approach, differing in three senses from the original MCTS algorithm. First, growth is performed using a select subset of nodes, aiming to close the tree quickly. Second, the algorithm grows until an end state and is evaluated immediately without a rollout step. Third, the structure blocks further exploration of invalid subtrees.

The process starts at the root node, the system control node. First, the algorithm searches the space for expandable nodes using `get_propagation_node`. Upon reaching a proper node, the tree expands at that node until it forms a complete equation. The method `get_options` assigns which operators or operands are available and used. To promote parsimony and prevent runaway expressions, the method tries to close the expression as fast as possible using the variable `open_slots` as a guide, see explanation below. When an equation is formed, it is evaluated, and results are propagated up along the tree. The tree is constructed using postfix notation, the proper syntax is enforced in `get_options`. The general process can be seen in figure 11 with pseudocode in algorithms 1 - 4.

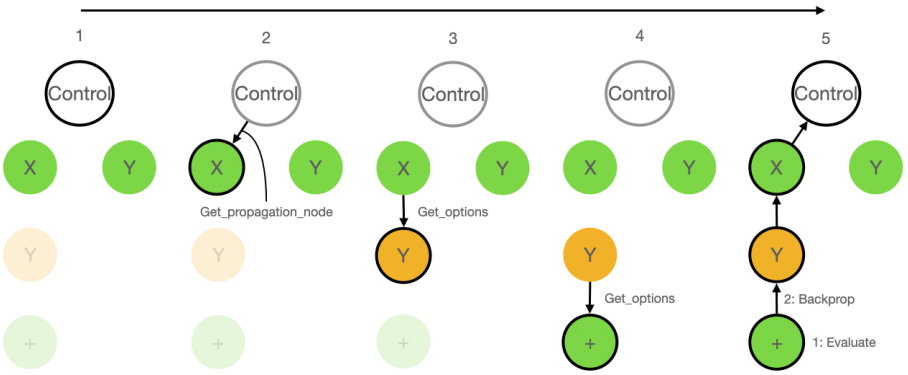

Figure 11: One step of the evolution process using variable set $\{X, Y\}$ and operation set $\{+, -\}$. The selected node is marked with a black ring, complete equations in green and incomplete equations in yellow. After step 5, the process can be continued from 1 by including the transparent nodes.

Algorithm 1 Evolve

Require: X = X_data, $Y = f(*X)$ = Y_data and interface.
Require: Sym = set of allowed operators.
Require: N_0 = instance of MCTS control node.

if Node has no options **then**
    Node = Get_propagation_node (Node)
end if
Options = Get_options(Node)
Grow(N, Options)
while Open_slots > 0 **do**
    Options = Get_options(N, Sym)
    Grow(Node, Options)
end while
Score, Params = Evaluate (Node, X, Y)
Backpropagate (Node, Score)
Update_solutions (N_0, Node)

We formulate and use the `open slots` variable to ensure that only syntactically valid equations can form in the postfix notation. Any binary operator requires two operands to close the expression. Similarly, using two binary operators requires two operands for the first and two for the second. However, one of the operators will form a subexpression, which can be used in the other operator, allowing for a syntactically valid expression with $N_{op} + 1$ operands. This is best visualized as a tree structure with operators as branches and operands as leaves.

This corresponds to parsing the string from left to right, adding one required operator for each operand passed. From this point of view, this measure can guide the selection of syntactically valid equations without a complete expression. It generalizes to operators with an arbitrary number of operands by changing the "cost" of an operator, and a complete equation is formed every time `open slots` is zero. For the measure, each operand provides $+1$ while each operator costs $-(N_{op} - 1)$, where N_{op} is the number of operands the operator requires.

This is because each operator looks backwards to find items to operate on. If there is an insufficient number of objects to operate on, it will not evaluate, meaning some leaves in the tree are unassigned. Likewise, if the measure is positive, there are unassigned operands which can not fit into the tree.

For example, the equation `xy+`, analyzed from left to right, first has `open slots` $= 0$, since `x` is a complete equation. Parsing to `y` gives $+1$, a state with dangling variables. Finally, `+` gives -1, bringing the total to zero, and $x+y$ is a complete equation.

4.6.1 Propagation and growth

The tree is propagated using the UCT algorithm, picking the highest-ranking node each time a new one is selected. When propagation is done, an option is selected, see algorithm 2, and a new node is created as a child of the explored node.

Algorithm 2 Get propagation node

```
Require: node
  Candidates = []
  for Node in Children do
      if Node != Exhausted then
          Candidates.Append(Node.UCT)
      end if
  end for
  return Candidates[Index(Max(UCT))]
```

4.6.2 Options

Option selection can be performed using any metric but is at baseline performed by random choice. The provided set of operands and operations are flexible in that they can use any structure that fits the open slot value. For instance, when zero slots are open, a variable or unitary operator needs to be introduced to maintain a syntactically valid state. However, any closed algebraic structure ($x + y$, e^x, $x^2 + x/y - e^{x-y^2}$, etc.) can substitute for operands.

To quickly evaluate the direction in the tree, the option selection attempts to close the equation immediately by following the rules seen in the *Get options*,

algorithm 3. Each node keeps track of visited options which are removed from the selection sets.

Algorithm 3 Get options

Require: Node
  if Open slots = 0 **then**
    Option = Random.Choice($\{Operands\} + \{Unary\_operations\}$)
  end if
  if Open slots = 1 **then**
    Option = Random.Choice($\{Binary\_operations\}$)
  end if
  return Option

4.6.3 Scoring and evaluation

When the algorithm finds an expression which can be evaluated, the code will convert the postfix string to infix and then translate it into an equation using the Sympy [75] `parse_expr` function. The function is then wrapped in a likelihood function wrapper assuming a Gaussian likelihood. If the equation contains free parameters, the code optimises the parameters using the Scipy [76] function `optimise. minimise` . The initial parameter guess is `ones(N_{Param})`. If the optimiser fails, it retries with `uniform(0,1,N_{Param})`. The log-likelihood is estimated using the model:

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \tag{68}$$

where y_i is the measured value, $f(x_i)$ is the trial function at point i and ϵ_i is an assumed Gaussian error with a fixed mean and standard deviation of $\mu = 0$ and σ set by the user. The effective log-likelihood calculation becomes:

$$\log L(f, \sigma) = \sum_{i=1}^{n} \frac{(y_i - f(\boldsymbol{x}_i))^2}{2\sigma^2}, \tag{69}$$

where n is the number of data points. The scoring of the equation is done using BIC or MDL. Both the BIC and MDL use a regularisation along with the log-likelihood, thus penalising certain structures in the solution. For all solutions, the normalisation in the likelihood is dropped since it is a constant and does not affect the exploration. The MDL score is based on the above likelihood:

$$MDL = -\log(L(f, \sigma)) + L_d(f) + n_{param} \cdot 32 \log_2 e, \tag{70}$$

where L is the likelihood, $L_d(f)$ is the description length of the equation and n_{param} is the number of parameters in the equation. The description length

is calculated as $L_d(f) = d \log L_{opset}$ based on the number of operators and variables, L_{opset}, in the lexicon, and d is the length in number of components. The contribution from parameters is based on the size of 32-bit floats converted to $\log_e$ basis. The BIC is calculated as defined in section 4.3.

For the full background of the description length see section 2.4.1 and section 4.7 for the lexicon of operators and operands. To transfer the score to an UCT-compatible score, an order preserving normalisation is performed:

$$S_{UCT} = \frac{1}{1 + S_{Metric}}, \tag{71}$$

where S_{Metric} is the score in BIC or MDL, inspired by [60]. Backpropagation is performed after each full equation where the Q score of each node is the mean of all backpropagations through the node. The total UCT is calculated according to equation 66.

Algorithm 4 Evaluate

Require: Terminal node
  Eqn = Parse_expression (Node)
  Log_likelihood = -Log(Sum((Eqn(X_data, Parameters) - Y_data)^2)/σ^2)

  if Equation has parameters **then**
    p_0 = Ones(N_{Params})
    ML_param = Minimize(Log_likelihood, p_0)
    if optimise fails **then**
      p_1 = Random(0,1,N_{Param})
      ML_param = Minimize(Log_likelihood, p_1)
    end if
  end if

  Max_likelihood = Log_likelihood(ML_param)
  if Metric = "BIC" **then**
    BIC = 2 $\cdot$ Max_likelihood + $N_{Param}\cdot$ Log(N_{Data})
  return BIC, ML_Parameters
  end if

  if Metric = "MDL" **then**
    MDL = Max_likelihood + $L_d(Eqn)$ + $N_{Param}\cdot$ Log(Precision(Params))
  return MDL, ML_Parameters
  end if

4.6.4 Data updating

Any data set may be limited in its power to explain a phenomenon. Upon manual request, the algorithm can request new data points from the target equation,

estimated by the point of maximum variance between the top solutions:

$$\boldsymbol{x}_{new} = \arg\max_{\boldsymbol{x}} : \sum_{n=1}^{N_{sol}} \frac{f_n(\mathbf{x})^2}{N_{sol}} - \frac{1}{N_{sol}} \sum_{n=1}^{N_{sol}} f_n(\mathbf{x}), \tag{72}$$

where f_n is the n:th top solution and N_{sol} is the number of solutions considered while finding a new point. The variance is optimised using Scipy's minimize function and new calculated boundaries. After a new datapoint is received, all the basis solutions are reevaluated, and the chain is updated. The new limits on data are set using the bounds of available data as the width:

$$W_i = max(\boldsymbol{x}_i) - min(\boldsymbol{x}_i), \tag{73}$$

where W_i is the domain width in dimension i and $\boldsymbol{x}_i$ are the samples in dimension i. The width is then used to extend the allowable domain as:

$$x_{i\_newMax} = max(x_i) + \frac{W_i}{2} \cdot u_i, \tag{74}$$

where x_i and W_i are the i:th dimension of $\boldsymbol{x}$ and $\boldsymbol{W}$ while u_i is a draw of a uniform random variable $U(0, 1)$ to avoid inter-variable correlations in the new data. The lower boundary is updated in a similar manner.

4.6.5 Hamiltonian module

The algorithm also includes a Hamiltonian module which can be used to search for potentials which fit the data. The expansion and search of the tree is performed as normal, but the output equation is used to produce the Hamiltonian system:

$$\frac{d\boldsymbol{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}}, \ \frac{d\boldsymbol{x}}{dt} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}, \tag{75}$$

where $\boldsymbol{x}$ is canonical position and $\boldsymbol{p}$ is canonical momenta, refer to section 3.3 for details. The module assumes cartesian coordinates and a time-independent Hamiltonian, allowing for the simplified search:

$$\frac{dp}{dt} = -\frac{\partial V(x)}{\partial x}, \ \frac{dx}{dt} = \frac{p}{m}. \tag{76}$$

Where $V(x)$ is a position-dependent potential and m is the mass of the observed particle. The trial solution is differentiated using Sympy and then solved for given initial conditions and time points using Scipy `odeint`. Equations containing parameters are optimised using the same strategy as above. The sum of squares is calculated in position space and used to calculate scores according to the metric of choice.

4.7 Benchmarking

The code was benchmarked using the Nguyen and Feynman symbolic regression benchmarks. The full equation sets used can be found in Appendix B. For all equations, the available set of constants and functions were:

$$\text{Constants: } [1, 2, 3, e, \pi, Par_1/a],$$
$$\text{Operations: } [+, -, \times, /, \log, \exp, \text{pow}, \text{sqrt}, \sin, \cos].$$

The full output of all runs is consolidated in Appendix C. A summary of the results follows below. optimiser settings for all runs were: `method = "Powell"`, `maxiter = 100` and `xtol = 1E-3`. All computations were performed on a 13" 2020 MacBook Pro with 8GB of RAM and a 1.4GHz Intel i5 quad-core processor.

4.7.1 Nguyen equations

For the Nguyen benchmark [67], there seems to be little difference in the correct predictive power of BIC or MDL in the noiseless case. MDL consistently produces smaller expressions than BIC. See figure 12 and table 6 for an example of convergence over the number of iterations and consolidated predictions.

Nguyen equation search

Ref. eq.	BIC Correct %	MDL Correct %
$x^3 + x^2 + x$	0	0
$x^4 + x^3 + x^2 + x$	0	0
$x^5 + x^4 + x^3 + x^2 + x$	0	0
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	0	0
$\sin x^2 * \cos x - 1$	0	0
$\sin x + \sin(x + x^2)$	0	0
$\log(x + 1) \log(x^2 + 1)$	0	0
$\sqrt{x}$	100	40
$\sin x + \sin y^2$	0	0
$2 * \sin x * \cos y$	0	0

Table 6: Correct predictions of the Nguyen equations on 5 runs across each equation with a search time of $50s$. Noiseless data, $\sigma = 1.0$.

Several equations approach unity in UCT score for the noiseless evaluation of the Nguyen equations, see figure 12. In the case of MDL, the maximum score is capped by the use of the complexity term, meaning that a correct inference at best reaches $1/(1 + l(f_j))$, where $l(f)$ is the length of f in number of elements.

The algorithm fails to find the correct solution for all equations except for number eight, see table 12. Equation nine does approach the correct solution

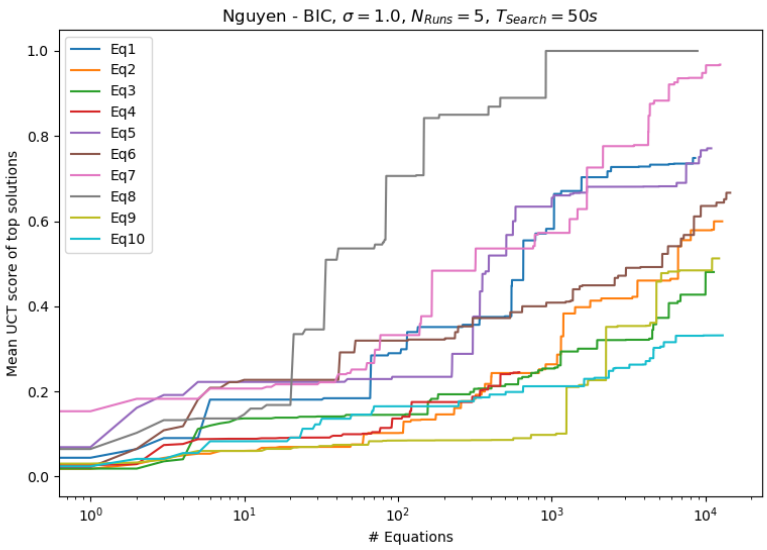

Figure 12: Mean UCT score for top solutions of 5 runs of $50s$ with BIC as a function of explored equations.

in several cases but does not find it. For both the MDL and BIC, the number of evaluated solutions is typically on the order of 10^4 for a runtime of $50s$, with a slight slowdown as a higher number of evaluations are performed. Noisy data does not seem to affect the outcome to a large degree, see table 7.

Nguyen equation search with noisy data

Ref. eq.	BIC Correct %	MDL Correct %
$x^3 + x^2 + x$	0	0
$x^4 + x^3 + x^2 + x$	0	0
$x^5 + x^4 + x^3 + x^2 + x$	0	0
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	0	0
$\sin x^2 * \cos x - 1$	0	0
$\sin x + \sin(x + x^2)$	0	0
$\log(x + 1) \log(x^2 + 1)$	0	0
$\sqrt{x}$	60	100
$\sin x + \sin y^2$	0	0
$2 * \sin x * \cos y$	0	0

Table 7: Correct predictions of the Nguyen equations on 5 runs across each equation with noise, for a search time of $50s$. Noise: $N(0, 0.1)$, $\sigma = 0.1$.

Both for MDL and BIC the predicted equations do have forms approximating the correct solutions. For the polynomials, many of the solutions have Taylor expansions similar to the correct solution, for example, $xe^x \approx x + x^2 + x^3/2 + x^4/6 + O(x^5)$.

When using the active learning module, the top ten solutions are used to select the new point. After running for half the duration, the algorithm resamples the spaces and updates the solution space. The resampled solutions seem to perform similarly to the standard approach, where some problems improve and some decrease in final UCT score, see figure 13 and table 8.

Nguyen equation search with resampling

Ref. eq.	BIC Correct %	MDL Correct %
$x^3 + x^2 + x$	0	0
$x^4 + x^3 + x^2 + x$	0	0
$x^5 + x^4 + x^3 + x^2 + x$	0	0
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	0	0
$\sin x^2 * \cos x - 1$	0	0
$\sin x + \sin(x + x^2)$	0	0
$\log(x + 1) \log(x^2 + 1)$	0	0
$\sqrt{x}$	100	100
$\sin x + \sin y^2$	0	0
$2 * \sin x * \cos y$	0	0

Table 8: Correct predictions of the Nguyen equations on 5 runs across each equation using resampling after $25s$ for a total search of $50s$. Noiseless data, $\sigma = 1.0$.

Some solutions seem to converge faster than in the non-resampled case. See, for instance, Nguyen equation 7. The resampling does generate problematic solutions in the multivariate case with negative UCT scores as can be seen in figure 13.

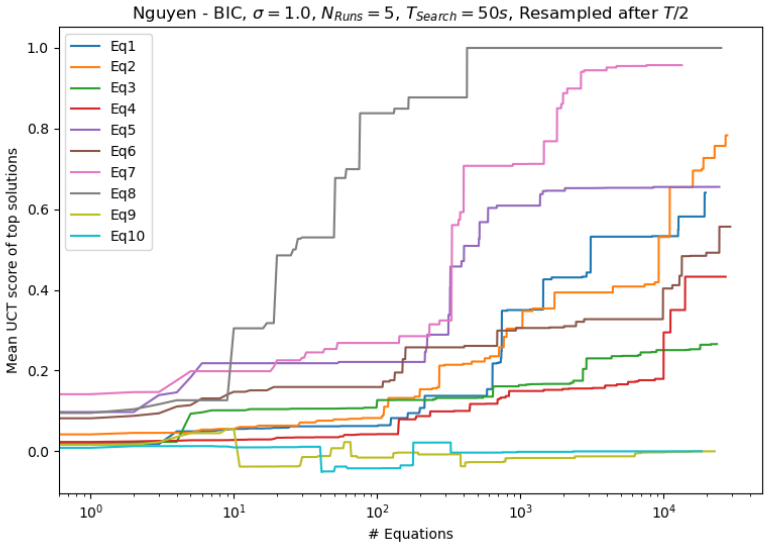

Figure 13: Mean UCT of top solutions for 5 runs of $50s$ using BIC with resampling after $25s$.

4.7.2 Feynman equations

For the Feynman equations [68], the regressor finds few of the solutions. In some cases, it finds the correct solution but uses an unnecessary parameter that typically defaults to 1 or 0. The MDL and BIC metrics seem to perform similarly with a slight advantage to MDL, see table 9 - 10.

In figure 14, several equations approach unity, showing that the suggested equations agree well with the data, even if it does not match the reference equation.

Feynman equation search

Eq.	BIC %	MDL %
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2}}}{2\sqrt{\pi}}$	0	0
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	0	0
$\frac{\sqrt{2}e^{-\frac{(\theta-\theta_1)^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	0	0
$\sqrt{(-x_1+x_2)^2+(-y_1+y_2)^2}$	0	0
$\frac{Gm_1m_2}{(-x_1+x_2)^2+(-y_1+y_2)^2+(-z_1+z_2)^2}$	0	0
$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$	0	0
$x_1y_1+x_2y_2+x_3y_3$	0	0
$N_n\mu$	60	100
$\frac{m\left(u^2+v^2+w^2\right)}{2}$	0	0
$\frac{q_1q_2}{4\pi\epsilon r^2}$	0	0

Table 9: Percentage correct predictions for Feynman equations 1-10, noiseless data with $\sigma = 1.0$.

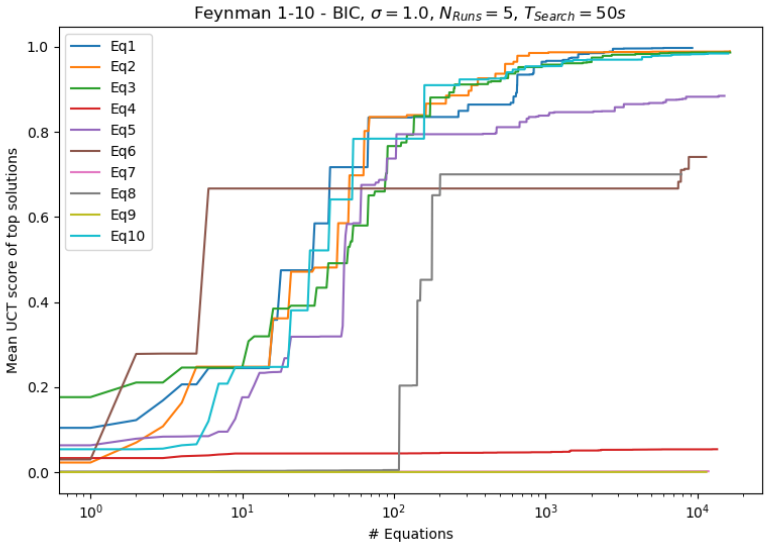

Figure 14: Mean UCT score for top solutions of 5 runs of $50s$ with BIC as a function of explored equations.

Feynman equation search with noisy data

Eq.	BIC %	MDL %
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2}}}{2\sqrt{\pi}}$	0	0
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	0	0
$\frac{\sqrt{2}e^{-\frac{(\theta-\theta_1)^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	0	0
$\sqrt{(-x_1+x_2)^2+(-y_1+y_2)^2}$	0	0
$\frac{Gm_1m_2}{(-x_1+x_2)^2+(-y_1+y_2)^2+(-z_1+z_2)^2}$	0	0
$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$	0	0
$x_1y_1+x_2y_2+x_3y_3$	0	0
$N_n\mu$	40	60
$\frac{m\left(u^2+v^2+w^2\right)}{2}$	0	0
$\frac{q_1q_2}{4\pi\epsilon r^2}$	0	0

Table 10: Percentage correct predictions for Feynman equations 1-10, data with noise:$N(0, 0.1)$ and $\sigma = 0.1$.

4.7.3 Hamiltonian module

The Hamiltonian solver was tested using four simple potentials for five runs of $100s$, evaluated at 100 positions equally spaced in time $t \in [0, 2]$. The ODE solver used a relative tolerance of `rtol` $= 1E^{-4}$ with a maximum number of iterations `mxstep` $= 200$. Mass was fixed at `m = 1.9kg` with optimiser settings the same as above. Initial conditions were independently randomized as $U(0.5, 2.0)$ for both position and velocity.

A typical run of solutions compared to the reference can be seen in figure 15. The algorithm consistently found the correct potential in roughly half the cases, see table 11. MDL seems to outperform BIC in this application slightly. In the case of $8x$, the BIC gets a numerically acceptable solution but suggests additional constants or factors such as ax/m or $ax + m$. Both formulations give the same trajectory for constant m, but the additional term is not considered an acceptable solution.

The number of evaluations is roughly two orders of magnitude smaller than the normal operation, see figure 16.

Potential search using Hamiltonian module

Potential	BIC %	MDL %
x^2	40	100
$5x^2$	0	0
$8x$	0[1)]	80
$\frac{1}{x^2}$	0	0

Table 11: Percentage correct predictions for select potentials using the Hamiltonian module, $\sigma = 1.0$. [1)] Reaches 80 % success but contains unnecessary components, for instance, squared parameters and added constants.

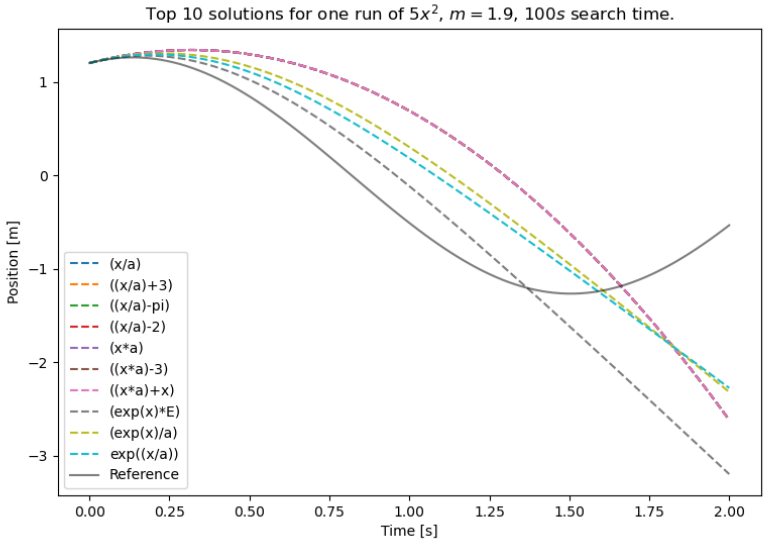

Figure 15: A typical run of the algorithm using the Hamiltonian module on a reference potential $5x^2$

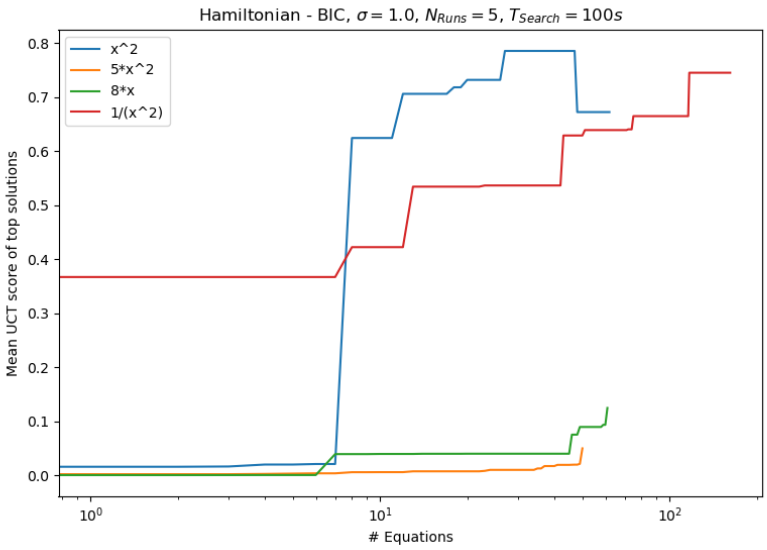

Figure 16: Mean UCT of top solutions for 100s search using BIC scoring metric and no noise. Dip in score for x^2 related to numerical issues in solver.

4.8 Discussion

The algorithm was inspired by the directed structural approach of MCTS. It can be seen as a variation on the O-MCTS approach using only options, with no rollout. The approach seems competitive compared to traditional MCTS for symbolic regression. Using the values from [1], the UCT score reaches similar levels for one to two orders of magnitude fewer iterations on Nguyen equation 7 while using the BIC. See figure 12. Note that this assumes that the error in the paper is the sum of squares, which is reasonable given the application and description in the paper. The MDL results are not fit for direct comparison.

Compared to the two other algorithms in the same paper, EJC and nested search, the performance of our algorithm seems roughly similar on Nguyen 7 regarding UCT score [1]. To compare the algorithms properly, the loss metric used in the paper needs to be confirmed.

Our algorithm generally struggles to find the correct expression for most problems, except for the short expressions. The algorithm's success in Nguyen equation eight is likely due to the structure of the search. The tree must explore `x` to progress at all. Given that `x` traces $\sqrt{x}$ fairly well for low x, the node will likely be selected for deeper exploration, leading directly to `x sqrt`.

The resampling strategy did not seem to affect the UCT scores significantly but gave a slight improvement in correct predictions for Nguyen 8. See figure 13

and table 8. The multivariate equations showed negative scores after resampling, likely caused by a data structure mismatch. The UCT scores should be monotonically increasing over time, as is the case for the univariate equations.

In Hamiltonian operation, some potentials with some initial conditions seem to diverge. This is most likely related to the ODE solver, which, under certain conditions, returns invalid values, resulting in the dip in figure 16. In both the Hamiltonian and the resampling case, the issues are most likely related to the specific implementation and not related to the method.

4.8.1 Algorithm structure

The option selection used in the algorithm is performed since most strings, even those made from the selection above, are not valid equations. For example, an expression can be written with general operators and variables corresponding to branches and leaves in a binary tree. The tree will require that the number of variables for binary operators be:

$$N_{var} = N_{Op} + 1. \tag{77}$$

In practice, the traditional MCTS relies upon random rollout, meaning a probability of selecting a variable or operator for each step. Thus, the valid permutations, with regards to the balance of each type for a given length of string, become a binomial distribution $p = 0.5$ with $k = (N_{tot} + 1)/2$. The probability of selecting the correct number of variables and operators, giving no preference to order, is less than 30% for a depth of 10, see figure 17.

If the probability is skewed by using an unbalanced number of variables and operators, the probability of finding the correct number generally decreases for long expressions. An even smaller subspace is valid if one also considers the order of operands and operators. Thus, a random search is unlikely to generate valid expressions, and a significant time can be spent searching syntactically invalid space.

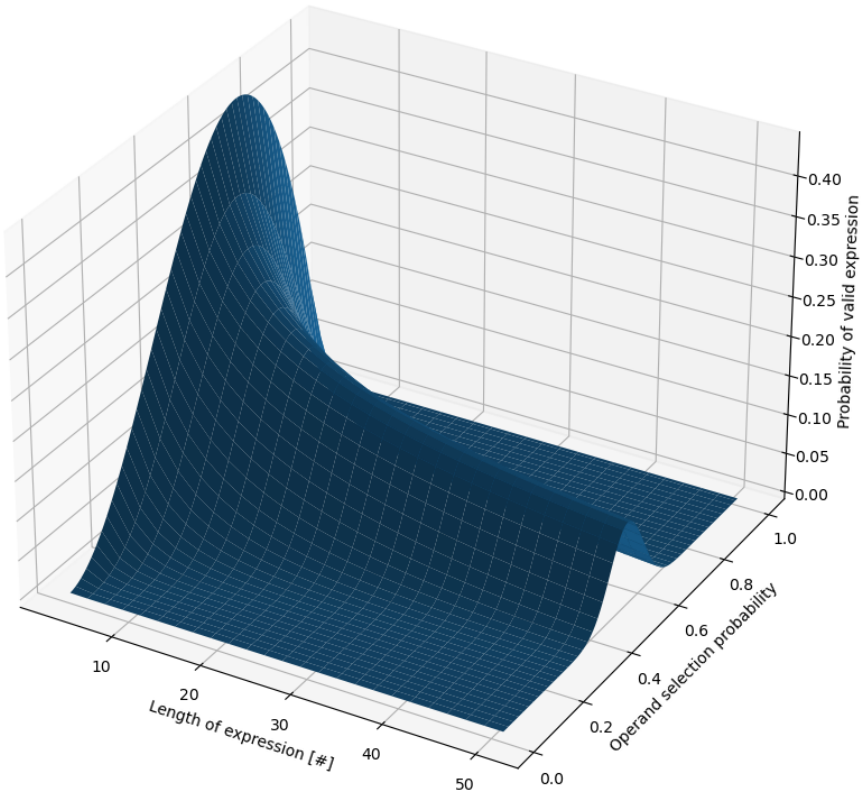

Figure 17: Probility of selecting the correct amount of variables given the probability of selecting a variable (p). The minimum number is limited to 3 for clarity.

The postfix approach combined with the tree structure offers a benefit in the search space: invalid subtree search limitation. Unlike the prefix notation, where equations grow inside out, postfix uses a wrapping approach:

$$\text{Prefix: } f(\_,\_) \to f(g(\_,\_),\_) \to f(g(x,\_),\_) \to f(g(x,y),\_) \to f(g(x,y),z).$$

$$\text{Postfix: } x \to x,y \to g(x,y) \to g(x,y) \to g(x,y),z \to f(g(x,y),z).$$

If $g(x,y)$ is evaluated and found to be unfit, for instance, if $g = /$ and $y = 0$, then no expression containing $g(x,y)$ can be a valid solution. Thus, the tree $[x,y,g]$ can be marked as exhausted, and no children of that tree need to be explored, greatly limiting the search space.

The implemented MCTS in the algorithm performs poorly on both the Nguyen and Feynman benchmarks and rarely finds the correct solution. For some equa-

tions, this is a result of the search structure which limits a significant portion of the search space. By concurrent selection of nodes in a way which closes the equation, the max limit on the open complexity is one. This leads to issues for certain of the Nguyen equations. In some cases, it constrains the possible allowable searches, for instance for equation one:

$$x^3 + x^2 + x = \texttt{x3\^{}x2\^{}+x+} = \texttt{x1+x*x*x+}.$$

The first postfix example has an open complexity string of $(0, 1, 0, 1, 2, 1, 0, 1, 0)$ while the second has $(0, 1, 0, 1, 0, 1, 0, 1, 0)$. Thus the first is rejected by the option provider meaning only certain paths can be evaluated. In other cases, for instance for equation six, the correct solution can't be reached (at least not through a straightforward path):

$$\sin(x) + \sin(x + x^2) = \texttt{xsin x2\^{}x+sin +}.$$

The explanation is that the "inside" expression of the second sinus term requires an open complexity of two to form since the complexity string is $(0, 0, 1, 2, 1, 2, 1, 1, 0)$.

The MDL application penalizes long expressions by adding to the score for each component in the equation. For any expression length longer than zero the UCT score can not reach unity anymore, even if the error is zero. For long expressions, the exploration term in UCT will become dominant even if the data is sufficiently informative. In retrospect, the MDL might constitute a valid scoring option but not without modifications to the original algorithm.

5 Discussion

The thesis aimed to investigate how hypotheses are formed using the bottom-up approach to identify potential inductive biases. The analysis focused on three main concepts: Symmetry, entropy and MDL. Two codes were produced to investigate how these approaches can be implemented: one implementing the process in [50] for symmetry finding and one MCTS-inspired code for general symbolic regression. In the exploration and implementation, several interesting questions of a general nature arose and will be discussed below briefly.

5.1 Complexity of a Hamiltonian system

In the MDL framework, the model is evaluated on the size of the compressed representation of the data. For any typical equation, the system is of the form:

$$f(x) = y. \tag{78}$$

As such, the output y is defined in terms of the input and the function. Disregarding deviations from the model, the full complexity of the system is then given by the complexity of the model plus the complexity of the input data:

$$C_e(D) = C_e(f) + C_e(x). \tag{79}$$

Where C_e is the effective complexity of the description, D is the combined data x and y. Normally, the complexity of the input x is not included when comparing models since it has an equal contribution in the different models. However, it should be recognized in the most general case.

The simplest representation of the data, corresponding to a prior estimate of the complexity, is the cost of simply storing the data set as-is. In other words, each data point is encoded using a small set of data corresponding to the values, which is then repeated independently for all points in the data set. This storage option represents the position of randomness, that there is no regularity to be compressed.

For a Hamiltonian-type system, the data to be compressed consists of the positions $\boldsymbol{q}$ and the momenta $\boldsymbol{p}$ given at time t. If one were to "naively" store the phase space data, it would require $2 \times n_{sample} \times n_{dim} \times d$, where n_{sample} is the number of samples and n_{dim} is the dimensionality of the position space. d is the complexity of each data point in bits. For simplicity in the analysis, assume double precision (64 bits). Thus, we can conclude that the information theoretical cost at each point in time is roughly 128 bits per sample and dimension, corresponding to 16 bytes. The time data also needs to be stored, but it is shared by both models.

In a Hamiltonian system, the trajectory can be fully described in terms of the initial conditions, time points and the potential. As such, the base description length can be estimated as follows:

$$C_e(D) = C_e(t) + C_e(V) + C_e(\boldsymbol{x}_0) + C_e(P(...)) \tag{80}$$

Where $\boldsymbol{x}_0$ are the initial conditions, V is the potential and t is the time points. There also exists an overhead complexity related to the program, which calculates the trajectory from the parameters, here denoted $C_e(P(...))$.

In practice, the initial condition complexity becomes the description length of a single data point, multiplied by the number of dimensions times two (since initial momentum is also needed). The complexity of the potential is more difficult to describe since the way the potential is encoded affects the descriptive length. For a rough complexity estimate in this framework, one needs a complexity description of a code. The code should describe the potential, describe the Hamiltonian, differentiate the potential and solve the ODE at the provided time points. This creates a significant overhead since all the required components must be included in the analysis.

The complexity of the parameters in a Hamiltonian program is trivial: $2 \times n_{dim} \times d$, corresponding to the initial conditions. If we again assume float64 numbers, the corresponding size will be 16 bytes per dimension. A direct measurement of the model is unfeasible. A rough upper limit estimate can be calculated using the size of symbolic regressor code from chapter 4.6 and its dependencies used, which amount to a few gigabytes, or on the order of 10^9 bytes. Since most of the code in the dependencies is unused, this is a clear overestimate.

Comparing the model size of the program with the initial condition size shows we can immediately disregard the initial condition complexity. Compared to the storage of data this is significantly larger and we would have to naively store on the order of 10^8 samples in order before the Hamiltonian model would constitute compression.

While the number seems vast it should be considered that this comparison is unfair since most of the overhead is valid for any Hamiltonian system. In other words, if the cost of this computational complexity is viewed in isolation, it fails to consider the large gain one might have compressing all available trajectories. Thus in a similar fashion to the meta-complexity briefly discussed in the introduction, if a code can compress a practically infinite number of solutions, then it may not be valid to view in isolation.

Arguably, if all trajectories are interesting (but not sampled), it would be accurate to assign the cost of this overhead as an average cost across all possibilities. Given the vast size of this space, the average overhead cost would approach zero. What would remain is the compressed representation in initial conditions and the description of the potential.

Constrained by this view, a simple encoding of the potential as a string would be as a set of ASCII characters, all with the complexity of 8 bits. Assuming a typical string with an equation is on the order of 10^1 symbols long, then compression will start to be effective around 10^1 samples, a significant improvement.

The argument is an argument in the ideal MDL sense, where the compression is explained in terms of a program that can be used to output the sought string. In the practical MDL framework, the user is free to make practical approximations for which we might disregard the overhead costs for this reason.

The reasoning also applies to general compression algorithms widely used on the internet. If one downloads a compressed file and then an algorithm for decompression, the total compressed representation of the data is thus the data plus the algorithm. If one then downloads another set of data with the same compression, is the complexity cost paid again or should the dataset be extended into both downloads?

As such, we propose that the MDL framework may be overly restrictive in the analysis of models. While the framework works well for models within the class and shares overhead, it is not obvious if one can accurately evaluate a method based on a subset of the data. Potentially, one needs to consider that for a general theory, the interesting property is not the compression of a special case but the ability to effectively compress *all* relevant cases. One should also highlight that if the goal is to encapsulate all regularity in the model, then it is likely that the model would become complex for complex systems.

5.2 The MDL paradox

The MDL attempts to formalize the view of complexity in terms of algorithmic complexity, which it manages in the ideal case and to a lesser degree in the crude case. However, it does not necessarily conform to the human view of complexity. For instance, compare the equation:

$$f_1(x) = \frac{2}{x}, \tag{81}$$

with:

$$f_2(x) = 2 + x. \tag{82}$$

In the MDL application used in section 4.6 of the thesis, both are equally complex if both operators are available in the operation space. The output space of the first function could be argued to be more complicated than the second due to the presence of the singularity. The second function has one non-zero derivative, while the first has infinitely many and is discontinuous at $x = 0$. As such, it is not clear that these two functions can be considered equal in the sense of complexity.

Imagine one has a data set with two hypotheses, both of which have high fitness and similar complexity. One traces a short curve between all points and the other moves almost erratically between the data points. For instance, compare $f(x) = \cos x + m$, where m is a free parameter to be optimized before use, with $g(x) = \cos 10x$ in figure 18.

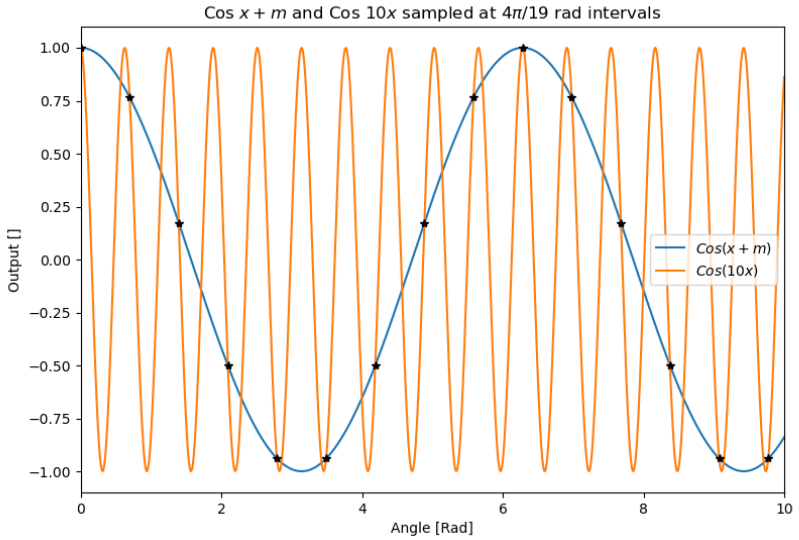

Figure 18: Example of two hypotheses with similar fitness and complexity but different perceived complexity. m is set to zero in the figure, being the most likely parameter value, but it is allowed to vary freely in the general case.

If the data is sampled such that the points align with the periodicity, then both are valid solutions to the problem, perhaps even to arbitrary precision. The first equation has a larger parameter space, suggesting it would be complicated according to the BIC criterion. If one were to evaluate the function behaviour on a "human" basis, it would be hard to justify the extra movements between the points.

From the perspective of MDL, there is little difference between the two options as both offer comparable compression of the data. From a human perspective (the author's at least), the second function looks unintuitive, not because it doesn't fit the data, but because it has excessive movement in the output space.

When comparing in-class hypotheses, one may also run into the problem of composite functions. For instance, arcsin is a composite function of exponents yet acceptable as a single operator in many cases. One method to ameliorate the difference is assigning different operators different complexities or using special

complexity penalties, such as in [77] or [64]. The general approach to this issue is an open problem.

Unlike the discussion in section 5.1, this is more likely related to the specific practical MDL approach. In the ideal MDL, the representations would all be minimal representations of each kind, meaning that the complexity of a composite function may differ from the approximate measure. To exemplify, in our symbolic regressor algorithm, the complexity of cos and sin is equal to that of any other operator. Sin is implemented in many different ways in computers. One implementation of sin, using the CORDIC algorithm, requires addition, subtraction and bitwise operations [78]. Thus, sin is likely more complicated than the addition operation. However, it is unclear if it is more complex than division or multiplications.

5.3 Symmetries in equations

In practical terms, the inductive bias of symmetries comes from constraining a function's possible behaviour. For instance, if one knows a mirror symmetry of $f(x) = y$ around $x = 0$ one can immediately rule out all odd functions from the hypothesis space. In a similar fashion, symmetries of continuous multivariate functions are transformations of $\mathbf{x}$ such that the total change is zero:

$$\nabla f(\mathbf{x}) \cdot d\mathbf{x}_s = 0. \tag{83}$$

Where f is some scalar function of $\mathbf{x}$ and $d\mathbf{x}_s$ is an infinitesimal step in the symmetry "direction". From this definition, it should be clear that a symmetry can involve a subset of the parameter space, for instance, if one has the relation:

$$f(\mathbf{x}) = f(x, y, z) = \frac{1}{\sqrt{x^2 + y^2 + z^2}}, \tag{84}$$

then a symmetry transformation can be realized as a 2D or 3D rotation, providing a significant reduction in potential allowable solutions to the problem of finding f. In AI Feynman 2.0 [63], this is expressed through their definition of generalized symmetry. If one can find a subset of parameters for which one may form an independent subexpression [63]:

$$f(x_1, x_2, x_3, ...) = g(h(x_1, x_2), x_3, ...), \tag{85}$$

where g is f expressed in terms of the sub-function h of x_1, x_2. From this it should be clear that if h is invariant under some transformation $h(x_1, x_2) = h(x_1', x_2')$ then so is $g(h(x_1', x_2'), x_3, ...)$. In the paper, their implementation is expressed in terms of the gradients through the chain rule since $\nabla_{x_1,x_2} f(x_1, x_2, x_3, ...) \propto \nabla h(x_1, x_2)$, allowing for direct comparison between

the normalized gradients [63]. Thus, knowledge of h would clearly be helpful in evaluating the full equation f.

Symmetry also influences solutions in the parametric space. Given a model with a parametric symmetry, the symmetry allows the model to map multiple values of the parameters to the same value in output space. Using the Bayesian framework, a simple constructed example would be a linear model:

$$M_1 = kx, \text{ and } M_2 = |k|x. \tag{86}$$

With the parameters on the space of the real numbers, one can effectively model the second model using the first and an extreme prior in the negative k space. Both models will yield the same likelihood for a given data set in the positive output space. But all the prior probability in negative k will fold over to the positive k domain. A more convincing example might be $\sin \theta x$ compared with $\sin(x + \theta)$. While the behaviour is different in both functions, the periodicity of the second allows the prior to be limited to $[0, 2\pi]$.

5.4 Relationship between entropy, symmetry and MDL

Previously we have discussed the potential inductive biases independently, now we will attempt to highlight the possible connections. The three concepts come from different fields of science but all relate to different descriptions of structure.

Entropy is typically associated with thermodynamics and statistical mechanics, being a measure of the multiplicity of states. In simple terms, the Boltzmann entropy of a state is correlated to the number of ways to realize the specific combination. According to Rosen et al. [79], a maximally disordered state corresponds to one with the highest number of symmetries since entropy is positively correlated with symmetry. Lin [80] posits that with Shannon/Gibbs entropy, the maximum entropy is reached when all states are indistinguishable from one another, meaning maximum symmetry. This belief is not shared by E. Bormashenko [31], who argues that increased entropy is the lack of symmetry using N-magnet systems as an example.

The discrepancy in the discussion is likely due to the difference in the mathematical and individual perspectives on symmetry. In math, a symmetry is an isomorphism that leaves the observed metric invariant. This definition of entropy implies that Bormashenko's proposed state already has several inherent symmetries. To illustrate, imagine a two-particle system $\boldsymbol{P}_{s2}$ which can occupy two different states s_0, s_1. If the metric is the number of particles in state s_1, then each state can be characterized in terms of their symmetry transformations. Given the metric M and transformations S_i, the symmetries affect the state as:

$$M(\boldsymbol{P}) = M(S_i(\boldsymbol{P})). \tag{87}$$

Where M is the sum of all states s_1. In the case of two indistinguishable particles, there is only the identity symmetry of the edge states (2 s_0 or 2 s_1) since all permutations are identical. For the combined states, another allowed symmetry is the exchange of particle states. Thus, the mixed state metric is invariant under more transformations, which is conceptually equivalent to having more acceptable permutations. Forcing a mirror symmetry, equivalent to an identity transformation on the edge states, would break the exchange symmetry and only allow an identity transformation on each state. The Gibbs/Shannon entropy:

$$-(2\frac{\ln(1/4)}{4} + \frac{\ln(1/2)}{2}) > -2\frac{\ln(1/2)}{2}. \tag{88}$$

Thus, the enforced symmetric state has less entropy. This is more apparent for any individual macrostate of N particles. The number of combinations of k s_1 states is given by the binomial coefficient. Noting that each state is an isomorphism of another state, it becomes obvious that N over k is not only the number of permutations, it's also related to the number of symmetry transformations. Enforcing the mirror symmetry on this system such that particle i:

$$P_{i+N/2} = P_i. \tag{89}$$

Where N is an even integer. Since half the states are determined by the symmetry, this is equivalent to picking N/2 over k/2. In all but the edge cases, $\binom{N}{k}$ is larger than $\binom{N/2}{k/2}$ since:

$$\binom{n/2}{k/2} = \left(\prod_{i=0}^{k/2} \frac{k-i}{n-i}\right)\left(\prod_{j=0}^{n/2-k/2} \frac{n-k-j}{n-k/2-j}\right)\binom{n}{k}. \tag{90}$$

See details in Appendix A. All the factors are strictly smaller than one, unless $k = 0$ for which both systems have the same multiplicity. Imposing the mirror symmetry condition limits other less apparent symmetries. This can also be conceptualized as the new system being smaller, thus requiring less complexity to explain.

A similar argument relating entropy and symmetry has been put forward by Shu-Kun Lin [80], who holds that symmetry is in correspondence with entropy through:

$$S = -\log\sigma = \log w_s, \tag{91}$$

where S is the Boltzmann entropy with K_b set to unity, σ is the inverse multiplicity of symmetry transformations and w_s is the apparent number of the symmetrical states. As discussed earlier in the section, this relationship is clear

in the Boltzmann entropy, which is the limiting case of Gibbs or Shannon's entropy with $p(x_i) = p(x_j) \forall i, j$, see section 2.3. The other entropies can assume smaller values, meaning the system has a lower degree of apparent symmetry and is also further from equilibrium [80]. It should be noted that Lin's line of reasoning only applies to closed systems but remains an interesting connection between symmetry and information.

As mentioned in section 2.3, entropy is closely connected to the concept of MDL. To summarize, the MDL is a method for model selection such that the total representation is as small as possible. In contrast, the maximum entropy approach can be described as finding a solution such that the data is maximally informative through its connection to information. As shown by Li and Vitányi [81], the maximum entropy approach can be considered a special case of the MDL approach under certain conditions. They exemplify it using die casts, which have a maximal MDL description [81]:

$$l_{max} = k \log_2 n + \log_2 \binom{n}{n_1, n_2, ..., n_k} + O(\log_2 \log_2 n), \tag{92}$$

where l_{max} is an upper limit on description length in bits, k is the number of outcomes and n is the number of tries. By using Stirling's approximation for the multinomial, the equation can be simplified [81]. Expanding on their solution process, the intermediate stage is:

$$\begin{aligned} k \log_2 n + n \log_2 n - n \log_2 e - \sum_{i=1}^{k} (n_i \log_2 n_i - n_i log_2 e) \\ = k \log_2 n + n \log_2 n - \sum_{i=1}^{k} n_i \log_2 n_i, \end{aligned} \tag{93}$$

where terms of order $O(\log_2 n)$, $O(\log_2 n_i)$ and $O(\log_2 \log_2 n)$ have been dropped. By multiplying each n_i term by n/n and extracting the nominator outside the sum, the first term can be combined with the sum:

$$l_{max} = k \log_2 n - n \sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}, \tag{94}$$

where the first $n \log_2 n$ is split into $(n_1 + n_2 + ...n_k) \log_2 n$, which is the result of Li and Vitányi. The second term is proportional to the Shannon entropy with probabilities corresponding to relative frequencies. Thus, a maximum entropy solution for the die casts is analogous to minimizing the description length [81].

5.5 Important paths and constraints

One of the main issues of symbolic regression is the vast search space a small handful of operators and variables generate. As outlined in 1.6, an efficient search algorithm must make a prior decision on how to search.

Given the syntactical context of the symbolic regressor (the bracket-free notation/equations built from components), the first obvious constraint is to limit the search to syntactically correct expressions, as outlined in section 4.8. There are also functionally equivalent expressions (`xx+` = `x2*`) that don't need to be evaluated. As commented on by Bartlett et al. [38], the number of unique equations is smaller than the full combinatoric space, sometimes orders of magnitude smaller.

In many cases, the possibility of constraining the search further also exists by only considering forms which provide the correct unit. For instance, if one tries to compose new quantities from $\{x, m, t\}$ with units $\{m, kg, s\}$ and operators $\{+, -, \times, /\}$ the unconstrained number of combinations with one operator is $2 \cdot 3! + 2 \cdot 3^2 = 30$. Upon requiring units to make sense ($kg + m$ is not compatible, $kg \times m$ is.) the number drops to $2 \cdot 3 + 3! + 3^2 = 21$. If one constrains the output unit to be velocity (ms^{-1}), only one combination remains.

With three operands and non-commuting operators, the valid permutations are 27 per operator set. With two non-commuting operators in a single tree structure, four operator combinations per tree and two valid trees yield 216 combinations. Allowing less complex equations, all the previous permutations are also valid, bringing the total to at least 240. Thus, one can safely assume that the total number of permutations is well above 200. Yet only the same single combination has the correct output dimension. The effect of this constraint is expected to be on a few orders of magnitude for a total number of operators and operands $\approx 10^1$. Note that this constraint applies on top of the above syntactical constraints. Combining both could potentially limit the search by several orders of magnitude.

5.6 Inference on limited data

The discussions on complexity, biases and search space have not yet touched upon the issue of limited data. Several theorems used in data analysis only hold in the large N limit, a large amount of samples. In practical science, the number of data points is finite and, in the case of expensive experiments or observations, few compared to the large N limit.

One good example is the BIC, which approaches the log probability of the maximum likelihood estimate asymptotically in N [70], in addition to two constraints. Likewise, the Shannon entropy will approach the average prefix complexity for large N [82]. While the mathematical structure only holds in large N the tools seem useful even in small or medium N. But one question remains:

Is it possible to decisively make the correct inference on limited data?

In the MDL framework, the basic complexity of the data set is simply the cost of encoding each observed value. Given a single data point, it is unlikely that one could create a program simpler than storing the value, even if the relationship is simple. If one extends this to two points, one is still unlikely to find a shorter program even though both the points now "help" each other to encode the other. In practical terms, if one knows that the data follows some polynomial, then for two points the linear polynomial would offer a perfect fit with lower complexity even if the data follows a higher-order distribution. Thus, It is obvious that there isn't sufficient evidence to use a complex model in the MDL framework in the low data regime. A complex model requires a complex data set to justify, even if it is the "true" model.

In the Bayesian framework, a single data point is most effectively encoded as a single zero:th order polynomial. Given a model $y_i = f_j(x_i) + \epsilon_j$ with Gaussian error:

$$p(\boldsymbol{\theta}|D) = \frac{1}{\sqrt{2\pi\theta_0^2}} \frac{e^{-\frac{1}{2}(\theta_1 - y_0)^2/\theta_0^2}}{\int p(D|\theta)p(\theta)d\theta} p(\theta) \tag{95}$$

For simplicity, limit the prior standard deviation and constant to two independent uniform distributions in $]0, 10]$. Upon observing a single value at $y = 5$ the posterior distribution of the data is:

$$p(\boldsymbol{\theta}|D) = \frac{e^{-\frac{1}{2}(\theta_1 - 5)^2/\theta_0^2}}{\sqrt{2\pi\theta_0^2}} \frac{1}{C} \frac{1}{100}, \tag{96}$$

where C is a normalizing constant. By inspection we deduce that the posterior is maximized by setting the mean to the observed value, irrespective of the standard deviation. With θ_1 fixed, maximizing $p(\boldsymbol{\theta}|D)$ means minimizing θ_0. In the limit of $\lim_{\theta_0 \to 0}$, the Gaussian becomes a Dirac delta distribution, meaning the point is most effectively stored as the value itself. For a higher-order model, the maximal likelihood will be unchanged since both fit perfectly, but their prior volume and parametric volume will be larger, penalizing the solution through Occam's factor.

Unsurprisingly, both MDL and the Bayesian framework hold that simple data can not justify a complex model. This is in itself a strong indication that the answer to the question is no; it is not possible to make decisive inferences on limited data. The argument also applies to figure 18 in section 5.2. The models become distinguishable if more data is sampled between the observed points. However, if the underlying model is not part of the hypothesis space, then no amount of data will find the correct relationship.

Several contemporary symbolic regressors use neural networks as part of their algorithm. The networks usually require a large training dataset and, depending on the application, might need large amounts of specific data. In AI Feynman,

the neural network used for testing is trained on 80,000 samples. Similarly, the symmetry search algorithm implemented in our paper uses 1000 points to learn the linear motion relationship. This data needs to be considered part of the process, even if the data used for inference is just a subset of the samples. Any approach which relies on this large pool can not constitute a solution to, for instance, the Nguyen equations. For clarity, one should clearly separate two training cases:

1. Trained on auxiliary data.
2. Trained on problem data.

The first case should be considered perfectly acceptable in the benchmark since the training data is disconnected from the problem and would constitute *general knowledge*. The second case constitutes *problem-specific knowledge* and offers an unfair advantage in one specific task in limited data inference. In general, when solving problems, the second kind is unavailable to us especially when considering expensive experimental data.

6 Conclusion

In this thesis, we investigated the foundational principles researchers use to derive and formulate testable hypotheses. The hypothesis generation process is a fundamental part of the scientific method and arguably at the core of understanding. The process is, however, not formalized to the degree that it can be implemented into an algorithmic approach. Nor is it clear from the literature if it can be formulated algorithmically. Therefore, this thesis studied the hypothesis generation process in physics to investigate the possibility of formulating an algorithmic approach to hypothesis generation. If such an algorithm could be formulated, it would likely significantly accelerate the scientific progress.

In the process, we investigated three well-established concepts for their use as inductive biases in the algorithmic process: symmetry, entropy and minimum description length. In chapters 2 - 3, we introduce the concepts and their respective properties. Using the concepts, we implemented two algorithms, one for symmetry searching in chapter 3 and one for symbolic regression in chapter 4. Finally, we discuss the implications and results in chapter 5. Below, we will briefly summarize the work, and conclusions as well as provide an outlook for future work and potential applications of the results.

6.1 Concepts for inductive biases

The minimum description length is a robust framework to apply Occam's razor in practice. In summary, MDL holds that a theory that maximally compresses the data is preferred. The concepts used are well researched and has, with some limitation, good practical approximations, see section 2.4.1. While the framework provides a clear path in several constrained cases, the implementation might require further clarification for a general solution using the method.

The second concept of entropy is widely used in science and has a close connection to MDL through Shannon entropy. Shannon entropy relates the probabilistic descriptions of systems to the information contained in them, with the maximum entropy when all the allowable states have uniform probability. The maximum entropy principle is application-dependent but generally holds that one should adopt a view such that each point of data is maximally informative.

Symmetry is an interesting and ubiquitous tool in science, present in almost all fields. The main property of symmetry in hypothesis generation is that it constrains the relationship in the variable space. A known symmetry gives insight into the full form of the theory, as any theory can be expressed in terms of its symmetries [40]. The approach and importance of symmetries in theory building is widely recognized and the process of symmetry inference is actively researched.

In an attempt to uncover other underlying structures, the connection between the concepts was also investigated. The link between MDL and entropy is strong through the noiseless coding theorem, as well as through optimal coding and

Shannon entropy. Symmetry is defined in terms of group theory and differs significantly from MDL and entropy. In the case of statistical mechanics, there is a clear link between symmetry and entropy. Given a macrostate of a system, the entropy is related to the equivalent microstates or equivalent permutations. These equivalent permutations are correlated to symmetry transformations on the state, see section 5.4.

6.2 Symmetry searching algorithm

To construct a symmetry-searching algorithm we, follow the implementation of [50]. The algorithm uses machine learning to identify the symmetric manifold present in a Hamiltonian system. In the process, the algorithm seems to infer the correct distributions of data compared to the reference paper, but we could not regress the symbolic relationships. Learning a manifold representation seems to be a viable alternative in cases with enough data to train a neural network. Similar approaches can be found in the literature [53].

The general principles used will likely form the basis of a viable symmetry-searching algorithm in more general cases. In the current form, the algorithm samples important relationships which need to be analyzed by other algorithms. As such, the algorithm might grow hand-in-hand with the symbolic regressors used for inferring the specific relationships.

6.3 General symbolic regressor

In chapter 4, we implemented a symbolic regression algorithm, inspired by Monte Carlo tree search, to serve as a general hypothesis generator. The regressor builds a tree of options, representing different components of an equation. By enforcing syntactically correct expressions in the tree using rules, and evaluating the resulting equations' performance, each result guides further exploration of the equations. Similar symbolic regressors exist in literature [65].

In our algorithm, we implemented four novel ideas for the regressor which we denote: quick evaluation, subtree limitation, active learning and the MDL. The ideas attempt to promote parsimony, limit the search space and use data efficiently, principles related to the inductive biases explored earlier.

The first two ideas refer to the growth strategy of the algorithm, quick evaluation attempting to grow the tree such that it can be evaluated readily and subtree limitation attempting to stop any exploration in space which is syntactically valid but logically invalid. The strategy enforces correct syntax in the rules for growth and limits how many components each guess may grow. Since the tree uses postfix notation, any valid subexpression is preserved in the tree. If a subexpression is invalid then so is any expression containing that subtree, meaning it can be blocked from further exploration. The growth strategy does not cover the full equation space currently and requires some changes to become general. Both strategies combined potentially improved the efficiency of the

algorithm.

The active learning component used the variance between the top solutions to generate interesting positions and then update the tree with the new data. The new positions would be highly informative for distinguishing the top solutions and provide new data which guides further growth. The approach did not seem to provide significant improvements in resolving the underlying function, or in scoring.

Using MDL as the scoring mechanism allowed for a natural restriction on the length of the generated expressions and gave smaller expressions as a result. However, The MDL interacted with the UCT score in the sense that long but correct expressions were penalized. The UCT score of the solution in the correct case is dominated by the description length, lowering the effective score. This means the code would focus more on exploration as the solutions became longer, which was an unintended but predictable change.

The regressor does seem to be an improvement over traditional MCTS for symbolic regression, see section 4.8, at least for the evaluated case. While it is not competitive with contemporary algorithms such as AI Feynman or SymPhysLearn in its current configuration, several clear paths for improvement could potentially close the gap.

6.4 Outlook

The main question of the thesis is if it is possible to automate the scientific process with a specific focus on hypothesis generation. At this point, the thesis has not identified any obstacle to such an approach beyond being challenging. As such, we hypothesize that there likely exists a pathway towards a general hypothesis generator, starting with generators that suggest partial theories or constraints such as symmetries.

For instance, given a trajectory, a basic symbolic regressor can be used to infer a model potential used in Hamiltonian dynamics, as demonstrated in the thesis. This could be used for finding effective potentials, which is useful both from a modelling and inference perspective.

Problems with large data sets available could start by identifying important inherent symmetries. Using a trained neural network, a researcher may identify regions where the problem likely is invariant to specific changes in a subset of the variables. Using this limitation the high-dimensional problem might be decomposed into a set of smaller problems which are easier to solve.

6.5 Open questions

The following questions arose during the thesis and would be interesting to investigate further.

Unit augmented MCTS

Can the MCTS be augmented to only sample syntactically and unit-valid space? By limiting the search space in some intelligent manner, the efficiency of each evaluation would be much improved.

Scarce data symmetry search

Among the investigated symmetry search algorithms, there is an abundance of methods which rely on large amounts of data. An algorithm that could indicate the potential presence of symmetries in the small data regime would be highly interesting.

Complexity of universally valid compressors

The Hamiltonian dynamics framework allows for efficient compression of any trajectory into a force descriptor and a set of initial conditions. If one uses this to compress a single sampled trajectory the compression might be mediocre or negative. The compression can be increased by simply including more samples from the same trajectory or by including another trajectory. How would one evaluate such a functional compressor in the MDL/complexity framework? See section 5.1 for a longer explanation.

References

[1] D. R. White, S. Yoo, and J. Singer, "The programming game: Evaluating mcts as an alternative to gp for symbolic regression," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15, Madrid, Spain: Association for Computing Machinery, 2015, pp. 1521–1522, ISBN: 9781450334884. DOI: 10.1145/2739482.2764655. [Online]. Available: https://doi.org/10.1145/2739482.2764655.

[2] M. Valipour, M. Panju, B. You, and A. Ghodsi, "Symbolicgpt: A generative transformer model for symbolic regression," in *NeurIPS ENLSP*, 2022. [Online]. Available: https://neurips2022-enlsp.github.io/accepted_papers.html.

[3] S.-M. Udrescu and M. Tegmark, "Ai feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, eaay2631, 2020. DOI: 10.1126/sciadv.aay2631. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.aay2631. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.aay2631.

[4] M. Longair, "'... a paper ... i hold to be great guns': A commentary on maxwell (1865) 'a dynamical theory of the electromagnetic field'," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2039, p. 20 140 473, 2015. DOI: 10.1098/rsta.2014.0473. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2014.0473. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2014.0473.

[5] G. Gutting, "Einstein's discovery of special relativity," *Philosophy of Science*, vol. 39, no. 1, pp. 51–68, 1972, ISSN: 00318248, 1539767X. [Online]. Available: http://www.jstor.org/stable/186592.

[6] S. Banerji, "How einstein discovered the special theory of relativity," *Resonance*, vol. 11, pp. 27–42, 2 Feb. 2006. DOI: 10.1007/BF02837273. [Online]. Available: https://doi.org/10.1007/BF02837273.

[7] H. R. Crane, "The energy and momentum relations in the beta-decay, and the search for the neutrino," *Rev. Mod. Phys.*, vol. 20, pp. 278–295, 1 Jan. 1948. DOI: 10.1103/RevModPhys.20.278. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.20.278.

[8] M. Riordan, "Pauli's ghost: The conception and discovery of neutrinos," in *Current Aspects of Neutrino Physics*, D. O. Caldwell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1–16, ISBN: 978-3-662-04597-8. DOI: 10.1007/978-3-662-04597-8_1. [Online]. Available: https://doi.org/10.1007/978-3-662-04597-8_1.

[9] A. Einstein, "Physics reality," *Daedalus*, vol. 132, no. 4, pp. 22–25, 2003, ISSN: 00115266. [Online]. Available: http://www.jstor.org/stable/20027877.

[10] *Scientific Method, The Hypothetico-Experimental Laboratory Procedure of the Physical Sciences*, 1st ed. Springer Dordrecht, 2012, ISBN: 978-94-010-2758-8. DOI: https://doi.org/10.1007/978-94-010-2758-8.

[11] I. B. Cohen, "Newton's discovery of gravity," *Scientific American*, vol. 244, no. 3, pp. 166–181, 1981, ISSN: 00368733, 19467087. [Online]. Available: http://www.jstor.org/stable/24964334.

[12] A. C. Benjamin, "The mystery of scientific discovery," *Philosophy of Science*, vol. 1, no. 2, pp. 224–236, 1934, ISSN: 00318248, 1539767X. [Online]. Available: http://www.jstor.org/stable/184391.

[13] B. C. Jantzen, "Discovery without a 'logic' would be a miracle," *Synthese*, vol. 193, no. 10, pp. 3209–3238, 2016, ISSN: 00397857, 15730964. [Online]. Available: http://www.jstor.org/stable/24897999.

[14] P. Kosso, "Empirical testing," in *A Summary of Scientific Method*. Dordrecht: Springer Netherlands, 2011, pp. 13–20, ISBN: 978-94-007-1614-8. DOI: 10.1007/978-94-007-1614-8_3. [Online]. Available: https://doi.org/10.1007/978-94-007-1614-8_3.

[15] P. Gregory, *Markov chain Monte Carlo*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511791277.013.

[16] F. James, *Statistical Methods in Experimental Physics*, 2nd ed. World Scientific Publishing Co. Pte. Ltd., Singapore, 2006, ISBN: 978-9812705273.

[17] J. Skilling, "Classic maximum entropy," in *Maximum Entropy and Bayesian Methods: Cambridge, England, 1988*, J. Skilling, Ed. Dordrecht: Springer Netherlands, 1989, pp. 45–52, ISBN: 978-94-015-7860-8. DOI: 10.1007/978-94-015-7860-8_3. [Online]. Available: https://doi.org/10.1007/978-94-015-7860-8_3.

[18] P. K. S. P. Gurfil, *Celestial Mechanics and Astrodynamics: Theory and Practice*. Springer-Verlag Berlin Heidelberg, 2016, pp. 1–23, ISBN: 978-3-662-50370-6.

[19] N. R. Hanson, "The mathematical power of epicyclical astronomy," *Isis*, vol. 51, no. 2, pp. 150–158, 1960, ISSN: 00211753, 15456994. [Online]. Available: http://www.jstor.org/stable/226846.

[20] C. Nordling and J. Österman, "Physics handbook for science and engineering," in 9th. Studentlitteratur AB, Lund, 2021, ch. M-12, pp. 449–452, ISBN: 978-91-44-128061.

[21] P. K. S. P. Gurfil, *Celestial Mechanics and Astrodynamics: Theory and Practice*. Springer-Verlag Berlin Heidelberg, 2016, pp. 95–119, ISBN: 978-3-662-50370-6.

[22] "Nasa planetary fact sheet - metric." Accessed: 22-12-30. (), [Online]. Available: https://nssdc.gsfc.nasa.gov/planetary/factsheet/.

[23] E. T. Jaynes, "Model comparison," in *Probability Theory: The Logic of Science*, G. L. Bretthorst, Ed. Cambridge University Press, 2003, pp. 601–614. DOI: 10.1017/CBO9780511790423.022.

[24] C. Nordling and J. Österman, "Physics handbook for science and engineering," in 9th. Studentlitteratur AB, Lund, 2021, ch. F-1.1, ISBN: 978-91-44-128061.

[25] S. Shalev-Shwartz and S. Ben-David, "Preface," in *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014, pp. xv–xvi. DOI: 10.1017/CBO9781107298019.004.

[26] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.010.

[27] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997. DOI: 10.1109/4235.585893.

[28] P. Gregory, "The how-to of bayesian inference," in *Bayesian Logical Data Analysis for the Physical Sciences: A Comparative Approach with Mathematica® Support*. Cambridge University Press, 2005, pp. 41–72. DOI: 10.1017/CBO9780511791277.013.

[29] *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Science+Business Media, LLC, 2008, ISBN: 978-0-387-49820-1. DOI: 10.1007/978-0-387-49820-1.

[30] S. Devine, "The insights of algorithmic entropy," *Entropy*, vol. 11, no. 1, pp. 85–110, Mar. 2009, ISSN: 1099-4300. DOI: 10.3390/e11010085. [Online]. Available: http://dx.doi.org/10.3390/e11010085.

[31] B. E, "Information, and symmetry: Ordered is symmetrical," *Entropy*, vol. 22, 1 2019. DOI: https://doi.org/10.3390/e22010011.

[32] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x.

[33] E. T. Jaynes, "Repetitive experiments: Probability and frequency," in *Probability Theory: The Logic of Science*, G. L. Bretthorst, Ed. Cambridge University Press, 2003, pp. 270–313. DOI: 10.1017/CBO9780511790423.011.

[34] D. V. Schroeder, "An introduction to thermal physics," in Oxford University Press, Oxford, 2021, pp. 74–84, ISBN: 978-0-19-289555-4. DOI: 10.1093/oso/9780192895547.001.0001.

[35] W. H. Zurek, "Algorithmic randomness and physical entropy," *Phys. Rev. A*, vol. 40, pp. 4731–4751, 8 Oct. 1989. DOI: 10.1103/PhysRevA.40.4731. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.40.4731.

[36] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. DOI: 10.1109/JRPROC.1952.273898.

[37] P. D. Grünwald, *The Minimum Description Length Principle*. The MIT Press Cambridge, Massachusetts, 2007, ISBN: 9780262256292.

[38] D. J. Bartlett, H. Desmond, and P. G. Ferreira, "Exhaustive symbolic regression," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023, ISSN: 1941-0026. DOI: 10.1109/TEVC.2023.3280250.

[39] R. J. Solomonoff, "The Kolmogorov Lecture The Universal Distribution and Machine Learning," *The Computer Journal*, vol. 46, no. 6, pp. 598–601, Jan. 2003, ISSN: 0010-4620. DOI: 10.1093/comjnl/46.6.598.

eprint: https://academic.oup.com/comjnl/article-pdf/46/6/598/1070220/460598.pdf. [Online]. Available: https://doi.org/10.1093/comjnl/46.6.598.

[40] L. L. V. Kreinovich, "Unreasonable effectiveness of symmetry in physics," *International Journal of Theoretical Physics*, pp. 1549–1555, 7. DOI: https://doi.org/10.1007/BF02084960.

[41] J. Schwichtenberg, *Physics from Symmetry*, 2nd ed. Springer Cham, 2017, ISBN: 978-3-319-66631-0. DOI: https://doi.org/10.1007/978-3-319-66631-0.

[42] "Symmetry in science: An introduction to the general theory," in Springer New York, NY, 1995, ISBN: 978-1-4612-2506-5.

[43] S. De Haro and J. Butterfield, "On symmetry and duality," *Synthese*, vol. 198, no. 4, pp. 2973–3013, Apr. 2021.

[44] J. Rosen, "The symmetry principle," *Entropy*, vol. 7, no. 4, pp. 308–313, 2005, ISSN: 1099-4300. DOI: 10.3390/e7040308. [Online]. Available: https://www.mdpi.com/1099-4300/7/4/308.

[45] V. I. Arnold, *Mathematical Methods of Classical Mechanics*, 2nd ed. Springer New York, NY, 2013, ISBN: 978-1-4757-2063-1. DOI: https://doi.org/10.1007/978-1-4757-2063-1.

[46] J. Schwichtenberg, "Physics from symmetry," in 2nd ed. Springer Cham, 2017, pp. 38–50, ISBN: 978-3-319-66631-0. DOI: https://doi.org/10.1007/978-3-319-66631-0.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[48] A. Heinecke, J. Ho, and W.-L. Hwang, "Refinement and universal approximation via sparsely connected relu convolution nets," *IEEE Signal Processing Letters*, vol. 27, pp. 1175–1179, 2020. DOI: 10.1109/LSP.2020.3005051.

[49] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, 4 1989. DOI: 10.1007/BF02551274. [Online]. Available: https://doi.org/10.1007/BF02551274.

[50] Y.-i. Mototake, "Interpretable conservation law estimation by deriving the symmetries of dynamics from trained deep neural networks," *Phys. Rev. E*, vol. 103, p. 033 303, 3 Mar. 2021. DOI: 10.1103/PhysRevE.103.033303. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.103.033303.

[51] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[52] "Intro to autoencoders." (), [Online]. Available: https://www.tensorflow.org/tutorials/generative/autoencoder (visited on 03/19/2023).

[53] Z. Liu and M. Tegmark, "Machine learning conservation laws from trajectories," *Phys. Rev. Lett.*, vol. 126, p. 180 604, 18 May 2021. DOI: 10.1103/

PhysRevLett.126.180604. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.126.180604.

[54] N. Dehmamy, R. Walters, Y. Liu, D. Wang, and R. Yu, "Automatic symmetry discovery with lie algebra convolutional network," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 2503–2515.

[55] J. Yang, R. Walters, N. Dehmamy, and R. Yu, "Generative adversarial symmetry discovery," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, Honolulu, Hawaii, USA: JMLR.org, 2023.

[56] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn, "Noether networks: Meta-learning useful conserved quantities," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 16 384–16 397. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/886ad506e0c115cf590d18ebb6c26561-Paper.pdf.

[57] J. G. Borgqvist and S. Palmer, "Occam's razor gets a new edge: The use of symmetries in model selection," *Journal of The Royal Society Interface*, vol. 19, no. 193, p. 20 220 324, 2022. DOI: 10.1098/rsif.2022.0324. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2022.0324. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2022.0324.

[58] M. Virgolin and S. P. Pissis, "Symbolic regression is NP-hard," *Transactions on Machine Learning Research*, 2022, ISSN: 2835-8856. [Online]. Available: https://openreview.net/forum?id=LTiaPxqe2e.

[59] Z. Liu, V. Madhavan, and M. Tegmark, "Machine learning conservation laws from differential equations," *Physical Review E*, vol. 106, no. 4, Oct. 2022. DOI: 10.1103/physreve.106.045307. [Online]. Available: https://doi.org/10.1103%2Fphysreve.106.045307.

[60] F. Sun, Y. Liu, J.-X. Wang, and H. Sun, "Symbolic physics learner: Discovering governing equations via monte carlo tree search," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=ZTK3SefE8_Z.

[61] J. B. Orjuela-Quintana, S. Nesseris, and W. Cardona, "Using machine learning to compress the matter transfer function $T(k)$," *Phys. Rev. D*, vol. 107, p. 083 520, 8 Apr. 2023. DOI: 10.1103/PhysRevD.107.083520. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.107.083520.

[62] S. A. and S. R., "A systematic review of explainable artificial intelligence models and applications: Recent developments and future trends," *Decision Analytics Journal*, vol. 7, p. 100 230, 2023, ISSN: 2772-6622. DOI: https://doi.org/10.1016/j.dajour.2023.100230. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S277266222300070X.

[63] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark, "Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph

modularity," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 4860–4871. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/33a854e247155d590883b93bca53848a-Paper.pdf.

[64] M. Cranmer, *Interpretable machine learning for science with pysr and symbolicregression.jl*, 2023. arXiv: 2305.01582 [astro-ph.IM].

[65] T. Cazenave, "Nested monte-carlo expression discovery," in *European Conference on Artificial Intelligence*, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:9757779.

[66] P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, "End-to-end symbolic regression with transformers," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=GoOuIrDHG_Y.

[67] N. Q. Uy and M. Hoai Nguyen Xuan andO'Neill, "Semantically-based crossover in genetic programming: Application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, 2 Jun. 2011. DOI: 10.1007/s10710-010-9121-2.

[68] "Feynman symbolic regression database." Retrieved 23-09-12. (), [Online]. Available: https://space.mit.edu/home/tegmark/aifeynman.html.

[69] C. L. Hamblin, "Translation to and from Polish Notation," *The Computer Journal*, vol. 5, no. 3, pp. 210–213, Nov. 1962, ISSN: 0010-4620. DOI: 10.1093/comjnl/5.3.210. eprint: https://academic.oup.com/comjnl/article-pdf/5/3/210/1172943/5-3-210.pdf. [Online]. Available: https://doi.org/10.1093/comjnl/5.3.210.

[70] P. Stoica and Y. Selen, "Model-order selection: A review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 36–47, 2004. DOI: 10.1109/MSP.2004.1311138.

[71] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293, ISBN: 978-3-540-46056-5.

[72] M. de Waard, D. M. Roijers, and S. C. Bakkes, "Monte carlo tree search with options for general video game playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016, pp. 1–8. DOI: 10.1109/CIG.2016.7860383.

[73] A. Tharwat and W. Schenck, "A survey on active learning: State-of-the-art, practical challenges and research directions," *Mathematics*, vol. 11, no. 4, p. 820, Feb. 2023, ISSN: 2227-7390. DOI: 10.3390/math11040820. [Online]. Available: http://dx.doi.org/10.3390/math11040820.

[74] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[75] A. Meurer, C. P. Smith, M. Paprocki, *et al.*, "Sympy: Symbolic computing in python," *PeerJ Computer Science*, vol. 3, e103, Jan. 2017, ISSN: 2376-

5992. DOI: 10.7717/peerj-cs.103. [Online]. Available: https://doi.org/10.7717/peerj-cs.103.

[76] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[77] M. Kommenda, A. Beham, M. Affenzeller, and G. Kronberger, "Complexity measures for multi-objective symbolic regression," in *Computer Aided Systems Theory – EUROCAST 2015*, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., Cham: Springer International Publishing, 2015, pp. 409–416, ISBN: 978-3-319-27340-2.

[78] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959. DOI: 10.1109/TEC.1959.5222693.

[79] "Symmetry in science: An introduction to the general theory," in Springer New York, NY, 1995, pp. 145–150, ISBN: 978-1-4612-2506-5.

[80] S.-K. Lin, "Correlation of entropy with similarity and symmetry," *Journal of Chemical Information and Computer Sciences*, vol. 36, no. 3, pp. 367–376, 1996. DOI: 10.1021/ci950077k. eprint: https://doi.org/10.1021/ci950077k. [Online]. Available: https://doi.org/10.1021/ci950077k.

[81] "An introduction to kolmogorov complexity and its applications," in Springer Science+Business Media, LLC, 2008, pp. 382–401, ISBN: 978-0-387-49820-1. DOI: 10.1007/978-0-387-49820-1.

[82] E. Desurvire, "Algorithmic entropy and kolmogorov complexity," in *Classical and Quantum Information Theory: An Introduction for the Telecom Scientist*. Cambridge University Press, 2009, pp. 96–126. DOI: 10.1017/CBO9780511803758.009.

A Appendix A - Mathematics

Binomial identities

Given the binomial coefficient, $\binom{n}{k}$ the relation to its neighbouring coefficients can be found as:

$$\binom{n}{k} \equiv \frac{n!}{k!(n-k)!} \rightarrow \binom{n-1}{k-1} = \frac{(n-1)!}{(k-1)!(n-k)!}.$$

Since $n! = n(n-1)!$ one can extract the largest factors in the $n!$ and $k!$ and rewrite:

$$\binom{n-1}{k-1} = \frac{k}{n}\binom{n}{k}.$$

Similarly, to move one step in n, one may identify the change in factors:

$$\binom{n-1}{k} = \frac{(n-1)!}{k!(n-k-1)!} = \frac{n-k}{n}\binom{n}{k}.$$

Using both of these relations, one can show that $\binom{n/2}{k/2}$ always is smaller than $\binom{n}{k}$ unless $k = 0, n$. Let $n, k \in \mathrm{N}$ be two even numbers where $k \leq n/2$. Since $k < n$: $k/2 < n/2$, there are fewer integer steps to move to $k/2$ than to $n/2$. Using the first relation above, one may move to the intermediate stage by repeated application:

$$\binom{n-1}{k-1} = \frac{k}{n}\binom{n}{k} \rightarrow \binom{n-k/2}{k/2} = \left(\prod_{i=0}^{k/2} \frac{k-i}{n-i}\right)\binom{n}{k}.$$

Using the second relation, we may change n to the sought $n/2$ by repeated application:

$$\binom{n-1}{k} = \frac{n-k}{n}\binom{n}{k} \rightarrow \binom{n/2}{k/2} = \left(\prod_{j=0}^{n/2-k/2} \frac{n-k-j}{n-k/2-j}\right)\binom{n-k/2}{k/2}$$

Putting both together, one may write:

$$\binom{n/2}{k/2} = \left(\prod_{i=0}^{k/2} \frac{k-i}{n-i}\right)\left(\prod_{j=0}^{n/2-k/2} \frac{n-k-j}{n-k/2-j}\right)\binom{n}{k}.$$

Since $k \leq n/2$, all individual factors in both series are smaller than one and bigger than zero except in the edge case $k = 0$ when both binomial coefficients are one. Showing that $\binom{n/2}{k/2} \leq \binom{n}{k}$. This can be extended to $k > n/2$ by substituting $k = n - k$ since $\binom{n}{k}$ is symmetric around $n/2$.

B Appendix B - Symbolic regression benchmarks

Nguyen

The Nguyen benchmark [67] contains ten equations: eight univariate and two bivariate.

Equation	Data distribution	# of points
$x^3 + x^2 + x$	$U(-1, 1)$	20
$x^4 + x^3 + x^2 + x$	$U(-1, 1)$	20
$x^5 + x^4 + x^3 + x^2 + x$	$U(-1, 1)$	20
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	$U(-1, 1)$	20
$\sin x^2 * \cos x - 1$	$U(-1, 1)$	20
$\sin x + \sin(x + x^2)$	$U(-1, 1)$	20
$\log(x + 1) \log(x^2 + 1)$	$U(0, 2)$	20
$\sqrt{x}$	$U(0, 4)$	20
$\sin(x + y^2)$	$U(-1, 1) \times U(-1, 1)$	100
$2 * \sin x * \cos y$	$U(-1, 1) \times U(-1, 1)$	100

Table 1: Nguyen benchmark equations with data parameters. $U(a, b)$ denotes a uniform distribution with lower limit a and upper limit b.

Feynman

The Feynman benchmark [68] is based on equations from the Feynman lectures on physics and contains 100 univariate and multivariate equations. Below, we list all the equations used in the paper. All equations in the Feynman tests used 20 points.

Equation	Variables	Data
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2}}}{2\sqrt{\pi}}$	θ	$U(1,3)$
$\frac{\sqrt{2}e^{-\frac{\theta^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	σ, θ	$U(1,3)^2$
$\frac{\sqrt{2}e^{-\frac{(\theta-\theta_1)^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$	σ, θ, θ_1	$U(1,3)^3$
$\sqrt{(-x_1+x_2)^2+(-y_1+y_2)^2}$	x_1, x_2, y_1, y_2	$U(1,5)^4$
$\frac{Gm_1m_2}{(-x_1+x_2)^2+(-y_1+y_2)^2+(-z_1+z_2)^2}$	$m_1, m_2, G, x_1, x_2, y_1, y_2, z_1, z_2$	$U(1,2)^{1)}$, $U(3,4)^{2)}$
$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$	m_0, v, c	$U(1,5), U(1,2)$, $U(3,10)$
$x_1y_1+x_2y_2+x_3y_3$	$x_1, x_2, x_3, y_1, y_2, y_3$	$U(1,5)^6$
$N_n\mu$	N_n, μ	$U(1,5)^2$
$\frac{m\left(u^2+v^2+w^2\right)}{2}$	m, v, u, w	$U(1,5)^4$
$\frac{q_1q_2}{4\pi\epsilon r^2}$	q_1, q_2, ϵ, r	$U(1,5)^4$

Table 2: First 10 Feynman equations with variable and data distributions used. $U(a,b)^n$ denotes a uniform distribution with lower limit a, upper limit b and n is the number of consecutive variables using the same limit. Limits have the same ordering as the listing order. $^{1)}$ Limit applies to m_1, m_2, G and positional variables with subscript 2. $^{2)}$ Limit applies to positional variables with subscript 1.

C Appendix C - Symbolic regressor output

Below is the full output of the symbolic regression algorithm with all UCT graphs and top equations. Refer to the implementation chapter for information on the graphs and tables. Note that all equations in the tables are simplified using Sympy for presentation purposes. The exact output from the algorithm may be slightly different (`x1+1-` = `x`).

Nguyen equations

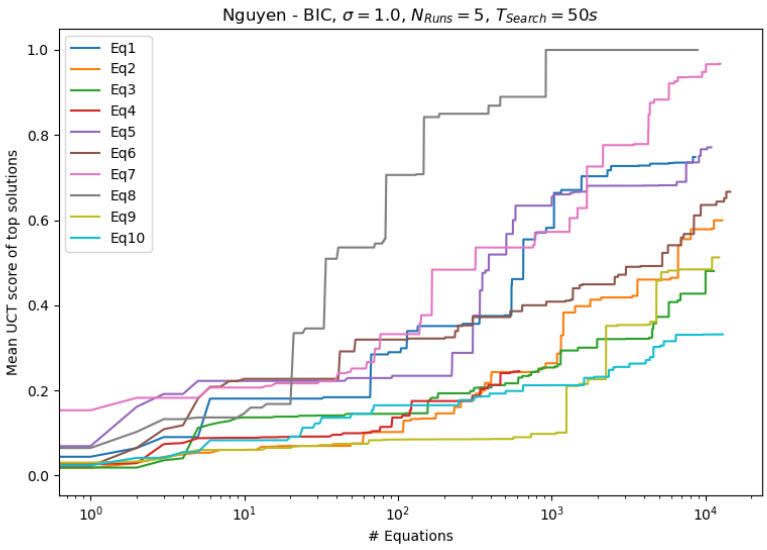

Figure 1: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using BIC scoring.

Nguyen equation search using BIC

Eq.	1st	2nd	3rd	4th	5th
1	$\frac{(e^x-1)^2}{x}$	$3^x x$	$3^x x$	$3^x x$	$x\left(x+2\right)^x$
2	$x\left(x+3\right)^x$	$\left(x+\cos\left(\cos\left(3\right)\right)\right)^3$	$x\left(x+3\right)^x$	$x\left(\pi^x x+1\right)$	$x^2\left(x+1\right)^2$
3	$2\cdot 2^x x$	π^x-1	$e^{2^x}-e$	$x\left(x+3\right)^x$	$x\left(x+3\right)^x$
4	$a^x x$	x^2e^{2x}	$-x+e^{2x}-1$	$2x^2e^x+x$	$\pi^x x$
5	$-1+\cos^e\left(1\right)$	$\cos\left(x-3\right)$	$\sin^3\left(\cos\left(x\right)-e\right)$	$-x+e^x-2$	$\sin\left(\frac{\sin\left(e\right)}{\pi}-1\right)$
6	$e^{\sin\left(xe^{\frac{1}{2}}\right)}-1$	$2\sin\left(\pi^x-1\right)$	$x\sqrt{x+e}$	$2\sin\left(\pi^x-1\right)$	$2\sin\left(e^x-1\right)$
7	$\frac{x}{e}+x$	$x+\sin\left(x\sin\left(e\right)\right)$	$\frac{ex}{2}$	$\frac{ex}{2}$	$\frac{x}{e}+x$
8	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$
9	$x+y^4$	$x+\frac{\log\left(3\right)}{3}$	$x+y\sin\left(y\right)$	$x+\sin\left(y^2\right)$	$x+\sin\left(y^2\right)$
10	$\frac{\pi x}{2}$	$x+\sin\left(\log\left(\pi^{\frac{x}{2}}\right)\right)$	$x\cos\left(1\right)+x$	$\frac{ex\log\left(\pi\right)}{2}$	$-x+e\sin\left(x\right)$

Table 1: Top solutions to the Nguyen equations for a search of $50s$ using BIC scoring.

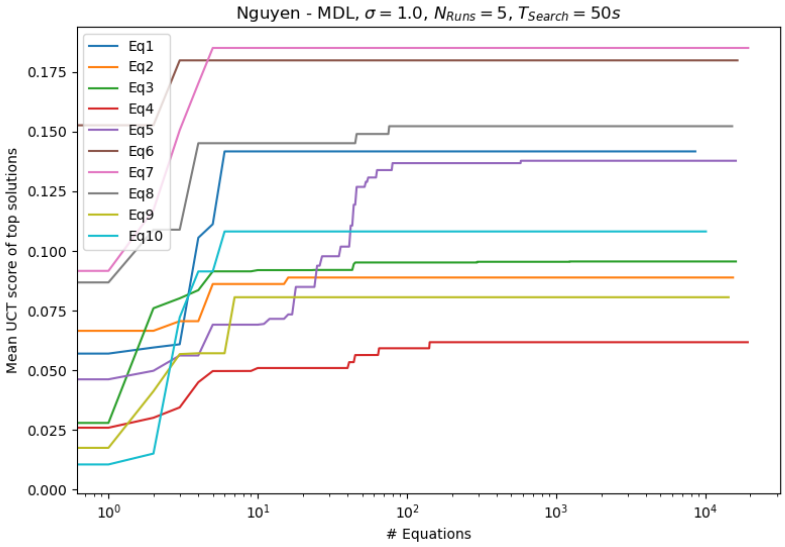

Figure 2: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using MDL scoring.

Nguyen equation search using MDL

Eq #.	1st	2nd	3rd	4th	5th
1	x	x	x	x	x
2	x	x	x	x	e^x
3	xe^x	$x + e^x$	e^x	x	x
4	e^x	e^x	x	e^x	e^x
5	$\cos(e)$	$\cos(e)$	$\cos(e)$	$\cos(e)$	$\cos(e)$
6	x	x	x	x	x
7	x	x	x	x	x
8	$\sqrt{x}$	1	1	1	$\sqrt{x}$
9	x	x	x	x	x

Table 2: Top solutions to the Nguyen equations for a search of $50s$ using MDL scoring.

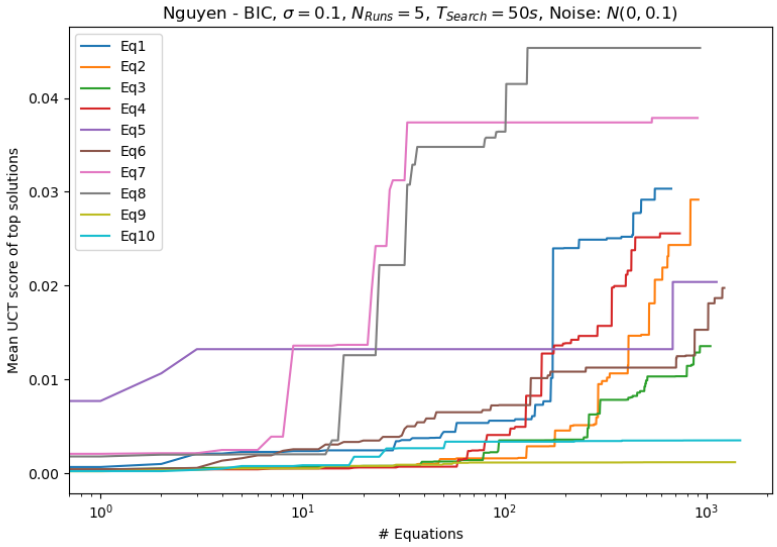

Figure 3: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using BIC scoring with added Gaussian noise (N(0,0.1)).

Nguyen equation search using BIC with noisy data

Eq #.	1st	2nd	3rd	4th	5th
1	$\frac{3(a^x-1)}{\pi}$	$a\left(x+\frac{xe^x}{a}\right)$	$-a+x\left(3^x+a\right)$	$\frac{x(3^x+a)}{2a}$	$x+e^{-a+x(x+2)}$
2	$\frac{axe^{ax}}{2}$	$\frac{ex(a^x)^e}{3}$	$-x+e\left(\frac{a^x}{2}-\frac{1}{2}\right)$	$x\left(a+\frac{xe^x}{a}\right)$	$\frac{xe^{a+x}}{3}-\frac{x}{3}$
3	$ax\left(a+xe^x\right)$	$x\left(-x+\sqrt{a^x}\right)$	$x\left(e^{-a+e^x}\right)^a$	$x^2e^{\frac{x}{a}}+x$	a^xx^2+x
4	$\frac{e\left(\frac{e^{ax}}{3}-\frac{1}{3}\right)}{\pi}$	$\frac{x+\frac{e^{ax}}{3}}{e}$	$\frac{e^{-a+3x}-1}{\pi}$	$\frac{2(a^x-1)}{3e}$	$\frac{x(e^a)^x+x}{a}$
5	$ea\cos\left(a-x+e\right)$	$\sin\left(a-x+e\right)$	$\sin\left(a+x-1\right)$	$2a-\cos\left(a+x\right)$	$\frac{\pi\cos\left(a-x\right)}{a}$
6	$\frac{a\sin\left(3^x-1\right)}{3}-\frac{x}{3}$	$\frac{a\sin\left(\sin\left(e^x-1\right)\right)}{2}-x$	$\frac{a\sin\left(e^x-1\right)}{3\pi}$	$2\sin\left(\sin\left(2^xx\right)\right)$	$-\frac{x}{2}+\frac{\sin\left(e^x-1\right)}{2a}$
7	$\left(\frac{4ax}{3}\right)^a$	$\left(ax+x\right)^{3a}$	$\frac{-a+\frac{x}{ea}}{e}$	$x\left(x+1\right)^a$	$\left(\frac{ax}{e}+x\right)^a$
8	$\sqrt{x}$	$\sqrt{x}$	$\left(xe^a\right)^{3a}$	$a\left(\frac{\sqrt{x}}{a}\right)^a$	$\sqrt{x}$
9	$-a+\sin\left(x\right)-\pi$	$a^2\sin\left(\frac{x}{2}+\frac{y^2}{2}\right)$	$\frac{x+\frac{y^2}{a}}{a^2}$	$-a+x+1+e$	$x+e^{-a+y}$
10	$2\pi\sin\left(x\sqrt{e^a}\right)$	$a\sin\left(x\right)$	$a\sin\left(\sin\left(\sin\left(x\right)\right)\right)+x$	$-x+\frac{\sin\left(x\right)}{a}$	$3\sin\left(a\sin\left(x\right)\right)$

Table 3: Top solutions to the Nguyen equations for a search of $50s$ using BIC scoring with added Gaussian noise (N(0,0.1)).

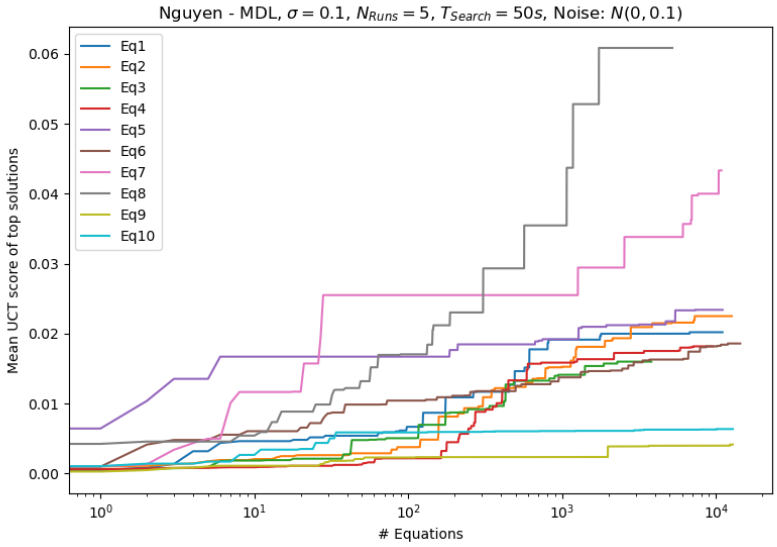

Figure 4: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using MDL scoring with added Gaussian noise (N(0,0.1)).

Nguyen equation search using MDL with noisy data

Eq #.	1st	2nd	3rd	4th	5th
1	xe^x	$a^x - 1$	$a^x - 1$	$a^x - 1$	xe^x
2	xe^{ax}	$a^x x$	$a^x x$	$a^x x$	$x^2 e^x + x$
3	$e\left(a + x\right)^3$	$e^{\sqrt{2}x} - 1$	$\frac{a^x}{2} - \frac{1}{2}$	$a^x x^2 + x$	$x\sqrt{a^x - x}$
4	$e^{a^x x} - 1$	$a^x x$	$\frac{a^x x + x}{e}$	$a^x x$	$x\sqrt{a^x}$
5	$\cos\left(a + x\right)$	$\sin\left(2\log\left(\cos\left(\cos\left(x\right)\right)\right)\right)$	$\cos\left(a + x\right)$	$\cos\left(a - x\right)$	$\sin\left(2\sin\left(\cos\left(2\right)\right)\right)$
6	$a\sin\left(3^x - 1\right)$	$x\sqrt{x + e}$	$e^{\sin\left(xe^{\frac{1}{2}}\right)} - 1$	$x\left(x + e^{\frac{1}{2}}\right)$	$2 \cdot 2^x - 2$
7	$\sqrt{2}x$	$\sqrt{2}x$	$\frac{ex}{2}$	$x\sin\left(e\right) + x$	$2x\log\left(2\right)$
8	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$
9	$a + \sin\left(x\right)$	$x + \frac{\cos\left(x\right)}{3}$	$x + \sin\left(y^2\right)$	$x + \sqrt{y^2}$	$-a + \sin\left(x\right)$
10	$2\sin\left(\sin\left(\sin\left(x\right)\right)\right)$	$\frac{3x}{2}$	$x + \log\left(2^x\right)$	$x + \frac{\sin\left(x\right)}{2}$	$2\sin\left(\sin\left(\sin\left(x\right)\right)\right)$

Table 4: Top solutions to the Nguyen equations for a search of $50s$ using MDL scoring with added Gaussian noise (N(0,0.1)).

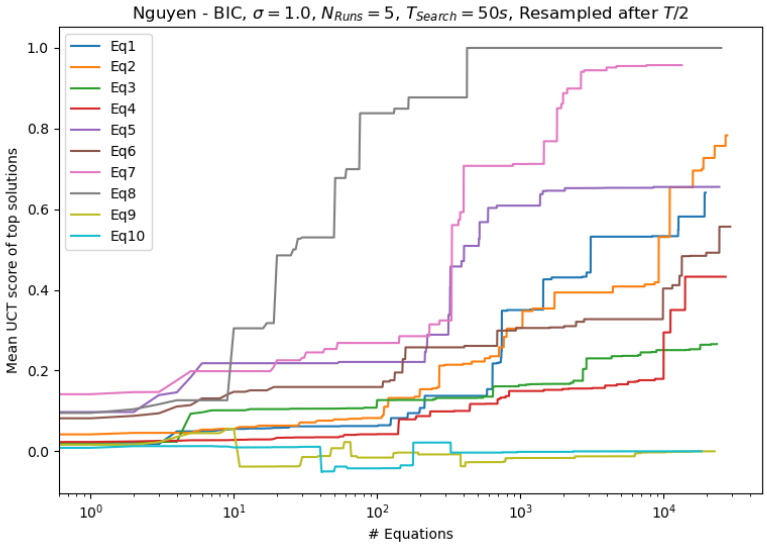

Figure 5: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using BIC scoring with resampling after $25s$ with the top 10 solutions used for active learning.

Nguyen equation search using BIC with resampling

Eq #.	1st	2nd	3rd	4th	5th
1	$(e^x)^{\frac{e}{2}} - 1$	$(e^x)^{\frac{e}{2}} - 1$	$(e^x)^{\frac{e}{2}} - 1$	$(x+3)^x - 1$	$2x + 2\sin(3)$
2	$3^x x$	$x(x+3)^x$	$2^{2x}x$	$2^{2x}x$	$2^{2x}x$
3	$-a + 2x^3 + x + 3$	$a^{2x}x^2 + x$	$\pi^x - 1$	$2(a+x)^3$	$3(a+x)^3$
4	$x + (3^x - 1)^2$	$-x + e^{2x} - 1$	$x(\pi x + 1)$	$xe^{\sqrt{\pi}x}$	$x(a + \pi x)$
5	$\sin\left(\frac{x}{3} - 1\right)$	$\sin(2\sin(\cos(2)))$	$\sin\left(\log\left(\cos\left(\sqrt[4]{2}\right)\right)\right)$	$\sin\left(\log\left(\frac{\sin(1)}{2}\right)\right)$	$-\sin(\sin(1))$
6	$e^{\sin\left(\sqrt{\pi}x\right)} - 1$	$2\sin(2^x x)$	$2x\cos(x)$	$x + \sin\left(\frac{3^x}{2}\right)$	$e^{\sin(2x)} - 1$
7	$\frac{x}{e} + x$	$e\log\left(\sqrt{e^x}\right)$	$2x\log(2)$	$\frac{ex}{2}$	$x + \log\left(e^{\frac{x}{e}}\right)$
8	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$
9	$x\left(-2a + y^2\right)$	$\frac{x + \frac{y}{a}}{y}$	$y(-3a + 3y - e)$	$\frac{y(a+y)}{\pi}$	$\frac{a((y+\pi)^x - 1)}{y}$
10	$\frac{a + e^{y^2} - 1}{x}$	$\frac{3a\left(\frac{x}{2} + \frac{y^2}{2}\right)}{y}$	$\frac{\pi^x}{ay}$	$\frac{a + \left(y^2\right)^\pi}{x}$	$y\left(-a + \sin\left(y^2\right) + 3\right)$

Table 5: Top solutions to the Nguyen equations for a search of $50s$ using BIC scoring and resampling at $25s$ with the top 10 solutions used for active learning.

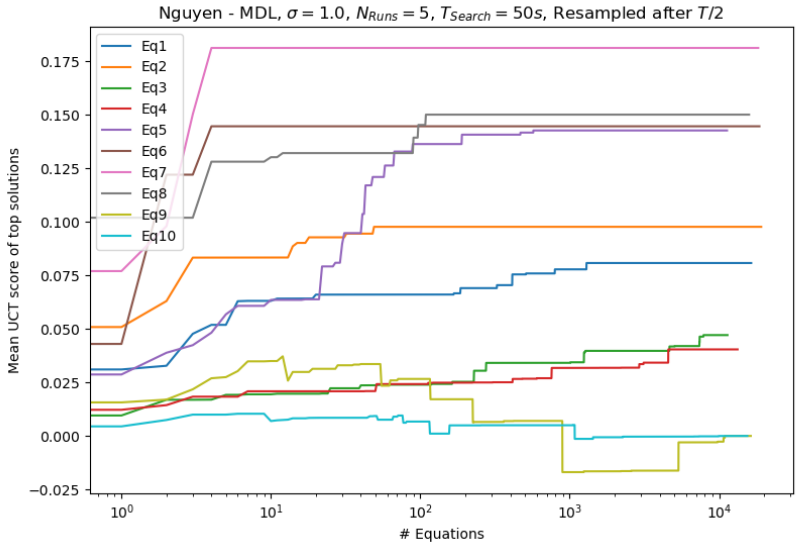

Figure 6: Average UCT score of top solution as a function of number of evaluations for five independent runs. Nguyen equations with a search of $50s$ using MDL scoring with resampling after $25s$ with the top 10 solutions used for active learning.

Nguyen equation search using MDL with resampling

Eq #.	1st	2nd	3rd	4th	5th
1	x	xe^x	xe^x	xe^x	x
2	e^x	e^x	x	x	e^x
3	$\frac{3^{3x}}{8}$	$\frac{e^{ex}}{\pi}$	$x\,(x+e)$	x	$\frac{e^{ex}}{3}$
4	$e^{x^2}-1$	e^x	e^x	$(x+2)^x$	e^{e^x-2}
5	$\cos(e)$	$\cos(e)$	$\cos(e)$	$\cos(e)$	$\cos(e)$
6	x	x	x	x	x
7	x	x	x	x	x
8	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$
9	$\frac{xe^{\frac{1}{2}}}{a}$	$\frac{-a+\cos(e)}{y}$	$\frac{a}{3y}$	$\frac{a}{xy}$	$\frac{a}{y}$
10	$\frac{a+\frac{y}{a}}{x}$	$y\,(a-x)$	$\frac{a^3y^3}{x}$	$\frac{ae^y}{3y}$	$\frac{\pi^y}{ay}$

Table 6: Top solutions to the Nguyen equations for a search of $50s$ using MDL scoring and resampling at $25s$ with the top 10 solutions used for active learning.

Feynman equations 1-10

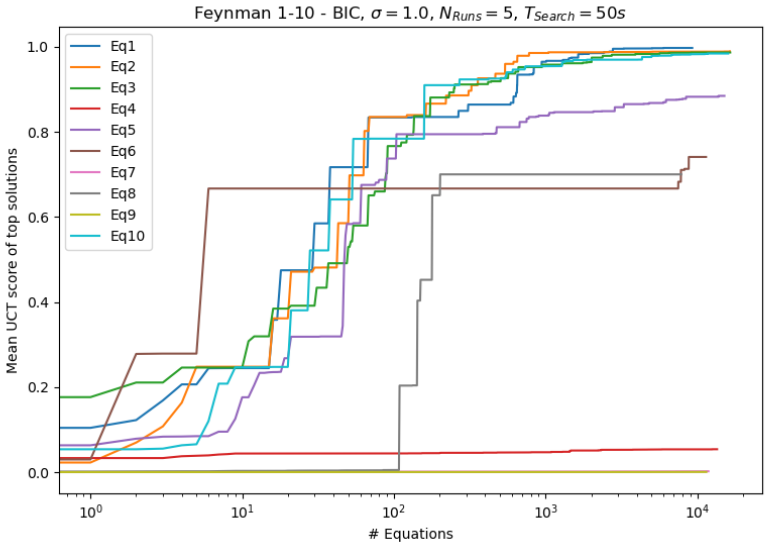

Figure 7: Average UCT score of top solution as a function of number of evaluations for five independent runs. Feynman equations 1-10 with a search of $50s$ using BIC scoring.

Feynman equation search using BIC

Eq #.	1st	2nd	3rd	4th	5th
1	$\left(\frac{\log(2)}{3}\right)^{\theta}$	$\sqrt{\left(\sin^{\theta}(e)\right)^{\pi}}$	$\left(\frac{\sqrt{\cos\left(\sqrt{2}\right)}}{e^{\frac{1}{2}}}\right)^{\theta}$	$\sin^{\pi\theta}(\log(2))$	$\sin^{\pi\theta}(\log(2))$
2	$\frac{\sin^2(\theta)}{e^2}$	$\sin\left(\frac{\cos^e(1)}{\theta}\right)$	$\frac{\cos^{\theta}(1)}{3}$	$\frac{\log\left(\sqrt{\pi}\right)^{\theta}}{\pi}$	$\sin\left(\frac{\sin(e)}{2\theta}\right)$
3	$\sin\left(\frac{1}{\sigma e^{-\cos(3)}}\right)$	$\frac{e\sin(3)}{\sigma}$	$\frac{\log(2)^{\sigma}}{e}$	$\frac{\cos\left(\sqrt[4]{2}\right)}{\sigma}$	$\frac{\log\left(\sqrt{2}\right)}{\sigma}$
4	$e\cos(\cos(y_2))$	$\cos(y_1)+e$	$\frac{x_2+e}{x_2}$	$e^{\cos(y_1)}+1$	$2+\frac{1}{y_2}$
5	$\frac{2G}{x_1 z_1}$	$\frac{m_2 x_2}{e y_1}$	$x_2\sin\left(\log\left(\sqrt{m_2}\right)\right)$	$\frac{Gm_2}{3e}$	$\frac{m_1 m_2}{3e}$
6	$m_0-1+\log(\pi)$	m_0	$m_0+\log(\log(\pi))$	$m_0-1+\sqrt{\log(\pi)}$	m_0
7	$par_1+x_3+y_1y_3+y_2$	$\sqrt{par_1y_1y_3}+e$	$x_2+\frac{x_3(y_2+y_3)}{par_1}$	$(x_2+y_1+e)^{par_1}$	$x_1+2(y_1+\pi)^{par_1}$
8	$N_n\mu$	$\pi N_n\mu par_1$	$N_n\mu$	$N_n\mu$	$N_n\left(\frac{N_n}{par_1}+\mu\right)$
9	$mpar_1w+v$	$v(m+u)^{par_1}+w$	$m(v+w)^{par_1}$	$\frac{u(mw+v)}{par_1}$	$\pi\left(\frac{muv}{par_1}+par_1\right)$
10	$\frac{\left(\frac{\log(q_1)}{q_1}\right)^{r}}{e}$	$\sqrt{\sin^{\pi r}(e)}$	$\frac{\log(q_2)^2}{q_2^2 r}$	$\frac{2}{e\epsilon r^2}$	$\frac{\sin(\cos^r(\log(3)))}{\epsilon}$

Table 7: Top solutions to the first ten Feynman equations for a search of $50s$ using BIC scoring.

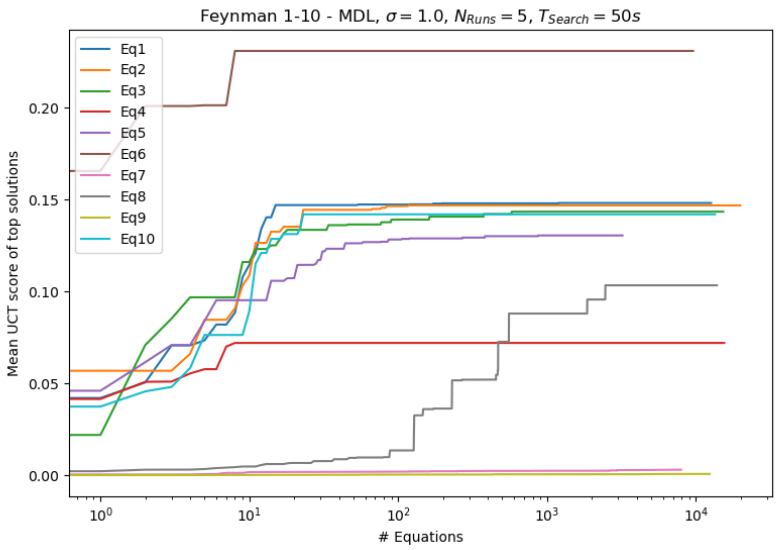

Figure 8: Average UCT score of top solution as a function of number of evaluations for five independent runs. Feynman equations 1-10 with a search of $50s$ using MDL scoring.

Feynman equation search using MDL

Eq #.	1st	2nd	3rd	4th	5th
1	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$
2	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$
3	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$	$\sin(3)$
4	2	e	2	2	2
5	$\log(m_2)$	$\sin(3)$	$\log(x_2)$	$\sin(e)$	$\log(m_1)$
6	m_0	m_0	m_0	m_0	m_0
7	$x_1 + \sqrt{par_1 x_2}$	$par_1 + x_1 + x_2 y_2$	$\pi(-par_1 + x_1 + y_3)$	$par_1 + x_3 y_3 + y_2$	$3par_1 + 3y_2 + 3y_3$
8	$N_n \mu$	$N_n \mu$	$N_n \mu$	$N_n \mu$	$N_n \mu$
9	$(mpar_1 + w)^{par_1}$	$u(mpar_1 + v)$	$m(par_1 w + u)$	$\frac{mvw}{par_1}$	$\pi\left(\frac{mw}{par_1} + u\right)$
10	0	0	0	0	0

Table 8: Top solutions to the first ten Feynman equations for a search of $50s$ using MDL scoring.

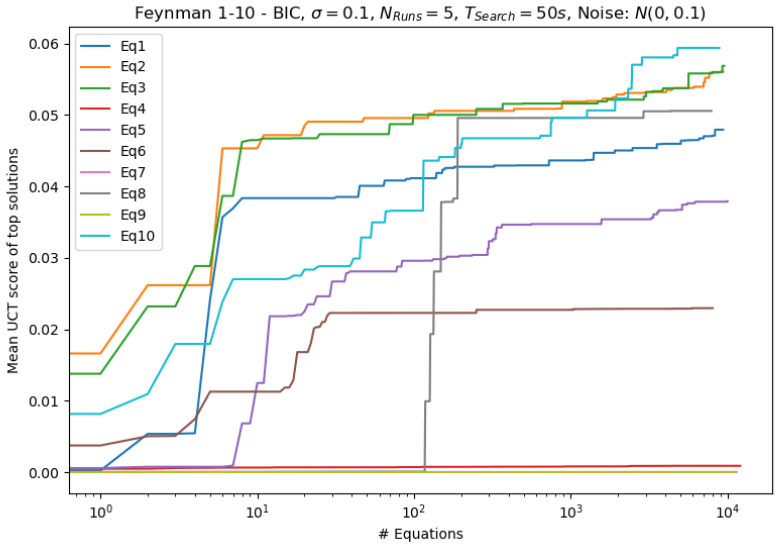

Figure 9: Average UCT score of top solution as a function of number of evaluations for five independent runs. Feynman equations 1-10 with a search of $50s$ using BIC scoring with added Gaussian noise (N(0,0.1)).

Feynman equation search using BIC with noisy data

Eq #.	1st	2nd	3rd	4th	5th
1	$\frac{\sin^2(\cos(1))}{\theta^2}$	$\frac{par_1\theta+1}{\pi}$	$\frac{3\cos^3(1)}{\theta^3}$	$\sin^\theta\left(\sin\left(\frac{\cos(1)}{e}\right)\right)$	$\left(\frac{e^{\frac{par_1}{\theta}}}{2}\right)^\theta$
2	$\frac{(e^{par_1})^\theta}{\pi}$	$\log(\log(par_1)^\sigma)^\theta$	$\sin^\theta\left(\frac{\sin(1)}{e}\right)$	$\frac{\sin(\cos^e(1))}{\theta}$	$\frac{par_1 e^{par_1\theta}}{\theta}$
3	$\frac{\cos(\sin(1))}{2\sigma_{x_1}}$	$\frac{\cos^{\frac{\pi}{2}}(1)}{\sigma}$	$par_1+\frac{par_1}{\sigma\theta}$	$\sin^{3\pi}(1)$	$\frac{\theta_1\cos^2(1)}{e}$
4	$\frac{e^{par_1^{x_1}}}{x_1}$	$par_1+\frac{x_1}{x_2}-e$	$e^{\frac{y_2}{par_1 y_1}}$	$-par_1+x_1^{par_1}$	$\frac{par_1(\cos(y_2)+2)}{3}$
5	$\frac{Gm_2par_1}{y_1}$	$x_2\cdot\left(2par_1+\frac{2}{x_1}\right)$	$\frac{Gpar_1x_2}{e}$	$\frac{z_2\sin(G)}{3x_2}$	$\frac{Gm_2}{2z_1}$
6	$par_1(m_0+1)-1$	$m_0\sqrt{(e^{par_1})^{m_0}}+par_1$	$m_0\sqrt{par_1^{m_0}}$	$m_0par_1par_1^{m_0}$	$par_1(m_0+2par_1-2)$
7	$\frac{x_3+y_3+\pi}{par_1}$	$2x_2+\frac{x_3+y_3}{par_1}$	$2par_1y_3+2y_2$	$3x_1+3y_1+3y_2$	$par_1(y_1+y_3)+x_3$
8	$6N_n\mu par_1+par_1$	$N_n\mu$	$N_n\mu par_1^3par_1^\mu$	$N_n\mu$	$\left(\frac{N_n^{par_1}\mu}{par_1^3}\right)^{par_1}$
9	$m(par_1+v)+w$	$\frac{m\sqrt{w}}{par_1}$	$muw-par_1+v$	$2m(par_1+u)-3$	$v(mu-par_1)$
10	$(q_1e^{par_1})^\epsilon$	$\frac{par_1^2}{4\epsilon^2r^2}$	$\frac{\left(\frac{1}{r}\right)^\epsilon}{r}$	$\left(\frac{\sin(1)}{3}\right)^r$	$\sqrt{e^{\frac{par_1\cos(1)}{q_2}}}$

Table 9: Top solutions to the first ten Feynman equations for a search of $50s$ using BIC scoring with added Gaussian noise (N(0,0.1).

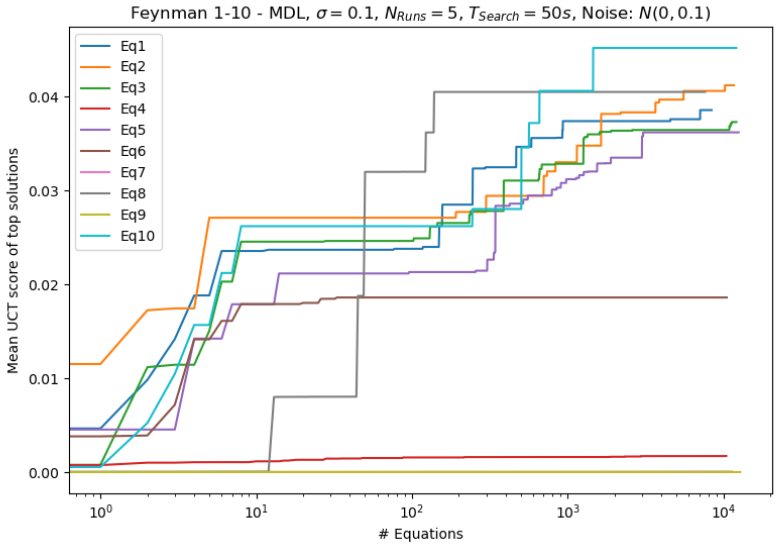

Figure 10: Top UCT score as a function of number of evaluations. Average score of top solution for five independent runs. Feynman equations 1-10 with a search of $50s$ using MDL scoring with added Gaussian noise (N(0,0.1)).

Feynman equation search using MDL with noisy data

Eq #.	1st	2nd	3rd	4th	5th
1	$\left(\frac{\sin(2)}{e}\right)^{\theta}$	0	0	$\cos^2\left(\sqrt{\theta}\right)$	0
2	$\cos^3(1)$	0	$\frac{\sin(e)}{\pi}$	$\cos\left(\sqrt{2}\right)$	$\cos^3(1)$
3	$\frac{\sin(2)}{e\sigma}$	$\frac{\cos(1)}{3}$	$\frac{\sin(e)}{2}$	$\frac{\log\left(\sqrt{2}\right)}{\sigma}$	$\frac{\cos(1)}{3}$
4	$e^{\frac{par_1 x_1}{x_2}}$	$par_1 y_2 (y_1 + 3)$	$e^{\frac{par_1 x_2}{x_1}}$	$\frac{par_1^{y_1}}{y_1}$	$\log(\pi)^{2y_1}$
5	$\log\left(\sqrt{m_2}\right)$	$\frac{\log(m_2)}{m_2}$	$\frac{1}{x_1}$	$\frac{\sin(y_2)}{y_1}$	$\frac{1}{z_1}$
6	m_0	m_0	$m_0^{par_1}$	$m_0^{par_1}$	$m_0^{par_1}$
7	$par_1(y_1 + y_2) + x_2 + y_1$	$\pi(par_1 + x_2 + y_1)$	$x_3 + (\pi(x_1 + y_2 + 3))^{par_1}$	$-2par_1 + 2x_2 y_1 + 2x_3$	$e(-par_1 + x_1 + x_3 + y_3)$
8	$N_n \mu$	$N_n \mu$	$N_n \mu$	$N_n(\mu + par_1)$	$N_n \mu par_1$
9	$par_1 + w(mv + \pi)$	$m(u + v)^{par_1}$	$\frac{muw}{par_1}$	$mpar_1\sqrt{v}$	$m(u + v)^{par_1} + 3$
10	0	0	0	0	0

Table 10: Top solutions to the first ten Feynman equations for a search of $50s$ using MDL scoring with added Gaussian noise (N(0,0.1)).

Hamiltonian solutions

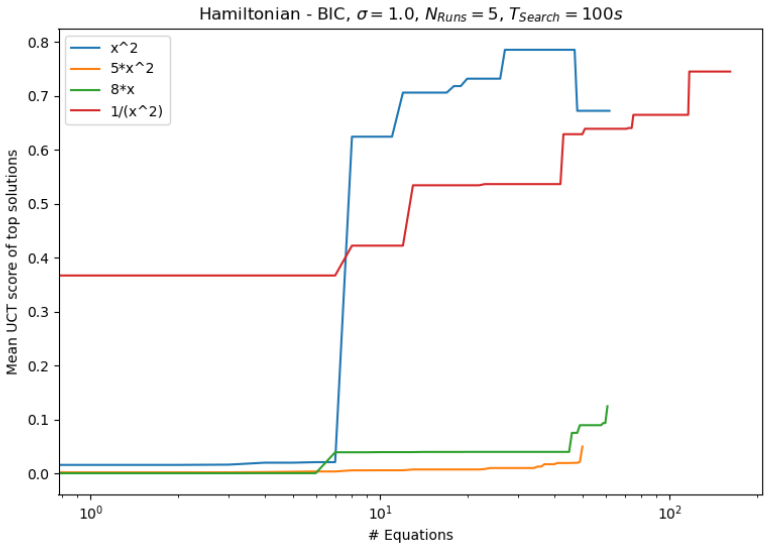

Figure 11: Average UCT score of top solution as a function of number of evaluations for five independent runs. Hamiltonian exploration for $100s$ using BIC scoring, $\sigma = 1.0$.

Hamiltonian potential search using BIC

Potential	1st	2nd	3rd	4th	5th
x^2	$2x$	$\pi^3 x^3$	$\log\left(x^2-3\right)$	x^2	$-m+x^2-2$
$5x^2$	$\frac{a^x}{e}$	$e^{\frac{x}{a}}$	$\left(a^x\right)^x$	ex^2	$\sqrt{a^x}$
$8x$	$\frac{ax}{m}$	x^3+x-2	a^2x	$ax+m$	$a\left(-x+e\pi x\right)-\pi$
$\frac{1}{x^2}$	$\frac{\sin(m)}{x}$	$\frac{m+\cos(3)}{x}$	$\frac{\sqrt{m}}{x}$	$\frac{\sin(2)}{x}$	$\sin(m-x)+2$

Table 11: Top solutions of listed Hamiltonian potentials for a search of $100s$ for five independent runs using BIC scoring. a denotes free parameter.

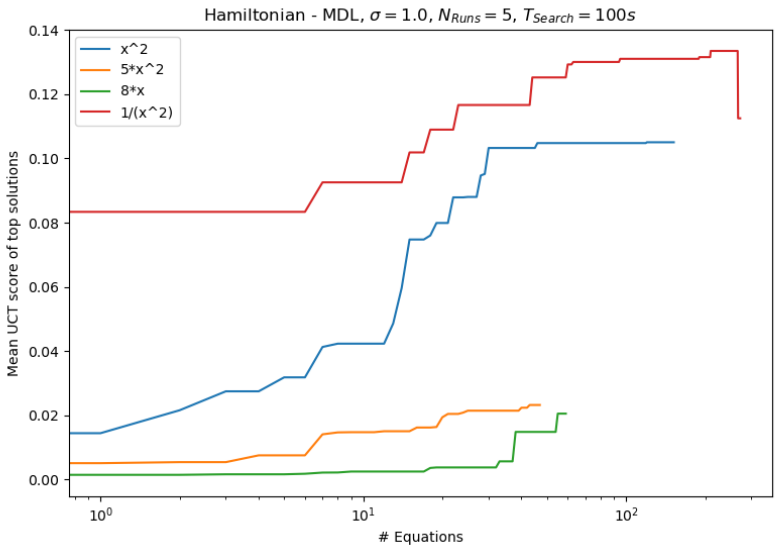

Figure 12: Average UCT score of top solution as a function of number of evaluations for five independent runs. Hamiltonian exploration for $100s$ using BIC scoring, $\sigma = 1.0$.

Hamiltonian potential search using MDL

Potential	1st	2nd	3rd	4th	5th
x^2	x^2	x^2	x^2	x^2	x^2
$5x^2$	ex	$4x^2$	π^x	3^x	π^x
$8x$	ax	ax	$\frac{x}{a}$	$\log(\cos(x))$	ax
$\frac{1}{x^2}$	m	$\log(2-x)$	$\log(\cos(x))$	$\frac{2}{x}$	$\frac{m}{x}$

Table 12: Top solutions of listed Hamiltonian potentials for a search of $100s$ for five independent runs using MDL scoring. a denotes free parameter.