MI-HET-???
July 15, 2025

Machine learning the Conformal Manifold of Holographic CFT2's

Bastien Duboeuf[1], Camille Eloy[1] and Gabriel Larios[2]

[1] *ENS de Lyon, CNRS, LPENSL, UMR5672, 69342, Lyon cedex 07, France*

[2] *Mitchell Institute for Fundamental Physics and Astronomy, Texas A&M University, College Station, TX, 77843, USA*

Abstract

...

Contents

1 Introduction

The AdS/CFT correspondence [1] stands as one of the most profound dualities in theoretical physics, establishing a remarkable equivalence between gravitational theories in Anti-de Sitter (AdS) spacetimes and conformal field theories (CFTs) living on their boundaries. This correspondence has revolutionized our understanding of both quantum gravity and strongly coupled field theories, providing unprecedented insights into the holographic nature of gravity.

Within this holographic framework, supergravity theories in AdS backgrounds serve as the low-energy effective descriptions of string theory compactifications, making them natural laboratories for exploring the gravitational side of the duality. Studying the CFT side is very interesting : CFT play a crucial role in statistical physics where they are known to describe phase transitions ; they are also fixed point of Renormalisation Group flow and this is this latter point that is our interest in this paper. An interesting question is indeed to know wether those CFT are isolated fixed points of RG flows, or belong to a continous family of CFT's. If this is the case, the space of deformations that takes from one CFT to another is called the conformal manifold. They are parametrized by exactely marginal operator, or in other words operators whose β functions exactly vanish.

From this perspective, the AdS/CFT correspondence relates conformal manifolds on the boundary theory to flat directions in the scalar potential of the dual supergravity theory. Along these directions, the scalar field configurations vary continuously while the cosmological constant remains fixed. In general, supersymmetry is believed to be necessary for the existence of holographic conformal manifolds, as non-supersymmetric AdS solutions are typically expected to be unstable [2,3]. However, recent investigations [4] have identified AdS_4 configurations that appear to evade this requirement, with no evidence of standard

decay channels—neither perturbative nor non-perturbative—being present. In the context of AdS_3/CFT_2, the situation may be even richer. A well-known counterexample has existed for some time [5, 6], which is well understood from both the field theory and gravitational perspectives. More generally, within the supergravity approximation, continuous deformations correspond to classically marginal operators. These operators become exactly marginal only if their marginality persists at finite values of both the rank N and the CFT coupling. Recent work [7] has demonstrated the existence of families of marginal deformations that are perturbatively stable, using the framework of Exceptional Field Theory (ExFT).

In this work, we propose a novel approach, where we study directly the scalar potential of supergravity. As already mentionned, the classical marginal deformations of CFT's in the large N limit, are in correspondance with the flat directions in the potential, i.e. when there is a continuous set a scalar fields such that $\nabla V = 0$. However, the explicit characterization of these flat directions presents formidable technical challenges. Supergravity scalar potentials, even in truncated models, typically involve dozens of scalar fields with intricate non-linear interactions. The resulting expressions for critical points—where all first derivatives vanish—quickly become too complex for traditional symbolic manipulation, rendering analytical approaches computationally intractable.

The emergence of machine learning techniques in theoretical physics opens new avenues for addressing such complex problems. Instead of solving the full symbolic system analytically from the outset, one can employ numerical methods to sample the solution space and subsequently apply symbolic regression techniques to extract analytical patterns from the data. This hybrid approach has the potential to bypass the computational bottlenecks inherent in purely symbolic methods, while still having the potential to uncover exact analytical expressions.

Machine learning strategies have previously been applied to identify new isolated vacua in SO(8) supergravity [8, 9]. More broadly, there has been increasing interest in applying machine learning and numerical techniques across various domains of high-energy physics. This includes, for instance, the characterization of Calabi–Yau metrics and hypersurfaces [10–17], as well as broader efforts to explore the string theory landscape using machine learning techniques [18–20]. Additional applications include studies in CFT [21], investigations of the supergravity landscape [22, 23], and explorations of the AdS/CFT correspondence [24]. More generally, machine learning has found utility in the study of string theory, geometry, and fundamental physics [25–27].

In this work, we demonstrate the viability of the aformentionned machine learning approach by applying it to a five-scalar subsector of a 13 scalar consistent truncation of six-dimensional non-chiral $\mathcal{N} = (1, 1)$ supergravity on $AdS_3 \times S^3$, or to type IIB supergravity on $AdS_3 \times S^3 \times T^4$. Our methodology combines gradient descent sampling of the flat direction manifold with some symbolic regression technique. There exist a large litterature on symbolic regression, using methods from genetic programming [28] such as [29–31], to deep machine learning [32, 33], generative modelling [34], diffusion models [35], and equation learner with the nodes being symbolic operations [36]. Another state-of-the art algorithm is the AIFeynman methods [37]. They use neural network to identify structure in the dataset (such as translational symmetries, multiplicative separability, compositionality, ...) to recursively define simpler problems on which they can fit the solutions with polynomials. However, in the spirit of trying to build a generalisable method, we develop here our own algorithm, which is based on symbolic regression using an Annealing Importance Sampling method (AIS) [38]. In this paper, we will study a 5 parameter truncation, which is itself coming from a 13 parameter consistent truncation of $\mathcal{N} = (1, 1)$ 6-dimensional supergravity.

The study of this 5 dimensional parameter potentiel is separated into several parts. We first use the gradient descent to efficiently sample the underlying conformal manifold. Combined with numerical analysis,

such as principal component analysis and clustering, we can identify the existence of a 3-dimensional continuous family. We then use an AIS technique combined with a sequential Monte Carlo approach (SMC) [39] in order to do symbolic regression. As we will indeed demonstrate in the paper, there exist polynomial constraints on the 5 parameters, viewed as embedding coordinates, that projects them on the 3 dimensional manifolds. We manage to identify 8 of those constraints on the data, not all independent, which once solved gives that two of the embedding coordinates can be expressed in terms of the others. This provides an explicit 3 dimensional parameters family

The paper is organized as follows. Section 2 establishes the supergravity setup, presenting the scalar potential in its full complexity and motivating the truncation to five fields. Section 3 details our numerical methodology, covering both the gradient descent sampling procedure and the annealed importance sampling approach to symbolic regression. Finally, Section 4 concludes with prospects for extending this approach to higher-dimensional cases and its broader implications for systematic studies of conformal manifolds in holographic theories.

2 Supergravity setup

Three-dimensional $\mathcal{N}=8$ (half-maximal) gauged supergravity is governed by the Lagrangian

$$e^{-1}\mathcal{L} = R + \frac{1}{8} g^{\mu\nu} D_\mu M^{\bar{M}\bar{N}} D_\nu M_{\bar{M}\bar{N}} + e^{-1}\mathcal{L}_{\rm CS} - V\,, \tag{2.1}$$

which comprises the Einstein-Hilbert term R, the kinetic term for scalar fields parametrized by the matrix $M_{\bar{M}\bar{N}}$, a Chern-Simons contribution $\mathcal{L}_{\rm CS}$, and a scalar potential V.

The gauging structure is encoded in an embedding tensor that takes the general form

$$\Theta_{\bar{K}\bar{L}|\bar{M}\bar{N}} = \theta_{\bar{K}\bar{L}\bar{M}\bar{N}} + \frac{1}{2}\Big(\eta_{\bar{M}[\bar{K}}\theta_{\bar{L}]\bar{N}} - \eta_{\bar{N}[\bar{K}}\theta_{\bar{L}]\bar{M}}\Big) + \theta\,\eta_{\bar{M}[\bar{K}}\eta_{\bar{L}]\bar{N}}\,. \tag{2.2}$$

where we have a fully antisymmetric component $\theta_{\bar{K}\bar{L}\bar{M}\bar{N}} = \theta_{[\bar{K}\bar{L}\bar{M}\bar{N}]}$, a symmetric traceless part $\theta_{\bar{L}\bar{K}} = \theta_{(\bar{L}\bar{K})}$, and a constant parameter θ. The metric $\eta_{\bar{K}\bar{L}}$ is the SO(8,4)-invariant bilinear form used for index contractions.

The scalar degrees of freedom parametrize the coset space

$$\frac{\mathrm{SO}(8,4)}{\mathrm{SO}(8)\times\mathrm{SO}(4)}\,, \tag{2.3}$$

through the symmetric matrix $M_{\bar{K}\bar{L}} = \mathcal{V}_{\bar{K}}{}^{\bar{M}}\mathcal{V}_{\bar{L}}{}^{\bar{N}}\delta_{\bar{M}\bar{N}}$, where $\mathcal{V}_{\bar{K}}{}^{\bar{M}}$ represents the coset representative.

The gauge covariant derivatives are constructed using the embedding tensor according to

$$D_\mu = \partial_\mu + A_\mu{}^{\bar{M}\bar{N}}\,\Theta_{\bar{M}\bar{N}|\bar{P}\bar{Q}}\,T^{\bar{P}\bar{Q}}\,, \tag{2.4}$$

where $A_\mu{}^{\bar{M}\bar{N}}$ are the gauge fields and $T^{\bar{P}\bar{Q}}$ are the generators of the SO(8,4) algebra satisfying

$$\big(T^{\bar{M}\bar{N}}\big)_{\bar{P}}{}^{\bar{Q}} = 2\,\delta_{\bar{P}}^{[\bar{M}}\,\eta^{\bar{N}]\bar{Q}}\,. \tag{2.5}$$

The covariant derivative acting on the scalar matrix becomes

$$D_\mu M_{\bar{M}\bar{N}} = \partial_\mu M_{\bar{M}\bar{N}} + 4\,A_\mu{}^{\bar{P}\bar{Q}}\,\Theta_{\bar{P}\bar{Q}|(\bar{M}}{}^{\bar{K}}\,M_{\bar{N})\bar{K}}\,, \tag{2.6}$$

ensuring gauge invariance of the scalar kinetic terms.

The Chern-Simons contribution is given by

$$\mathcal{L}_{\rm CS} = -\varepsilon^{\mu\nu\rho}\,\Theta_{\bar{M}\bar{N}|\bar{P}\bar{Q}}\,A_\mu{}^{\bar{M}\bar{N}}\left(\partial_\nu\,A_\rho{}^{\bar{P}\bar{Q}} + \frac{1}{3}\,\Theta_{\bar{R}\bar{S}|\bar{U}\bar{V}}\,f^{\bar{P}\bar{Q},\bar{R}\bar{S}}{}_{\bar{X}\bar{Y}}\,A_\nu{}^{\bar{U}\bar{V}}A_\rho{}^{\bar{X}\bar{Y}}\right), \tag{2.7}$$

where $f^{\bar{M}\bar{N},\bar{P}\bar{Q}}{}_{\bar{K}\bar{L}} = 4\,\delta_{[\bar{K}}{}^{[\bar{M}}\eta^{\bar{N}][\bar{P}}\delta_{\bar{L}]}{}^{\bar{Q}]}$ are the structure constants of the SO(8,4) Lie algebra, and $\varepsilon^{\mu\nu\rho}$ is the three-dimensional Levi-Civita symbol.

The scalar potential can be expressed in terms of the embedding tensor components as

$$\begin{aligned} V = \frac{1}{12}\,\theta_{\bar{K}\bar{L}\bar{M}\bar{N}}\theta_{\bar{P}\bar{Q}\bar{R}\bar{S}}\Big(&M^{\bar{K}\bar{P}}M^{\bar{L}\bar{Q}}M^{\bar{M}\bar{R}}M^{\bar{N}\bar{S}} - 6\,M^{\bar{K}\bar{P}}M^{\bar{L}\bar{Q}}\eta^{\bar{M}\bar{R}}\eta^{\bar{N}\bar{S}} \\ &+ 8\,M^{\bar{K}\bar{P}}\eta^{\bar{L}\bar{Q}}\eta^{\bar{M}\bar{R}}\eta^{\bar{N}\bar{S}} - 3\,\eta^{\bar{K}\bar{P}}\eta^{\bar{L}\bar{Q}}\eta^{\bar{M}\bar{R}}\eta^{\bar{N}\bar{S}}\Big) \\ + \frac{1}{8}\,\theta_{\bar{K}\bar{L}}\theta_{\bar{P}\bar{Q}}\Big(&2\,M^{\bar{K}\bar{P}}M^{\bar{L}\bar{Q}} - 2\,\eta^{\bar{K}\bar{P}}\eta^{\bar{L}\bar{Q}} - M^{\bar{K}\bar{L}}M^{\bar{P}\bar{Q}}\Big) + 4\,\theta\theta_{\bar{K}\bar{L}}M^{\bar{K}\bar{L}} - 32\,\theta^2\,. \end{aligned} \tag{2.8}$$

Decomposing SO(8,4) according to

$$\begin{aligned} \mathrm{SO}(8,4) &\longrightarrow \ \mathrm{GL}(3,\mathbb{R}) \times \mathrm{SO}(1,1) \times \mathrm{SO}(4)_{\rm global}\,, \\ X^{\bar{M}} &\longrightarrow \ \{X^{\bar{m}},\ X_{\bar{m}},\ X^{\bar{0}},\ X_{\bar{0}},\ X^{\bar{\alpha}}\}\,, \end{aligned} \tag{2.9}$$

where $\bar{m} \in [\![1,3]\!]$ and $\bar{\alpha} \in [\![9,12]\!]$ label the SL(3, $\mathbb{R}$) and $\mathrm{SO}(4)_{\rm global}$ vector representations, respectively. In this basis, the SO(8,4)-invariant tensor reads

$$\eta_{\bar{M}\bar{N}} = \begin{pmatrix} 0 & \delta_{\bar{m}}{}^{\bar{n}} & 0 & 0 & 0 \\ \delta^{\bar{m}}{}_{\bar{n}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\delta_{\bar{\alpha}\bar{\beta}} \end{pmatrix}. \tag{2.10}$$

Parametrizing the scalar matrix following [40], we have for the scalar matrix

$$M_{\bar{M}\bar{N}} = \begin{pmatrix} m + (\xi^2+\phi)m^{-1}(\xi^2-\phi) + 2\xi^2 & (\xi^2+\phi)m^{-1} & 0 & 0 & -\sqrt{2}\,[1+(\xi^2+\phi)m^{-1}]\xi \\ m^{-1}(\xi^2-\phi) & m^{-1} & 0 & 0 & -\sqrt{2}\,m^{-1}\xi \\ 0 & 0 & e^{2\tilde{\varphi}} & 0 & 0 \\ 0 & 0 & 0 & e^{-2\tilde{\varphi}} & 0 \\ -\sqrt{2}\,\xi^T[1+m^{-1}(\xi^2-\phi)] & -\sqrt{2}\,\xi^T m^{-1} & 0 & 0 & 1+2\,\xi^T m^{-1}\xi \end{pmatrix}, \tag{2.11}$$

which encodes 22 scalars of the theory ($22 = 32 - 10$, with 10 scalars gauge fixed using translations in the gauge group), which have been parametrized by $m = \nu\nu^T \in \mathrm{GL}(3,\mathbb{R})$ parametrizing the coset $\mathrm{GL}(3,\mathbb{R})/\mathrm{SO}(3)$, ϕ a 3×3 antisymmetric matrix, ξ a 3×4 matrix, and $\xi^2 = \xi\xi^T$, a dilaton $\tilde{\varphi}$. With this

parametrization, the potential takes the from ref. [40]:

$$
\begin{aligned}
V = 4\,e^{-4\tilde{\varphi}} + 2\,e^{-2\tilde{\varphi}}\Big[& -\operatorname{tr}\left(m+m^{-1}\right) + \operatorname{tr}\left(\phi m^{-1}\phi\right) - 2\,\operatorname{tr}\left(\phi m^{-1}\xi^2\right) - 2\,\operatorname{tr}\left(\xi^2\right) \\
& - \operatorname{tr}\left(\xi^2 m^{-1}\xi^2\right) + \frac{1}{2}\,\det\left(m^{-1}\right)\left(1 - \operatorname{tr}\left(\phi^2\right) - \operatorname{tr}\left(\xi^4\right) + \operatorname{tr}\left(\xi^2\right)^2\right) \\
& + \frac{1}{2}\,\mathrm{T}\left(m^{-1}(\xi^2-\phi), (\xi^2+\phi)m^{-1}, m + (\xi^2+\phi)m^{-1}(\xi^2-\phi) + 2\,\xi^2\right) \\
& + \frac{1}{4}\,\mathrm{T}\left(m^{-1}, m + (\xi^2+\phi)m^{-1}(\xi^2-\phi) + 2\,\xi^2, m + (\xi^2+\phi)m^{-1}(\xi^2-\phi) + 2\,\xi^2\right)\Big],
\end{aligned}
\tag{2.12}
$$

where $\mathrm{T}\,(A,B,C) = \varepsilon_{mnp}\,\varepsilon_{qrs}\,A^{mq}B^{nr}C^{ps}$. We can further restrict ourselves to a set of 13 scalars which is still a consistent truncation, with

$$
\begin{aligned}
\xi &= \begin{pmatrix} 0 & 0 & 0 & x_1 \\ 0 & 0 & 0 & x_2 \\ 0 & 0 & 0 & x_3 \end{pmatrix}, \\
\phi &= \begin{pmatrix} 0 & x_4 & x_5 \\ -x_4 & 0 & x_6 \\ -x_5 & -x_6 & 0 \end{pmatrix}, \\
\tilde{\varphi} &= \tilde{x}_{13}, \\
\nu &= e^{(6\,\tilde{x}_7 + 3\,\tilde{x}_8 + \sqrt{3}\,\tilde{x}_9)/6} \begin{pmatrix} 1 & \frac{x_{10}}{\sqrt{2}} & \frac{x_{11}}{\sqrt{2}} + \frac{x_{10}x_{12}}{4} \\ 0 & e^{-\tilde{x}_8} & \frac{e^{-\tilde{x}_8}\,x_{12}}{\sqrt{2}} \\ 0 & 0 & e^{-(\tilde{x}_8 + \sqrt{3}\,\tilde{x}_9)/2} \end{pmatrix}.
\end{aligned}
\tag{2.13}
$$

We note here that all dilaton fields are denoted with a tilde. This convention is adopted in anticipation of a later redefinition of the form $x_i = e^{\tilde{x}_i}$ for these fields. By reserving the notation without tildes for the exponentiated variables, we maintain a clear and consistent distinction for use in subsequent expressions.

A first numerical exploration suggests that attention can be focused on the five scalars $x_1, x_2, x_4, \tilde{x}_8$, and x_{10}. This subset, selected at random from the original set of thirteen scalar fields, serves to render the problem more tractable at the outset. It is important to emphasize, however, that this subset does not constitute a consistent truncation of the full theory. Consequently, the truncated theory involving only five scalar fields may not constitute a formally consistent truncation of the full 13-scalar theory, as higher-order interactions could, in principle, source the remaining scalar modes. As a result, the interpretation of the five-scalar theory as a consistent subsector of the original theory may no longer hold. Nonetheless, the five-scalar model remains a self-consistent theory in its own right and can be used independently. This distinction does not affect our numerical approach.

Nevertheless, the solutions we obtain remain valid solutions of the complete theory. This is because we identify the critical points of the scalar potential by first computing $\nabla V = 0$ with all scalar fields included, and only then setting the remaining fields to zero. By performing the differentiation before the truncation, we ensure that the resulting configurations satisfy the full equations of motion and are therefore legitimate solutions of the complete theory.

To compute the gradient, we exploit the fact that we possess an analytic expression for the scalar potential (2.12), which we implement in Mathematica for numerical analysis. Note however that it could have been fully done using a numerical approach, automatic differenciations methods (that we will use

later to do the gradient descent for the loss function).

Let us make a comment at this point. Since we have an explicit parametrisation of the potential with (2.12), one might argue that the problem of identifying flat directions could be addressed by directly computing the gradient of the expression, setting the appropriate scalar fields to zero and subsequently simplifying the resulting equations. However, attempting to carry out this procedure using a symbolic solver such as Mathematica quickly reveals its limitations: the resulting expressions are too complex to be simplified into a manageable form, and do not yield usable relationships that express some variables in terms of others. This complexity does not, however, rule out the possibility that simpler solutions satisfying the condition of vanishing gradient may exist. In this work, we aim to identify such solutions, which we seek using numerical methods.

An additional advantage of this approach is its broader applicability. In scenarios where an explicit analytic expression for the potential and its gradient is inaccessible—yet numerical evaluation remains feasible—our method retains its validity. In particular, one could replace the analytic gradient by automatic differenciation methods if one can only evaluate the potential. Hence, the numerical strategy adopted here may prove effective even in cases where an exact analytic expression for the potential is out of reach. **(bd: other motivation? Example of this ?)** ⟸

3 Numerical analysis

3.1 Gradient descent and local analysis

To identify the flat directions in the potential, we will use numerical tools. The procedure is as follows: first, we sample the underlying manifold, and then we use numerical techniques to extract analytical information from the resulting cloud of points. The first step is thus to sample the manifold.

3.1.1 Sampling the manifold

To do so, we perform a basic gradient descent. We begin by randomly and uniformly initializing points within a hypercube of range $[-2, 2]$. This choice is important to ensure that points are not restricted to the inner range $[-1, 1]$. This will be important later, when performing the symbolic regression [1]. For this example, we choose to generate 10^5 points. This number is motivated by the dimensionality of our problem. If we aim to populate all possible directions and want approximately $\mathcal{O}(10)$ points per direction, then we require $\mathcal{O}(10^5)$ points. Given that the true intrinsic dimensionality of the manifold is less than or equal to 5, this value serves as a conservative upper bound for the number of points needed to adequately sample the manifold.

We then perform gradient descent on the points using TensorFlow's automatic differentiation [41]. The loss function is defined as:

$$\mathcal{L} = \sum_{i=1}^{n_{points}} || \nabla V(\vec{X}^{\,i}) \Big|_{\vec{y}=0} ||^2 \tag{3.1}$$

[1] The rationale is that symbolic regression (SR) evaluates the data points $\vec{x}^i$ using a candidate polynomial function p. The loss function—used later to assess the quality of a given polynomial—is defined as $\sum_i p(\vec{x}^i)$. When all data points lie within the interval $[-1, 1]$, the SR algorithm tends to favor high-degree polynomials, as these often yield smaller values over this range, thereby artificially lowering the loss. Introducing data points with absolute values greater than 1 mitigates this bias by discouraging the selection of overly complex polynomials that exploit this effect.

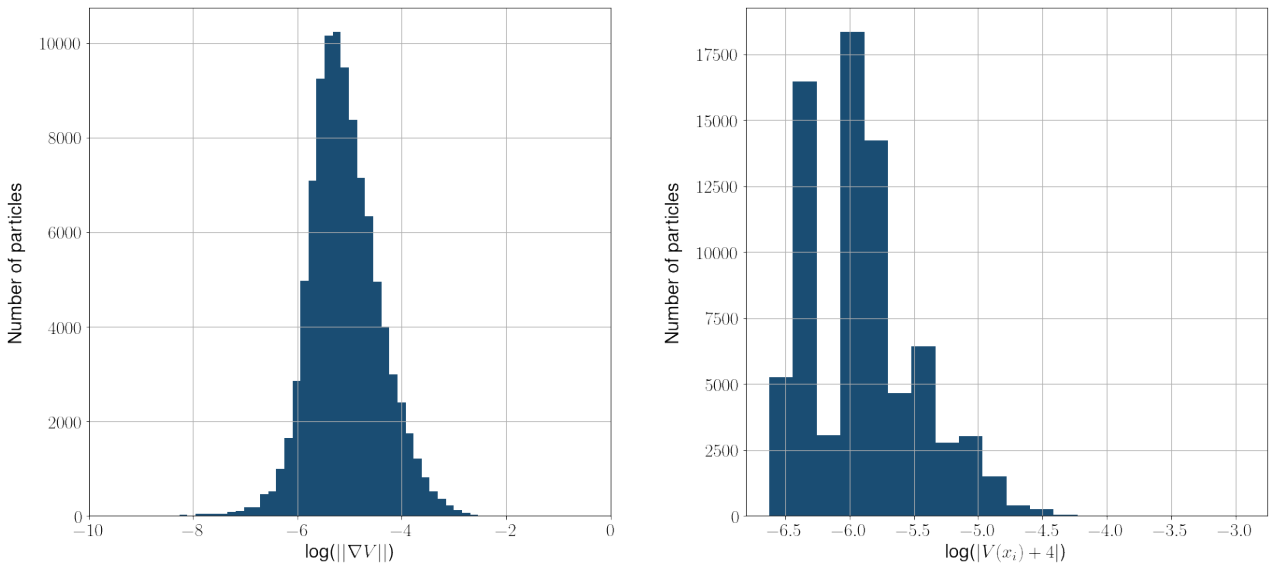


Fig. 1 Left panel: Histogram of the logarithm of the norm of the gradient for each point. Right panel: Histogram of the distance of the value of the potential for each point.

where $\vec{X}^{\,i} = (x_1^i, \ldots, x_6^i, \tilde{x}_7^i, \tilde{x}_8^i, \tilde{x}_9^i, x_{10}^i, \ldots, x_{12}^i, \tilde{x}_{13}^i)$ and $\vec{y}^{\,i} = (x_3^i, x_5^i, x_6^i, \tilde{x}_7^i, \tilde{x}_9^i, x_{11}^i, x_{12}^i, \tilde{x}_{11}^i)$ denotes the data points. This way we only keep $(x_1^i, x_2^i, x_4^i, \tilde{x}_8^i, x_{10}^i)$ alive after we have taken the derivative. As already mentionned above, we use the analytic formula for ∇V, but we could have perform the gradient descent by fully using automatic differentiation.

Through trial and error, we observed that periodically reinitializing the optimizer significantly improved the convergence rate. This technique bears similarity to the concept of warm restarts introduced in [42]. However, we found that simply scheduling the learning rate without resetting the optimizer yielded slower convergence. We interpret this as follows: when the optimizer is reinitialized, it effectively "forgets" its past gradient history. As a result, the actual learning rate used corresponds more closely to the specified value, rather than being internally adjusted based on accumulated past gradients. This effect seems to contribute to faster convergence in our case. The evolution of the loss function, along with the learning rate schedule, is shown in Fig. (2). As can be seen in this figure, the convergence rate improves significantly each time the optimizer is reinitialized. We also observe that, following the last few reinitializations, the loss exhibits a small bump immediately after the restart. We interpret this behavior as a consequence of the increased learning rate: some points that previously had low loss values may momentarily worsen before benefiting from faster convergence. This effect is likely due to points initially located in regions with weak attractive basins being pushed toward areas where the potential gradient is steeper, thus accelerating their convergence. This strategy introduces the risk of desampling certain regions in favor of others, introducing the risk to hide some flat directions in the potential. However, we believe that with enough points, it is very unlikely for a flat direction to be completely hidden from us and desampled.

The Adam optimizer was employed throughout the gradient descent procedure. It was reinitialized at iterations 250, 500, 750, with the learning rate fixed at 10^{-2}. At iteration 1000, the optimizer was reinitialized once more, this time with a reduced learning rate of 10^{-3}. A final reinitialization was performed at iteration 1500, setting the learning rate to 10^{-4}, and the optimization was continued for an additional 500 epochs.

Upon completion, it was observed that all data points converged to values close to -4. Specifically,

nearly all data points attained values within an absolute error of at least 10^{-4}, with the exception of four points whose deviations were on the order of 10^{-3}. Similarly, 99,739 out of 100,000 data points exhibited gradient norms smaller than 10^{-3}. These results are summarized in Figure 1.

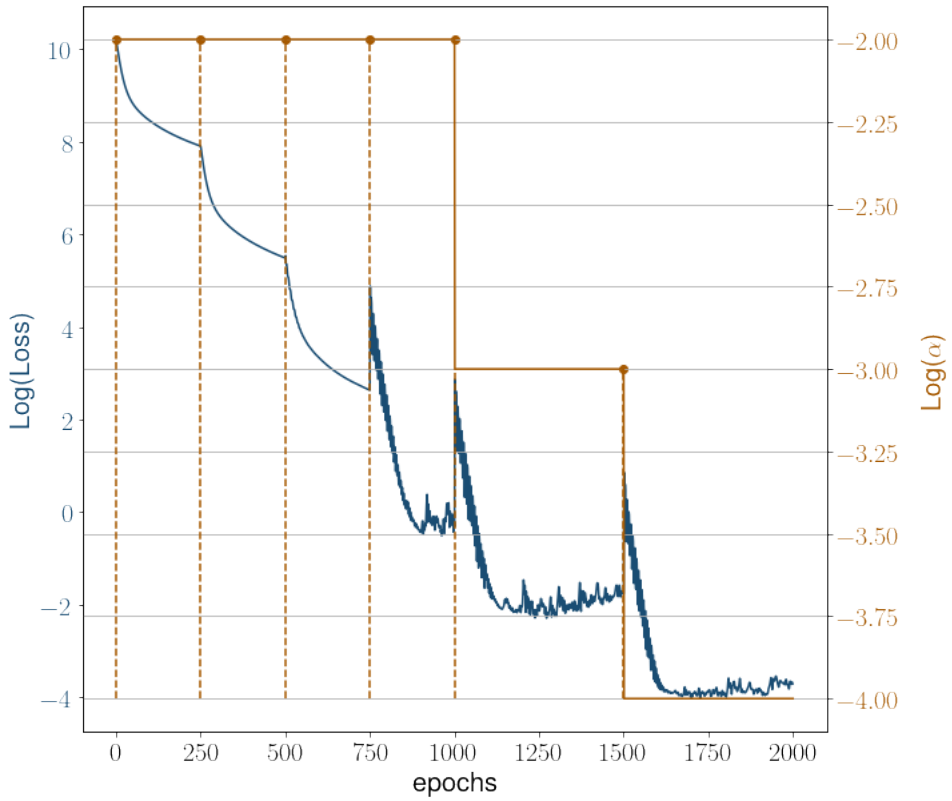


Fig. 2 Evolution of the loss function during gradient descent and corresponding learning rate schedule, noted α on the left y-axis, both plotted on a logarithmic scale. Dashed lines indicate epochs at which the optimizer was reinitialized.

As a first visualization, we present a triangular plot of the data in Fig. 3. This figure shows all possible 2D projections of the data, along with the 1D histograms of each coordinate after gradient descent. We can make a few comments on those plots. First, we observe non-trivial correlations in the data. Some of these may result from larger basins of attraction, such as in the x_1/x_2 plot. However, the structures seen in the $x_4/\tilde{x}_8$ or x_4/x_{10} plots likely reflect genuine features of the manifold, and will show latter that this is indeed the case. We also note that all directions appear to be well populated.

3.1.2 Determining the dimension

Once the gradient descent has been completed and the flat directions sampled, the next step is to identify the structure of the underlying manifold. Our goal is to eventually obtain an analytical expression, not just a numerical description. Before applying symbolic regression to search for such an expression, we can first perform some exploratory analyses to better understand the data. Specifically, we aim to determine the dimensionality of the manifold and wether it consists of a single connected component or multiple disjoint components (e.g., two intersecting hyperplanes). To this end, we apply a local Principal Component Analysis (PCA). For each point, we identify its k nearest neighbours and perform a PCA on that local neighbourhood. This procedure allows us to determine how many principal directions are needed to explain

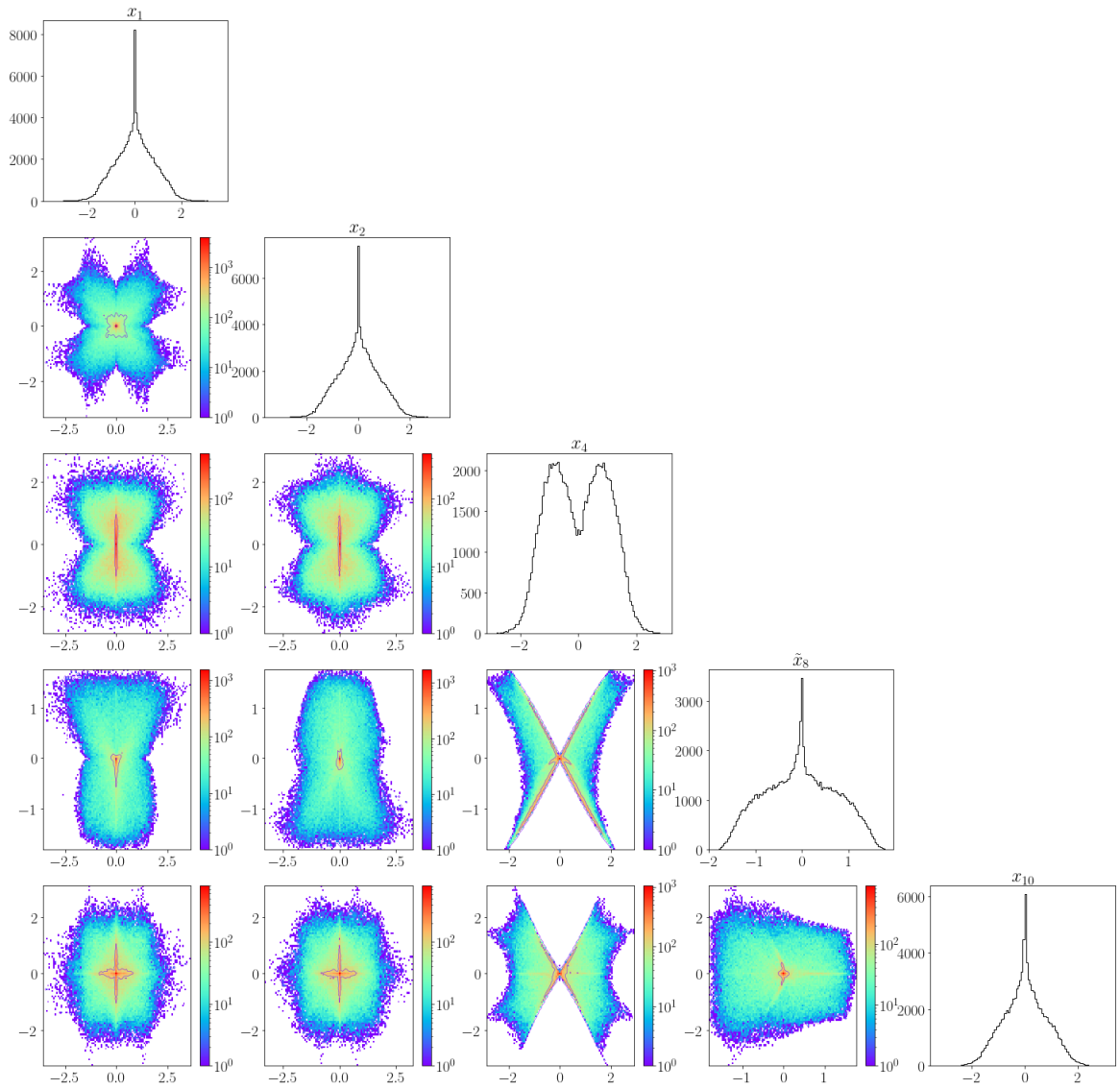


Fig. 3 Triangular plot showing $2d$ projections and $1d$ histograms of the data after the gradient descent

a given proportion ϵ of the data variance. In other words, it provides an estimate of the local dimensionality around each point. We perform this analysis for several values of k, namely $k \in \{5, 10, 20, 50, 100\}$, and we fix $\epsilon = 0.99$. The results are presented in Fig. 4.

We observe that for every choice of k, there is a prominent peak at $d = 3$, suggesting that the underlying manifold is three-dimensional. For $k = 5$, a noticeable fraction of points are assigned a dimensionality of 2. This can be attributed to the fact that if the true dimension is 3, then selecting only 5 neighbours may not sufficiently populate all three directions, leading the algorithm to underestimate the dimensionality for a fraction of the points. Additionally, for $k \geq 20$, we observe an increasing number of points being assigned dimensionalities of 4 or 5. This behavior can be explained by the loss of locality when the number of neighbours becomes too large: increasing k results in a coarser approximation, and the algorithm may then incorporate points that are no longer truly local. This artificial enlargement of the neighbourhood can cause the estimated local dimensionality to rise. We thus assume that the manifold under investigation has an intrinsic dimension of 3.

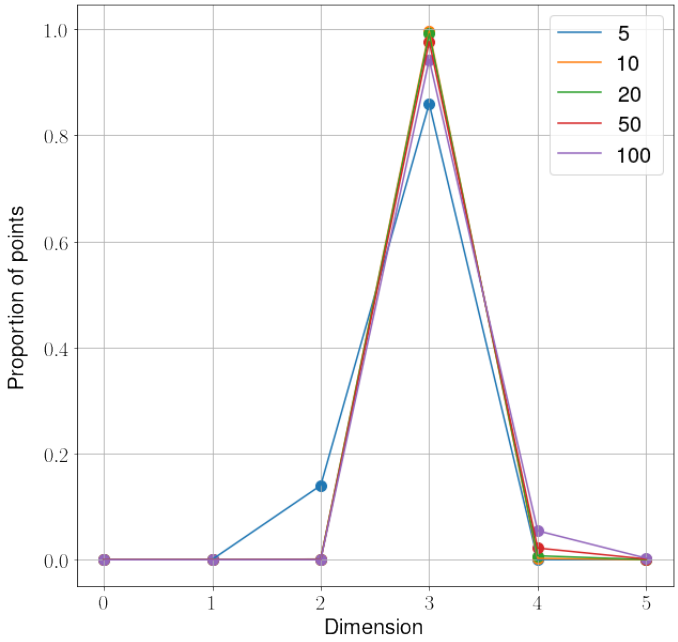


Fig. 4 Results of the local PCA analysis. The x-axis shows the dimension inferred by the algorithm, and the y-axis indicates the proportion of points for which that dimension was found. Each curve corresponds to a different value of k in the k-nearest neighbours.

One possible scenario is that our data actually consists of several distinct three-dimensional manifolds, and the points previously identified with dimension 4 may lie at the intersections of these manifolds. To illustrate, consider the intersection of two lines: at the intersection point, the local dimensionality estimated by the previous PCA algorithm would be 2. To rule out this possibility, we apply a clustering algorithm based on local density, where we only select the points of dimension 3. This, way, we remove the possible intersection points with dimension 4 or 5. For the purpose of the clustering, we use the HDBSCAN algorithm[2]. We set the minimum cluster size to 10 points. Out of the 98,606 3-dimensional points, the algorithm identifies three clusters: one of size 10, one of size 91,747, and one of size 6,849. The smallest cluster (10 points) is likely an artefact of local fluctuations in the data density and is not interpreted as physically meaningful. The last group, comprising 6,849 points, consists of points that the algorithm could not assign to any cluster. We interpret these as outliers or edge points rather than representatives of a separate manifold. What the algorithm does indicate, however, is that over 90% of the data belongs to a single dominant cluster. In Fig. 5, we show 3D projections of the data to visualize the clustering. In these scatter plots, turquoise points belong to the main cluster, while purple points are those that the algorithm failed to assign. In one of the plots, a few yellow points can be seen; these correspond to the smallest cluster of 10 points. From visual inspection, it appears that the unassigned (purple) points lie mostly on the boundary of the sampled region. We therefore interpret their unassigned status not as evidence of belonging to another manifold, but rather as a result of insufficient local density near the edges of the dataset.

From this analysis, we conclude that the gradient descent procedure has produced a sampling of a

[2]Details of the algorithm can be found in the original HDBSCAN paper or documentation [43]. Briefly, it is a density-based clustering method that extends DBSCAN by converting it into a hierarchical clustering algorithm and then extracting a flat clustering based on the stability of clusters.

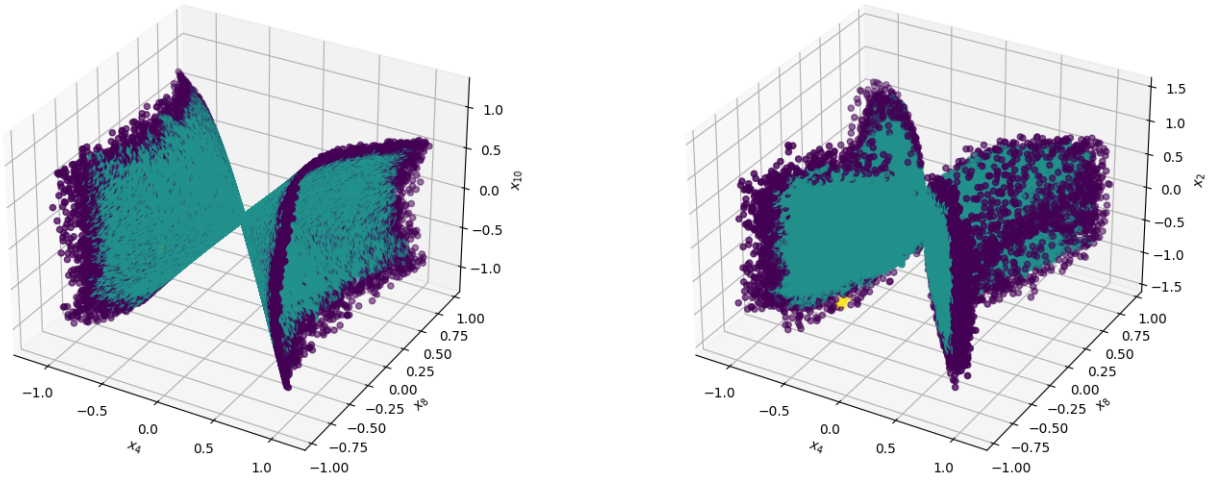


Fig. 5 3D plots of the data in selected coordinates. Left: (x_4, x_8, x_{10}). Right: (x_2, x_4, x_8).

single, connected, three-dimensional manifold.

After performing both PCA and clustering, we thus find that the data obtained after gradient descent samples a three-dimensional manifold.

3.2 Annealed Importance Sampling for polynomial symbolic regression

Now we have sampled the manifold and extracted some basic information about it (namely its dimension and the fact that it is made of one block), we would like to see if we can extract some analytic formula to characterise it. What we have at the moment, are points on a 3-dimensional manifold, which are embedded in a 5-dimensional space. Therefore we can conclude that, in order to characterise the manifold, we need two independent constraints on the embedding coordinates $\vec{x} = (x_1, x_2, x_4, \tilde{x}_8, x_{10})$. If we have a look at the form of the potential, we can observe that if we use $x_8 = e^{\tilde{x}_8}$, that, up to some potential global factor of the dilaton of the form $e^{\tilde{x}_8}$, this potential is actually a polynomial of the embedding coordinates. Therefore, the components of $|\nabla V|$ are also polynomials in those variables. We conclude that the constraints on the embedding coordinates we are looking for are polynomial constraints of the form $p(\vec{x}) = 0$, and that there should be at least two of those. Note that we are now taking $\vec{x} = (x_1, x_2, x_4, x_8, x_{10})$. Of course if one takes directly the gradient of (2.12), one ends up with such conditions, but none are usable directly to solve for two of the variables in terms of the others. The problem we are facing here is therefore a problem of symbolic regression: we are looking for analytic expressions (polynomial) that vanish once evaluated on our data points.

In the realm of symbolic regression, the aim is to uncover interpretable mathematical expressions that best describe a given dataset. In the problem discussed in this paper, we are looking for polynomial expressions such that $p(\vec{x}^i) = 0$, with $i \in \{1, \ldots, \texttt{n\_sample}\}$. This task involves navigating a vast, discrete, and often rugged search space of possible symbolic models, which poses significant challenges for traditional sampling methods. Markov Chain Monte Carlo (MCMC) techniques, while widely used, can struggle with

poor mixing and getting trapped in local optima, especially in high-dimensional or multimodal spaces.

3.2.1 Annealed Sequential Monte Carlo principle

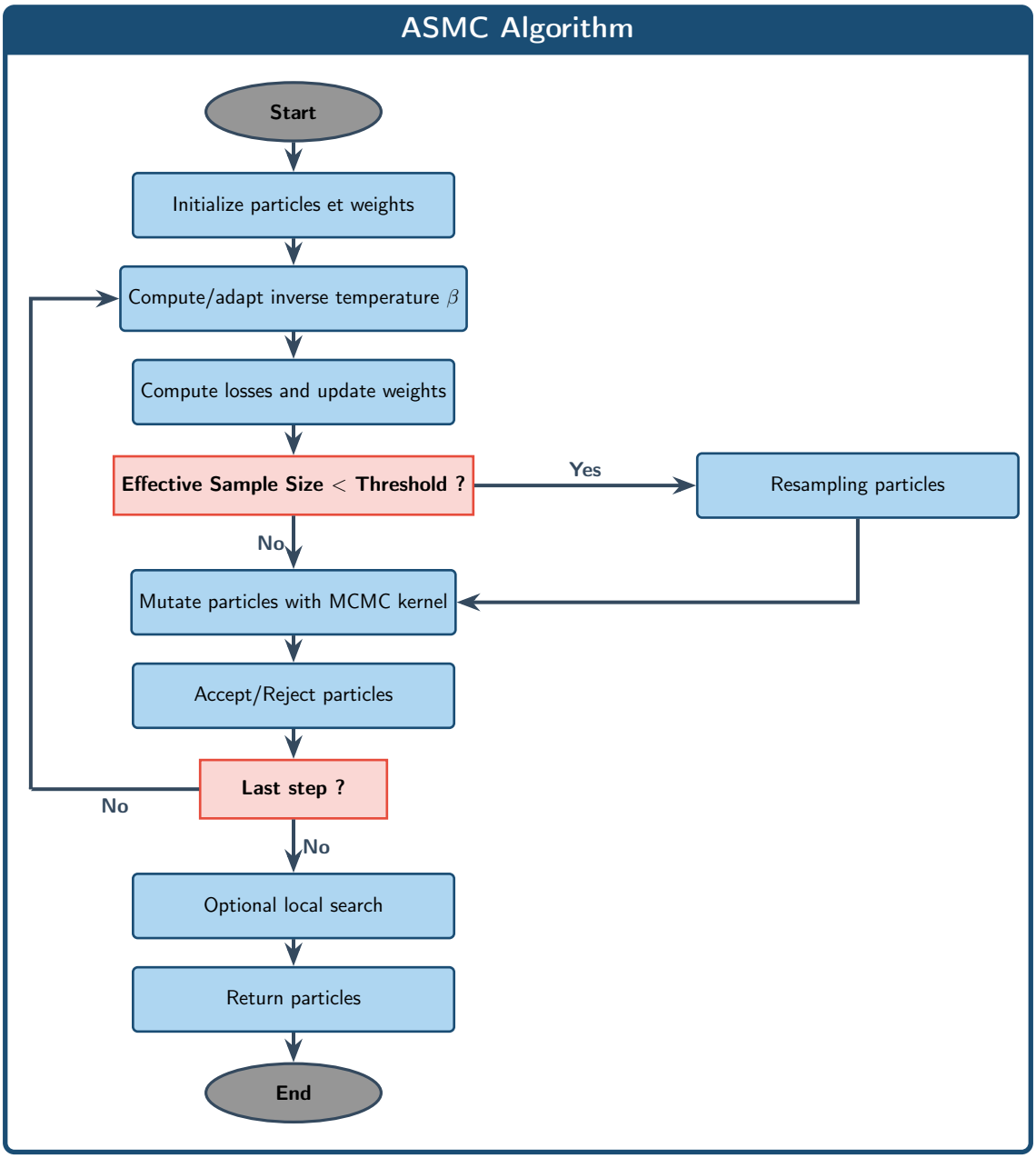


Fig. 6 The flow chart of the Annealed Importance Sampling - Sequential Monte Carlo algorithm

To address these challenges, we employ an Annealed Importance Sampling (AIS) combined with Sequential Monte Carlo (SMC) methods. AIS constructs a sequence of intermediate distributions that smoothly transition from an initial, tractable distribution (e.g., a prior over symbolic expressions) to the complex target posterior distribution. This annealing process is guided by a temperature-like parameter that gradually emphasizes the data likelihood, allowing for more efficient exploration of the probability landscape.

SMC enhances this procedure by propagating a population of particles—each representing a candidate symbolic expression—through the sequence of distributions. At each step, particles are reweighted based on the incremental change in the distribution, resampled to focus computational effort on high-probability regions, and mutated via operations. This combination of importance sampling, resampling, and mutation maintains diversity among the particles and prevents premature convergence to suboptimal models.

These features make AIS-SMC particularly well-suited for symbolic regression tasks, where the search space is not only high-dimensional but also structured and discontinuous.

Let us now explain in more details how does this procedure goes. The goal is to reconstruct some distribution function $d(z)$, where here z in going to be some polynomial. We will try to reconstruct this density function by series of density function $\pi_n(z_n) = \gamma_n(z_n)/Z_n$ with $n = 1, \ldots, N$ is going to be the number of annealing steps, π_n is defined in terms of an unnormalized density γ_n and we have the normalizing constant $Z_n = \int \gamma_n(z)\mathrm{d}z$. We also assume we have a sequence of inverse temperature constants β_n, where $0 = \beta_1 < \beta_2 < \cdots < \beta_N = 1$. We then define the unnormalized density at level n in terms of a prior distribution $p_0(z)$ over the hypothesis space and a loss function $L(z)$:

$$\gamma_n(z) := d_0(z)\ \exp\left(-\beta_n L(z)\right). \tag{3.2}$$

At each epoch, we have a set of particles and weights $\{z_{n-1}^k, w_{n-1}^k\}$ approximating π_{n-1} (meaning $\mathbb{E}_{\pi_{n-1}}[f] \approx \frac{\sum_k w_{n-1}^k f(z_{n-1}^k)}{\sum_k w_{n-1}^k}$), and we want to obtain a new set $\{z_n^k, w_n^k\}$ approximating π_n. To do so, for each particle z_{n-1}^k, we propose a new particle $z_n^k \sim q(z_n|z_{n-1}^k)$ and calculate the new unnormalized importance weight w_n^k. The latter are updated using the formula

$$w_n^k = w_{n-1}^k \times \alpha_n^k \tag{3.3}$$

where α_n^k is the incremental importance weight. The standard form for α_n^k, which requires introducing an auxiliary backward transition kernel $q(z_{n-1}|z_n)$, is:

$$\alpha_n^k = \frac{\gamma_n(z_n^k)q(z_{n-1}^k|z_n^k)}{\gamma_{n-1}(z_{n-1}^k)q(z_n^k|z_{n-1}^k)} \tag{3.4}$$

The weights are finally normalized and we get $w_n^k \to \frac{w_n^k}{\sum_j w_n^j}$.

Let's us now focus on the forward $q(z_n^k|z_{n-1}^k)$ and backward propagation kernel $q(z_{n-1}^k|z_n^k)$ for now. The forward propagation kernel defines how we generate the state at time n given the state at time $n-1$. It gives the probability to transition from z_{n-1} to z_n. The backward kernel represents a hypothetical probability of transitioning back from state z_n to state z_{n-1}. The purpose of q is to give a proposition for z_n given z_{n-1}. For the implementation of the two, we decide to implement an AIS-style MCMC code : we make some move in the space of polynomials, and then accept or reject those new polynomials based on an acceptance rate. So we first need to choose what are the available moves in the space of polynomials. Here are the choices we have made for our symbolic regression task :

- Coefficient pertubation. Given a polynomial z_{n-1}, we choose randomly one of its coefficient and modify it by a random noise from the guassian distribution $\mathcal{N}(0, \sigma^2)$. So for eg : $2x_1 + 3x_2^2 \to 2.1x_1 + 3x_2^2$.
- Variable multiplication. Given a polynomial z_{n-1}, we choose randomly one of his monomial, and multiply it by one of the available variable. So for eg : $2x_1 + 3x_2^2 \to 2x_1x_2 + 3x_2^2$.

- Variable division. Given a polynomial z_{n-1}, we choose randomly one of his monomial, and divide it by one of its variable. So for eg : $2x_1x_2 + 3x_2^2 \to 2x_1 + 3x_2^2$.

The polynomials get mutated at each epoch. The mutation is chosen randomly, assigning probabilities `p_modify` , `p_multiply` and `p_divide` to each possibility. This implementation enables a direct computation of both the forward and backward propagation kernels. In particular, for the case of coefficient perturbations, the transition is symmetric. That is,

$$q(z_n^k \mid z_{n-1}^k) = q(z_{n-1}^k \mid z_n^k), \tag{3.5}$$

which significantly simplifies the computation of the incremental importance weight.

However, this symmetry does not hold in general. For other types of operations—such as multiplication or division by a variable—the transition from z_{n-1}^k to z_n^k is generally not equivalent to the reverse transition from z_n^k to z_{n-1}^k. In such cases $q(z_n^k \mid z_{n-1}^k) \neq q(z_{n-1}^k \mid z_n^k)$ and particular care must be taken in the computation of the backward propagation kernel.

The acceptance ratio for the MCMC algorithm is given by :

$$A(z_n, z_{n-1}) = \min\left(1, \frac{\gamma_n(z_n^k) q(z_{n-1}^k | z_n^k)}{\gamma_{n-1}(z_{n-1}^k) q(z_n^k | z_{n-1}^k)}\right) \tag{3.6}$$

We then draw $u \sim \text{Uniform}(0,1)$, accept the new particle if $u < A(z_n, z_{n-1})$ and reject it otherwise.

We have adapted the classic approach outilned above. As can be seen in (3.2), what matters the most to determine γ_n is not the inverse temperature, but rather the product of the inverse temperature and the loss function. Therefore, instead of implementing an arbitrary schedule for β, we employed an adaptive temperature approach. The number of particles accepted at each run is monitored. We aim at following the following schedule as function of the `epoch`:

$$\texttt{acceptance\_ratio} = 0.8 - 0.5 \times \left(\frac{\texttt{epoch}}{\texttt{n\_epochs}}\right), \tag{3.7}$$

where `n_epochs` is the total number of epochs. If we accept more particle (in the 5 last epochs), we decrease the temperature by 5%, if we accept less we increase it by 5%. This encourages exploration during the early stages of training and gradually transitions to a more selective regime. **(ce: So we don't use eq.** (3.6) **at all? This is not clear to me.)** ⟸

Before closing the section, we need to discuss what is the Loss function we choose. It is required to compute the unormalized distribution γ_n. Assume that we have $z = \sum_k c_k X_k$, where c_k are the coefficients of the polynomials, and X_k is a short notations for all the possible monomials up to a given degree (so if we have x and y as variables, and the maximum degree is 2, then X_k are $1, x, y, x^2, y^2, xy$). For the Loss function, we decided to take :

$$L(z) = \sum_i z(x_i)^2 + \frac{\lambda}{\sum_k |c_k|} \tag{3.8}$$

So the first term is just the sum of the squared of the polynomial evaluated on the data. We want to make this 0 so we find polynomial that annihilates our data. The second part is a regularisation factor : it prevents the algorithm to send all the coefficient to 0, which would give a trivial solution to the problem. We typically take $\lambda \sim \mathcal{O}(1000)$.

Once the Annealing loop is over, we end up with a total of n_{sample} particles, which in principle should

be close to annihilate our data, but whose coefficients may need some more fine tuning. To deal with it, we ran a quick exploitation phase, where, for each polynomial, we now only modify its coefficients, with a pertubation $\epsilon \sim \mathcal{N}(0, \sigma_1)$, and keep the new particle, only if the Loss function is now getting smaller. This allow to fine-tune the coefficients. **(ce: Mention threshold? And number of iterations?)** ⟸

3.2.2 Results

The results of this method are as follow. The code finds the 8 different following polynomials:

$$p_1 = -\sqrt{2}\, x_1 + \sqrt{2}\, x_1 x_8 + x_2 x_8 x_{10} - \sqrt{2}\, x_2 x_4 x_8, \tag{3.9a}$$
$$p_2 = 2\, x_2 - 2\, x_2 x_8 + \sqrt{2}\, x_1 x_{10} + 2\, x_1 x_4 - x_2 x_8 x_{10}^2, \tag{3.9b}$$
$$p_3 = 2\, x_2 - 2\, x_2 x_8 + \sqrt{2}\, x_1 x_8 x_{10} + 2\, x_1 x_4 x_8 - 2\, x_2 x_4^2 x_8, \tag{3.9c}$$
$$p_4 = \sqrt{2}\, x_2 - \sqrt{2}\, x_2 x_8 + \sqrt{2}\, x_1\, x_4 + x_1 x_8 x_{10} - x_2 x_4 x_8 x_{10}, \tag{3.9d}$$
$$p_5 = -2\, x_1 - 2\, x_2 x_4 - 2\, x_1 x_4^2 + 2\, x_1 x_8 + \sqrt{2}\, x_2 x_{10} + x_1 x_8 x_{10}^2, \tag{3.9e}$$
$$p_6 = -2\, x_2 x_4 - 2\, x_1 x_4^2 + 2\, x_2 x_4 x_8 + \sqrt{2}\, x_2 x_{10} - \sqrt{2}\, x_2 x_8 x_{10} + x_1 x_8 x_{10}^2, \tag{3.9f}$$
$$p_7 = -\sqrt{2}\, x_1^2 x_4 + \sqrt{2}\, x_2^2 x_4 + \sqrt{2}\, x_1 x_2 x_4^2 - x_1^2 x_{10} - x_2^2 x_{10}, \tag{3.9g}$$
$$p_8 = -2 + 4x_8 - 2x_8^2 + 2x_4^2 x_8 - x_8^2 x_{10}^2, \tag{3.9h}$$

where $x_8 = e^{\tilde{x}_8}$. They are found with different frequencies with some polynomials occuring more often than others. The results are presented in (1).

Let us clarify a point regarding the expected structure of the solutions to the system defined by the critical points of the scalar potential, i.e., the variety defined by the vanishing of the gradient ∇V.

In algebraic geometry, the set of solutions to the equations $\nabla V = 0$ defines an algebraic variety over a certain base field. In our case, the potential V and its gradient ∇V are polynomial functions with coefficients lying in a specific field extension of $\mathbb{Q}$—typically of the form $\mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \dots)$, where $d_i \in \mathbb{Q}$. That is, the coefficients of the polynomials are combinations of rational numbers and square roots of rational numbers.

The conformal manifold (or the moduli space of solutions) is then a subvariety of this ambient space, defined as the common zero locus of the polynomials in ∇V. From the perspective of algebraic geometry, this variety is cut out by an ideal in the polynomial ring with coefficients in the aforementioned field extension of $\mathbb{Q}$. Since the defining equations have coefficients in this field, the variety itself is defined over that same field.

Now, consider the numerical solutions obtained via methods such as annealed importance sampling. If these numerical procedures converge to points on the same variety, and assuming that the variety is irreducible over the given field, then any point (i.e., any solution) on the variety must lie in the same field extension over which the variety is defined. This follows from basic principles in algebraic geometry: the set of algebraic points on a variety defined over a field K are themselves algebraic over K, and the field of definition of these points must be a (possibly finite) extension of K.

Therefore, even if the numerical solver does not explicitly reveal this, one should expect that the coordinates of the solutions (i.e., the values of the scalar fields at critical points) lie in the same field extension of $\mathbb{Q}$ as the coefficients of the original system of polynomials ∇V. That is, they should belong to $\mathbb{Q}(\sqrt{d_1}, \sqrt{d_2}, \dots)$ or a finite extension thereof.

This argument justifies the expectation that if the scalar potential V and its gradient ∇V involve only rational numbers and square roots, then the solutions to $\nabla V = 0$—regardless of how they are found—should

also lie within the same field extension.

This is this argument that allowed us to round the coefficients of the polynomials given by the annealed importance sampling algorithm, and found the polynomials in Eq. (4.5).

3.2.3 Setup of ASMC

We now provide further details about the implementation of our code. To recover the polynomials presented in Eq. (4.5), we performed 1000 independent runs on a cluster, each involving 1000 particles. As can be seen in (3.2), we see that what matters the most in the the inverse temperature, but rather the inverse temperature multiplied by the loss function. Therefore, instead of implementing an arbitrary schedule for β, we employed ans adaptive temperature approach, as previously described. The acceptance ratio was controlled by the following schedule:

$$\texttt{acceptance\_ratio} = 0.8 - 0.5 \times \left(\frac{\texttt{epoch}}{\texttt{n\_epochs}} \right), \tag{3.10}$$

which encourages exploration during the early stages of training and gradually transitions to a more selective regime. We initialized the algorithm with

$$\begin{aligned} &\texttt{n\_epochs = 1000}\,, \\ &\texttt{beta = 1e-6}\,, \\ &\texttt{p\_modify = 0.5}, \\ &\texttt{p\_multiply = 0.25}, \\ &\texttt{p\_divide = 0.25}. \end{aligned} \tag{3.11}$$

This strategy was determined empirically through trial and error; a systematic analysis of its optimization is left for future work. For the representation of polynomials, we adopt a vectorial approach. Specifically, each polynomial is represented as an array of its coefficients $\{c_k\}$, where $k \in \{1, \ldots, n_{\text{mon}}\}$ and n_{mon} denotes the number of authorized monomials, determined by the chosen maximum degree. We also impose a constraint on the maximum number of terms within a single polynomial to promote sparsity. An alternative approach would be to encourage sparsity through the prior distribution or within the loss function formulation. For the simulations we have done, we chose :

$$\begin{aligned} &\texttt{max\_degree} = 4, \\ &\texttt{max\_num\_monomials} = 6. \end{aligned} \tag{3.12}$$

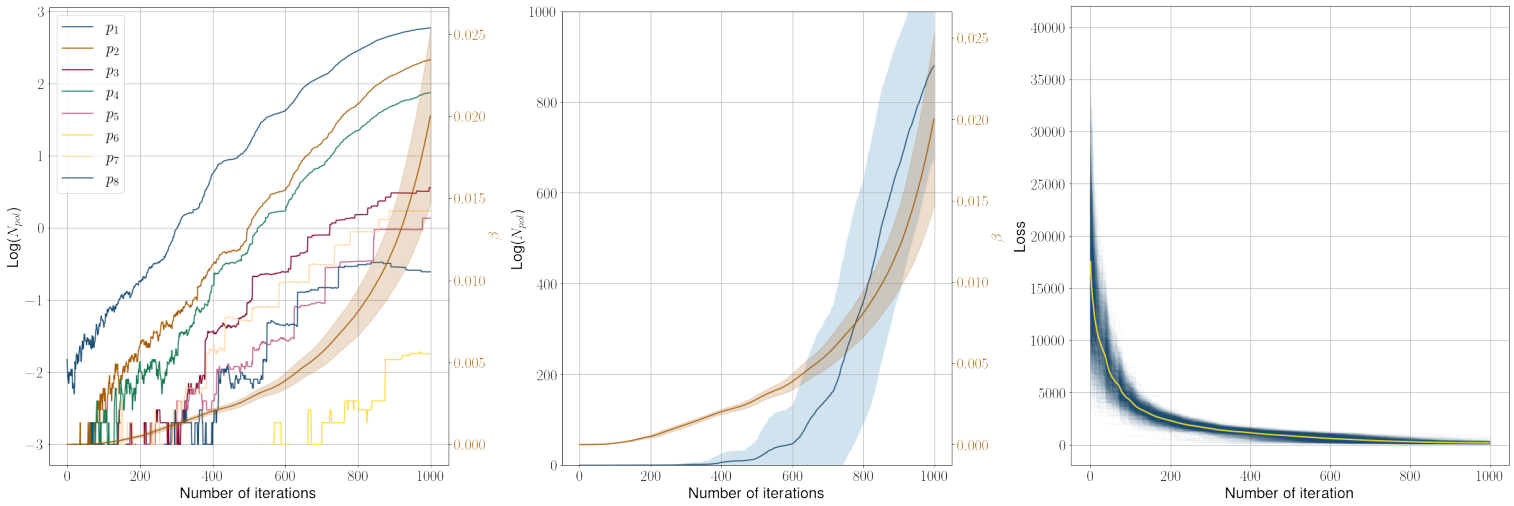
The time it takes to do a single run in approximately 600s on a regular computer. The main results are summarized in Fig. 10 (a) and (b) and the losses of the run are shown in Fig. 10 (c). The frequencies with which each of the polynomials in Eq. (4.5) was recovered are reported in the table below:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	$\varnothing$
Frequency	92.6%	75.0%	1.7%	51.9%	0.7%	0.1%	0.2%	0.7%	0.3%
Maximum # of Representant	1000	1000	869	991	710	18	879	207	$\varnothing$

(3.13)

With this approach, the average number of particles reproducing one of the target polynomials at the end of a run is 882, with a standard deviation of 200. Moreover, the algorithm typically identifies an

average of 2.24 distinct polynomials per run, with a maximum number of 4, demonstrating its capacity to uncover multiple solutions simultaneously. Finally the algorithm failed at finding a solution in only 3 runs. This is sumed up in Fig. 9. We see that for 77.4% on the time that the algorithm finds more than a single solution, demonstrating the robustness of the technique.



(a) (b) (c)

Fig. 7 `acceptance_ratio` $=$ **0.8 - 0.5** $\times$ **(**`epoch/n_epochs`**)** (a) Evolution of the mean number of particles reproducing each target polynomial (log scale). (b) Evolution of the mean number of particles reproducing a target polynomial, along with its 1σ deviation, for the parameters described in (4.1).

In comparaison with state of the art technique, we have used the AIFeynman algorithm on our data set, with again $x_8 = \exp(\tilde{x}_8)$, such that we again look for polynomial expressions. We have run it with the following configuration

$$\begin{aligned} &\texttt{BF\_try\_time} = 300, \\ &\texttt{polyfit\_deg} = 6, \\ &\texttt{NN\_epochs} = 5000. \end{aligned} \tag{3.14}$$

The first parameter fixes the time limit in seconds for each brute force call, `polyfit_deg` gives the maximum degree of the polynomial tried by the polynomial fit routine and `NN_epochs` is the number of training epoch for the internal neural network. The function used for the brut force tests are

$$\texttt{+*-/><} \sim \texttt{\R1} \tag{3.15}$$

The binary operations are addition, multiplication, substraction and division. The unary ones are inverse, increment, decrement, negation and square root. Finally there is a nonary one, the unity. For more details we refer to [37].

In this algorithm, one tries to fit one of the variables in term of the others. Here, we used it to fit x_{10} in terms of the others. The AIFeynman algorithm finds a solution in TIME :

```
x_10 = 1.414213551821*(x_4-((x_1/x_2)-((x_1/x_2)/x_8))
```
(3.16)

which after identifying the numerical factor with $\sqrt{2}$ and inverting the relation gives

$$-\sqrt{2}x_1 + \sqrt{2}x_1x_8 + x_2x_8x_{10} - \sqrt{2}x_2x_4x_8 = 0 \tag{3.17}$$

which corresponds to p_1 of Eq. (4.5). While the AIFeynman code is able to recover one of the constraints, it exhibits several drawbacks compared to our method. First, it is significantly slower: it required TIME (to be specified) compared to approximately 600 seconds for a single run of our ASMC algorithm. Moreover, the AIFeynman framework is based on expressing one variable as a function of the others, which assumes the invertibility of the underlying relation. In general, this is not guaranteed for the class of polynomials in Eq. (4.5), and obtaining a closed-form inverse can be nontrivial or even impossible for higher-degree expressions. In our case, each polynomial involves at most quadratic terms in any single variable, ensuring invertibility, but this property would not hold for more complex models. In addition, AIFeynman is limited to recovering one constraint per run, whereas our ASMC method can discover multiple constraints simultaneously. This is reflected in our results, where each run identified an average of 2.15 polynomials, with up to 4 found in the best case. Finally, the brute-force regression strategy used by AIFeynman makes it less effective for identifying higher-degree polynomials, whose complexity increases the search difficulty. In contrast, our method treats all polynomials as equally probable, regardless of their degree, enabling it to uncover more intricate structures more efficiently [3].

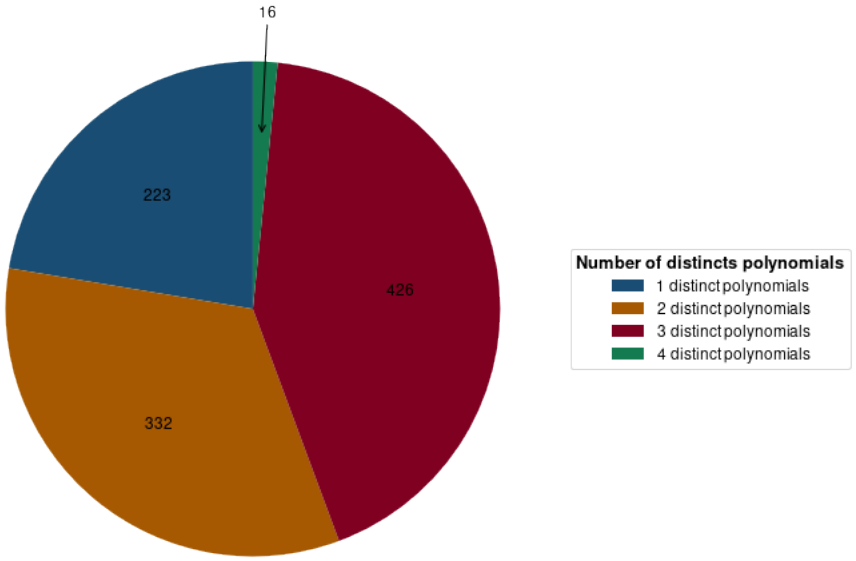


Fig. 8 Pie chart showing the repartition of the number of distinct polynomials found in the runs. Note that there is also the failure case, which corresponds 3 different runs.

[3]Note that we do not claim that we have used the AIFeynman in the most optimal way.

4 Results

Initialisation We initialized the algorithm with

$$\begin{aligned} &\texttt{n\_epochs = 1000},\\ &\texttt{beta = 1e-6},\\ &\texttt{p\_modify = 0.5},\\ &\texttt{p\_multiply = 0.25},\\ &\texttt{p\_divide = 0.25}. \end{aligned} \tag{4.1}$$

This strategy was determined empirically through trial and error; a systematic analysis of its optimization is left for future work. For the representation of polynomials, we adopt a vectorial approach: each polynomial is represented as an array of its coefficients $\{c_k\}$, where $k \in \{1, \dots, n_{\text{mon}}\}$ and n_{mon} denotes the number of available monomials, determined by the chosen maximum degree `max_degree`. We also impose a constraint on the maximum number of terms within a single polynomial to promote sparsity. An alternative approach would be to encourage sparsity through the prior distribution or within the loss function formulation. For the simulations we have done, we chose :

$$\begin{aligned} &\texttt{max\_degree = 4},\\ &\texttt{max\_num\_monomials = 6}. \end{aligned} \tag{4.2}$$

(ce: Give details on the basis of polynomials. 126.) ⟸

Typical run After a run, the 1000 particles are typically distributed into less than 10 different type of polynomials. **(ce: Check the exact number.)** Each particles of a given type share the same monomials, ⟸
but their coefficients fluctuate a bit. We select the representant of each type that features the minimal loss (without the regularisation, $\lambda = 0$, see eq. (3.8)). Here is an example of output for a given run: **(ce: Here** ⟸
`threshold` = 0.1 and `steps` = 1000 for the local search.)

$$1.999\,x_2 - 1.999\,x_2x_8 + 1.414\,x_1x_{10} + 1.999\,x_1x_4 - x_{10}^2x_2x_8, \quad L^{(\lambda=0)} \simeq 2\times10^{-4}, \tag{4.3a}$$
$$2\,x_2 - 1.999\,x_2x_8 + 1.415\,x_1x_8x_{10} + 2.001\,x_1x_4x_8 - 2.002\,x_2x_4^2x_8, \quad L^{(\lambda=0)} \simeq 3\times10^{-3}, \tag{4.3b}$$
$$x_1x_8x_{10} + 1.416\,x_1x_4x_8 - 1.516\,x_2x_4^2x_8, \quad L^{(\lambda=0)} \simeq 3\times10^{2}, \tag{4.3c}$$
$$2\,x_2 - 1.972\,x_2x_8 + 0.816\,x_1x_{10} + 1.369\,x_1x_4 - 0.442\,x_{10}^2x_2, \quad L^{(\lambda=0)} \simeq 3\times10^{2}, \tag{4.3d}$$

where we normalised the coefficients be setting one of them to 1 or 2. Then, we filter out the polynomials with loss higher than 1, as the polynomials (4.3c) and (4.3d), which do not annihilate the data. As discussed in sec. ?? **(ce: to do)**, the coefficients must be combinations of rational numbers or square roots thereof. ⟸
We then deduce from eq. (4.3a) and (4.3b) the candidate annihilator polynomials for the above example:

$$\begin{aligned} &2\,x_2 - 2\,x_2x_8 + \sqrt{2}\,x_1x_{10} + 2\,x_1x_4 - x_2x_8x_{10}^2,\\ &2\,x_2 - 2\,x_2x_8 + \sqrt{2}\,x_1x_8x_{10} + 2\,x_1x_4x_8 - 2\,x_2x_4^2x_8. \end{aligned} \tag{4.4}$$

Statistics We performed 1000 independent runs with the same parameters as above, each involving 1000 particles. It takes approximately 600 s to do a single run on a regular computer. **(ce: That's without** ⟸

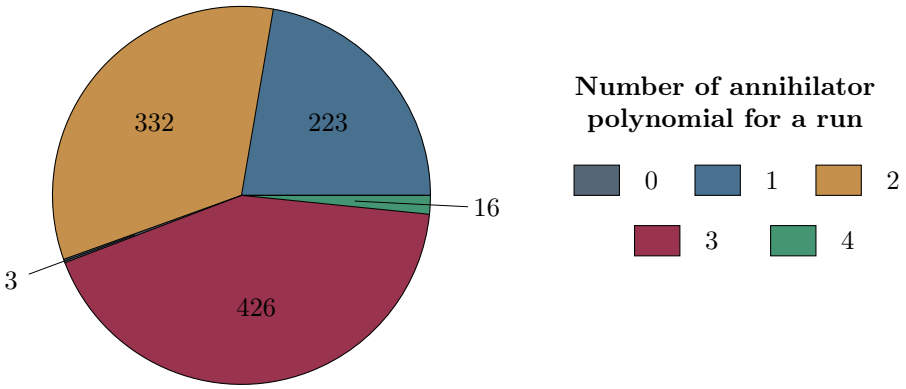


Fig. 9 Pie chart of the repartition of the number of distinct polynomials found in a run.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	$\varnothing$
Frequency	92.6%	75.0%	1.7%	51.9%	0.7%	0.1%	0.2%	0.7%	0.3%
Maximum # of representatives	1000	1000	869	991	710	18	879	207	–

Tab. 1 Statistics of the ability of the ASMC algorithm to produce the polynomials (4.5) on 1000 runs with parameters (4.1) and (4.2). The frequencies represent the percentage of runs featuring a given polynomial in its outputs, and the maximum number of representatives gives the maximum number of particles representing the polynomial in a given run. The column $\varnothing$ counts the runs that failed to produce an annihilator polynomial.

local search, right?) The code finds the 8 different following polynomials:

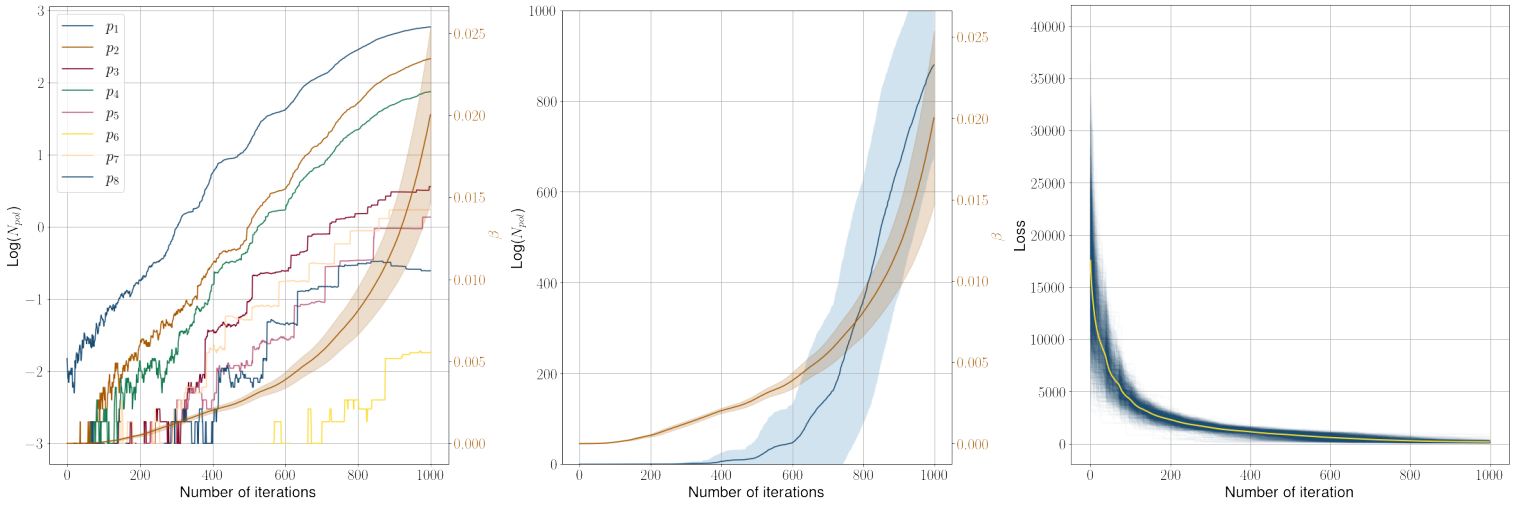
$$p_1 = -\sqrt{2}\,x_1 + \sqrt{2}\,x_1x_8 + x_2x_8x_{10} - \sqrt{2}\,x_2x_4x_8, \tag{4.5a}$$
$$p_2 = 2\,x_2 - 2\,x_2x_8 + \sqrt{2}\,x_1x_{10} + 2\,x_1x_4 - x_2x_8x_{10}^2, \tag{4.5b}$$
$$p_3 = 2\,x_2 - 2\,x_2x_8 + \sqrt{2}\,x_1x_8x_{10} + 2\,x_1x_4x_8 - 2\,x_2x_4^2x_8, \tag{4.5c}$$
$$p_4 = \sqrt{2}\,x_2 - \sqrt{2}\,x_2x_8 + \sqrt{2}\,x_1\,x_4 + x_1x_8x_{10} - x_2x_4x_8x_{10}, \tag{4.5d}$$
$$p_5 = -2\,x_1 - 2\,x_2x_4 - 2\,x_1x_4^2 + 2\,x_1x_8 + \sqrt{2}\,x_2x_{10} + x_1x_8x_{10}^2, \tag{4.5e}$$
$$p_6 = -2\,x_2x_4 - 2\,x_1x_4^2 + 2\,x_2x_4x_8 + \sqrt{2}\,x_2x_{10} - \sqrt{2}\,x_2x_8x_{10} + x_1x_8x_{10}^2, \tag{4.5f}$$
$$p_7 = -\sqrt{2}\,x_1^2x_4 + \sqrt{2}\,x_2^2x_4 + \sqrt{2}\,x_1x_2x_4^2 - x_1^2x_{10} - x_2^2x_{10}, \tag{4.5g}$$
$$p_8 = -2 + 4x_8 - 2x_8^2 + 2x_4^2x_8 - x_8^2x_{10}^2, \tag{4.5h}$$

where $x_8 = e^{\tilde{x}_8}$. They are found with different frequencies with some polynomials occurring more often than others, as reported in tab. 1. The average number of particles reproducing one of the annihilator polynomials at the end of a run is 882, with a standard deviation of 200. Moreover, the algorithm typically identifies an average of 2.24 distinct polynomials per run, with a maximum number of 4, demonstrating its capacity to uncover multiple solutions simultaneously. Finally, the algorithm failed at finding a solution in only 3 runs. This is summed up in fig. 9. We see that for 77.4% on the time that the algorithm finds more than a single solution, demonstrating the robustness of the method.

The dynamics of appearance of each annihilator polynomial is plotted in fig. 10(a), together with the evolution of the inverse temperature β, averaged on the 1000 runs. Note that the dynamics depend strongly

on the polynomial, some of the appearing after only few dozens of epochs, and others just before the end of the runs. **(ce: The polynomials are organize in three classes, the first ones appear after typically a dozen of iteration, the second one after 100 and the last one are very difficult to find and rare. The code favors the polynomials with lowest loss through resampling. Du fait de la structure, quand un polynome est trouvé il est favorisé au détriment des autres (resampling). Faire une figure ?)** The evolution of the inverse temperature, imposed by eq. (3.7), is polynomial. **(ce: Do a fit.)**. **(ce: At the end beta is not 1. This allows to find more than one candidate polynomials, favors the diversity. This is linked to the choice of the acceptance ration at the end. The lower the acceptance ration, the higher beta. If ratio too low, no diversoty. If ratio too high, shitty polynomials.)** The losses of the runs are shown in fig. 10(c). **(ce: Describe more the loss? It is the best loss at each step. Use the loss to show that we indeed converged)**.

(ce: Keep fig. 10(b)?)



(a) (b) (c)

Fig. 10 (a) Dynamics of the mean number of particles reproducing each polynomial in eq. (4.5) (left scale) and evolution of the inverse temperature β and its 1σ deviation (right scale). (b) Evolution of the mean number of particles reproducing a target polynomial, along with its 1σ deviation, for the parameters described in (4.1). (c) Evolution of the losses during the runs (in blue) and their average (in yellow).

(ce: Tests sur le gradient analytique pour voir si c'est un polynome annulateur.)

Comparison with AIFeynman In comparaison with state of the art technique, we have used the AIFeynman algorithm on our data set, with again $x_8 = \exp(\tilde{x}_8)$, such that we again look for polynomial expressions. We have run it with the following configuration

$$\begin{aligned} &\texttt{BF\_try\_time} = 300, \\ &\texttt{polyfit\_deg} = 6, \\ &\texttt{NN\_epochs} = 5000. \end{aligned} \tag{4.6}$$

The first parameter fixes the time limit in seconds for each brute force call, `polyfit_deg` gives the maximum degree of the polynomial tried by the polynomial fit routine and `NN_epochs` is the number of training epoch for the internal neural network. The function used for the brut force tests are

$$\texttt{+*-/><} \sim \texttt{\R1} \tag{4.7}$$

The binary operations are addition, multiplication, substraction and division. The unary ones are inverse, increment, decrement, negation and square root. Finally there is a nonary one, the unity. For more details we refer to [37].

In this algorithm, one tries to fit one of the variables in term of the others. Here, we used it to fit x_{10} in terms of the others. The AIFeynman algorithm finds a solution in TIME :

$$\texttt{x\_10 = 1.414213551821*(x\_4-((x\_1/x\_2)-((x\_1/x\_2)/x\_8))} \tag{4.8}$$

which after identifying the numerical factor with $\sqrt{2}$ and inverting the relation gives

$$-\sqrt{2}x_1 + \sqrt{2}x_1x_8 + x_2x_8x_{10} - \sqrt{2}x_2x_4x_8 = 0 \tag{4.9}$$

which corresponds to p_1 of Eq. (4.5). While the AIFeynman code is able to recover one of the constraints, it exhibits several drawbacks compared to our method. First, it is significantly slower: it required TIME (to be specified) compared to approximately 600 seconds for a single run of our ASMC algorithm. Moreover, the AIFeynman framework is based on expressing one variable as a function of the others, which assumes the invertibility of the underlying relation. In general, this is not guaranteed for the class of polynomials in Eq. (4.5), and obtaining a closed-form inverse can be nontrivial or even impossible for higher-degree expressions. In our case, each polynomial involves at most quadratic terms in any single variable, ensuring invertibility, but this property would not hold for more complex models. In addition, AIFeynman is limited to recovering one constraint per run, whereas our ASMC method can discover multiple constraints simultaneously. This is reflected in our results, where each run identified an average of 2.15 polynomials, with up to 4 found in the best case. Finally, the brute-force regression strategy used by AIFeynman makes it less effective for identifying higher-degree polynomials, whose complexity increases the search difficulty. In contrast, our method treats all polynomials as equally probable, regardless of their degree, enabling it to uncover more intricate structures more efficiently [4].

Supergravity In the previous section, we introduced a numerical method that enabled symbolic regression, yielding a set of polynomials that vanish on our dataset, as presented in Eq. 4.5. We can now solve the system

$$p_i = 0, \quad \forall i \in 1, \dots, 7 \tag{4.10}$$

which leads to the following expressions:

$$\begin{cases} x_1 = \dfrac{x_2}{\sqrt{2}} \dfrac{e^{x_8/2}}{e^{x_8}-1} \Big(- x_5\, e^{x_8/2} + \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)} \Big), \\ x_4 = \dfrac{e^{-x_8/2}}{\sqrt{2}} \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)}. \end{cases} \tag{4.11}$$

[4] Note that we do not claim that we have used the AIFeynman in the most optimal way.

Alternatively, the system can be recast as:

$$\begin{cases} \tilde{x}_8 = \dfrac{x1^2 + x_2^2}{x_2^2 + \left(x_1 - x_2 x_4\right)^2}, \\ x_{10} = \sqrt{2}, x_4, \dfrac{x_2^2 - x_1^2 + x_1 x_2 x_4}{x_1^2 + x_2^2}. \end{cases} \tag{4.12}$$

As anticipated, this defines a three-parameter solution corresponding to a three-dimensional slice of the half-maximal supergravity scalar potential.

Let us make a few remarks at this stage. First, it is possible to find solutions to Eq. A.1 which, however, are not solutions of $\nabla V = 0$. For instance $x_1 = x_2 = 0$; such cases are excluded, as they do not satisfy the stationarity condition and therefore do not correspond to valid physical solutions.

Secondly, one might argue that only two of the seven polynomials are sufficient to fully characterize the solution. That is, choosing any pair $(i, j) \in 1, \ldots, 7$ may suffice to extract a complete description. In practice, this is not entirely accurate. While such a pair can yield partial constraints—for example, recovering Eq.A.3—it may also produce alternative (and potentially less general) parameterizations. Upon inspection, all such partial rules are found to be consistent with, and included in, the most general expressions given in Eq.A.3.

The solution preserves a $\mathrm{U}(1) \times \mathrm{U}(1)$ gauged symmetry.

The (x_1, x_2, x_4) moduli space is most nicely parametrised using the change of coordinates

$$x_1 = r\cos(\theta)\cos(\Phi), \quad x_2 = r\cos(\theta)\sin(\Phi) \quad \text{and} \quad x_4 = r\sin(\theta), \tag{4.13}$$

for which the Zamolodchikov metric reads

$$\mathrm{d}^2 s_{\text{Zam.}} = -\,\mathrm{d}r^2 - r^2\left(\mathrm{d}\theta^2 - r\cos(\theta)\,\mathrm{d}\theta\mathrm{d}\Phi + \sin(\theta)\,\mathrm{d}r\mathrm{d}\Phi + \frac{1}{2}\left(3 + r^2 - \cos(2\theta)\right)\mathrm{d}\Phi^2\right). \tag{4.14}$$

(ce: Tests other change of variables? Test choices of variables other than (x_1, x_2, x_4)?) ⟸

Bosonic spectrum Vector fields:

$$\begin{aligned} m_{(1)}\ell_{\text{AdS}}: \quad & 0\ [2], \quad -2\ [5], \quad 2\ [1], \\ & -1 \pm \sqrt{\left(1 + r^2\right)^2 - 2\,r^2\,\cos(2\theta)}\ [2+2], \\ & 1 \pm \sqrt{1 + 4\,r^2 + r^4}\ [2+2]. \end{aligned} \tag{4.15}$$

The integers between square brackets indicate the multiplicity of each eigenvalue. The spectrum includes two massless vectors corresponding to the unbroken $\mathrm{U}(1) \times \mathrm{U}(1)$ gauge symmetry, although in three dimensions they are non-propagating.

Scalars:

$$\begin{aligned} \left(m_{(0)}\ell_{\text{AdS}}\right)^2: \quad & 0\ [5], \quad 8\ [1], \quad r^2\left(4 + r^2\right)\ [8], \\ & 2r\left(3\,r + r^3 \pm \left(2 + r^2\right)\sqrt{2 + r^2 - 2\,\cos(2\theta)} - r\cos(2\theta)\right)\ [2+2]. \end{aligned} \tag{4.16}$$

Gravitini:

$$m_{(3/2)}\ell_{\rm AdS}: \qquad \frac{1}{2}\left[1 \pm \sqrt{4+2\,r^2+r^4-2r^2\cos(2\theta)}\right]\,[4+4], \tag{4.17}$$

no SUSY enhancements other than $r = 0$.

No dependence on Φ.

5 Conclusion

In this work, we have presented a novel machine learning approach to systematically identify and characterize flat directions in supergravity scalar potentials. Our methodology combines gradient descent sampling with annealed importance sampling for symbolic regression, demonstrating its effectiveness on a 5-scalar subsector of 3d (1,1) supergravity derived from $AdS_3 \times S^3 \times T^4$ compactifications of IIB supergravity.

We developed a robust pipeline that transitions from numerical exploration to analytical understanding. The gradient descent procedure successfully samples the flat direction manifold, while local PCA analysis reveals its intrinsic dimensionality. Most notably, our annealed importance sampling approach to symbolic regression automatically discovers polynomial constraints characterizing the manifold, bypassing the computational complexity that renders direct symbolic manipulation intractable.

Our analysis reveals that the 5-dimensional scalar space contains a 3-dimensional conformal manifold, with one direction potentially gauge-fixable. The discovered solution preserves $U(1)\times U(1)$ gauged symmetry and exhibits a rich spectrum structure. The Zamolodchikov metric on the moduli space, parametrized by spherical-like coordinates, provides a concrete geometric description of the conformal manifold.

The method demonstrates remarkable efficiency and reliability. The algorithm discovers eight distinct polynomial relations with varying frequencies, suggesting a hierarchical structure in constraint discovery. Futhermore, upon 1000 the algorithm failed at finding any solution in only 3 cases, demonstrating the robustness of the method.

This approach opens several promising avenues for advancement. The scalability to higher-dimensional cases represents the most immediate challenge and opportunity. By increasing the number of particles, refining the annealing schedule, and optimizing polynomial search strategies, we anticipate extending this methodology to the full 13-scalar theory and potentially to other supergravity models.

Beyond technical improvements, this work suggests a broader paradigm for exctrating conformal manifolds in string theory and supergravity. This analytical constraints from numerical data could prove invaluable for classifying flat directions across different theories, understanding moduli stabilization mechanisms, and exploring the landscape of consistent supergravity backgrounds.

The marriage of machine learning techniques with traditional supergravity analysis represents a step toward more systematic approaches to understanding the rich geometric structures underlying these theories. As computational power increases and algorithms improve, we envision this methodology becoming a standard tool for exploring the intricate relationships between geometry, symmetry, and dynamics in supergravity theories.

While the specific solution found in this 5-scalar model may have limited direct physical applications, the demonstrated feasibility of our approach and its potential for systematic classification of flat directions across the supergravity landscape make it a valuable addition to the theoretical physicist's toolkit.

A Supergravity solutions

In the previous section, we introduced a numerical method that enabled symbolic regression, yielding a set of polynomials that vanish on our dataset, as presented in Eq. 4.5. We can now solve the system

$$p_i = 0, \quad \forall i \in 1, \dots, 7 \tag{A.1}$$

which leads to the following expressions:

$$\begin{cases} x_1 = \dfrac{x_2}{\sqrt{2}} \dfrac{e^{x_8/2}}{e^{x_8}-1} \Big(- x_5\, e^{x_8/2} + \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)} \Big), \\ x_4 = \dfrac{e^{-x_8/2}}{\sqrt{2}} \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)}. \end{cases} \tag{A.2}$$

Alternatively, the system can be recast as:

$$\begin{cases} \tilde{x}_8 = \dfrac{x1^2 + x_2^2}{x_2^2 + \big(x_1 - x_2 x_4\big)^2}, \\ x_{10} = \sqrt{2}, x_4, \dfrac{x_2^2 - x_1^2 + x_1 x_2 x_4}{x_1^2 + x_2^2}. \end{cases} \tag{A.3}$$

As anticipated, this defines a three-parameter solution corresponding to a three-dimensional slice of the half-maximal supergravity scalar potential.

Let us make a few remarks at this stage. First, it is possible to find solutions to Eq. A.1 which, however, are not solutions of $\nabla V = 0$. For instance $x_1 = x_2 = 0$; such cases are excluded, as they do not satisfy the stationarity condition and therefore do not correspond to valid physical solutions.

Secondly, one might argue that only two of the seven polynomials are sufficient to fully characterize the solution. That is, choosing any pair $(i, j) \in 1, \dots, 7$ may suffice to extract a complete description. In practice, this is not entirely accurate. While such a pair can yield partial constraints—for example, recovering Eq.A.3—it may also produce alternative (and potentially less general) parameterizations. Upon inspection, all such partial rules are found to be consistent with, and included in, the most general expressions given in Eq.A.3.

The solution preserves a $\mathrm{U}(1) \times \mathrm{U}(1)$ gauged symmetry.

The (x_1, x_2, x_4) moduli space is most nicely parametrised using the change of coordinates

$$x_1 = r\cos(\theta)\cos(\Phi), \quad x_2 = r\cos(\theta)\sin(\Phi) \quad \text{and} \quad x_4 = r\sin(\theta), \tag{A.4}$$

for which the Zamolodchikov metric reads

$$\mathrm{d}^2 s_{\text{Zam.}} = -\,\mathrm{d}r^2 - r^2 \left(\mathrm{d}\theta^2 - r\cos(\theta)\,\mathrm{d}\theta\mathrm{d}\Phi + \sin(\theta)\,\mathrm{d}r\mathrm{d}\Phi + \frac{1}{2}\big(3 + r^2 - \cos(2\theta)\big)\,\mathrm{d}\Phi^2 \right). \tag{A.5}$$

(ce: Tests other change of variables? Test choices of variables other than (x_1, x_2, x_4)?) ⟸

Bosonic spectrum Vector fields:

$$\begin{aligned} m_{(1)}\ell_{\rm AdS}: \quad & 0\ [2], \quad -2\ [5], \quad 2\ [1], \\ & -1 \pm \sqrt{(1+r^2)^2 - 2\,r^2\,\cos(2\theta)}\ [2+2], \\ & 1 \pm \sqrt{1+4\,r^2+r^4}\ [2+2]. \end{aligned} \tag{A.6}$$

The integers between square brackets indicate the multiplicity of each eigenvalue. The spectrum includes two massless vectors corresponding to the unbroken $\mathrm{U}(1) \times \mathrm{U}(1)$ gauge symmetry, although in three dimensions they are non-propagating.

Scalars:

$$\begin{aligned} (m_{(0)}\ell_{\rm AdS})^2: \quad & 0\ [5], \quad 8\ [1], \quad r^2\,\left(4+r^2\right)\ [8], \\ & 2r\left(3\,r + r^3 \pm (2+r^2)\,\sqrt{2+r^2-2\,\cos(2\theta)} - r\cos(2\theta)\right)\ [2+2]. \end{aligned} \tag{A.7}$$

Gravitini:

$$m_{(3/2)}\ell_{\rm AdS}: \qquad \frac{1}{2}\left[1 \pm \sqrt{4+2\,r^2+r^4-2r^2\cos(2\theta)}\right]\ [4+4], \tag{A.8}$$

no SUSY enhancements other than $r = 0$.

No dependence on Φ.

B Details on the algorithm

Algorithm 1 Local Dimension Estimation using PCA

Function LocalDim1Point(x, var_thres)
    Apply PCA to data matrix x
    Compute explained variance ratios: lambda_1, lambda_2, ..., lambda_d
    Compute cumulative variance: cumvar_k = $\sum_{i=1}^{k}$ lambda_i for k = 1, ..., d
    Find dim = min{k : cumvar_k $\geq$ var_thres}
    return dim

Function LocalDimNPoints(data, n_neig, var_thres)
    n_points $\leftarrow$ number of rows in data
    Initialize results $\leftarrow \emptyset$
    Compute neighbors_idx for all points in data
for i = 1 to n_points **do** ▷ Compute local dimensions for all points
    result_i $\leftarrow$ ComputeLocalDim(i, data, neighbors_idx, var_thres)
    Append result_i to results
end for
    return results

In this appendix, we detail the different algorithm that we used. We first here provide a pseudo-code for the AIS-SMC algorithm :

This pseudo code sketch how does our annealing loops work. Here is the pseudo-code for the initialization of each individual polynomial. We chose to generate more terms with a high number of monomials, because in principle there are more possibilities of polynomials with larger number of monomials. This does not prevent the code from finding polynomials with lower number of monomials as we can see on the results. The polynomial that the code find the most often is the polynomial p_1 which only has 4 terms instead of 5 or 6 for all others polynomials. The code can find such polynomials by putting gradually one or several of the coefficients to zero. Note also that we could include the possibility in the MH process too add/remove monomials. However, in the spirit of making "small" steps in the search space, we decided not to for now.

References

[1] J. M. Maldacena, *The Large N limit of superconformal field theories and supergravity*, *Adv. Theor. Math. Phys.* **2** (1998) 231–252, [hep-th/9711200].

[2] H. Ooguri and C. Vafa, *Non-supersymmetric AdS and the Swampland*, *Adv. Theor. Math. Phys.* **21** (2017) 1787–1801, [arXiv:1610.01533].

[3] E. Palti, *The Swampland: Introduction and Review*, *Fortsch. Phys.* **67** (2019), no. 6 1900037, [arXiv:1903.06239].

[4] A. Giambrone, A. Guarino, E. Malek, H. Samtleben, C. Sterckx, and M. Trigiante, *Holographic evidence for nonsupersymmetric conformal manifolds*, *Phys. Rev. D* **105** (2022), no. 6 066018, [arXiv:2112.11966].

Algorithm 2 Annealing Importance Sampling for Polynomial Discovery

Require: `n_iter` (number of iterations), `n_particles` (number of particles), `target_acc_rate`$(\cdot,\cdot)$ (target acceptance rate), `adaptation_strength` (temperature adaptation parameter)

Initialize particles $\{\texttt{z\_0}^{(k)}\}_{k=1}^{\texttt{n\_particles}}$ with sparse polynomials
Set $\texttt{w\_0}^{(\texttt{k})} = 1/\texttt{n\_particles}$ and $\texttt{w\_unorm\_0}^{(\texttt{k})} = 1$ for all `k` ▷ normalized and unnormalized weights
`z_prev` ← `z_0`, `best_loss` ← ∞, `acceptance_history` ← $\emptyset$, $\texttt{q\_ratio}^{(\texttt{k})} = 1$ for all k, `beta_0` ← `init_beta` ▷ q is the proposal ratio
for `i = 1` to `n_iter` **do** ▷ Temperature adaptation
    `beta_prev` ← `beta_current`
    `avg_acc` ← mean(`acceptance_history`) ▷ only of the last 5 epochs
    if `avg_acc < target_acc_rate(i, n_iter) - 0.05` **then**
        `beta_current` ← `beta_current` × `(1 - adaptation_strength)`
    else if `avg_acc > target_acc_rate(i, n_iter) + 0.05` **then**
        `beta_current` ← `beta_current` × `(1 + adaptation_strength)`
    end if
    for `k = 1` to `n_particles` **do** ▷ Reweighting step
        Compute $\texttt{log(gamma\_i(z\_i}^{(\texttt{k})}\texttt{))} = \texttt{- beta\_current} \times \texttt{loss}(\texttt{z\_i}^{(\texttt{k})})$
        Compute $\texttt{log(gamma\_(i-1)(z\_(i-1)}^{(\texttt{k})}\texttt{))} = \texttt{- beta\_prev} \times \texttt{loss}(\texttt{z\_prev}^{(\texttt{k})})$
        $\Delta \log \texttt{w\_i}^{(\texttt{k})} \leftarrow \texttt{log(gamma\_i(z\_i}^{(\texttt{k})}\texttt{) - log(gamma\_(i-1)(z\_(i-1)}^{(\texttt{k})}\texttt{) + log(q\_ratio}^{(\texttt{k})}\texttt{)}$
    end for
    Update unnormalized weights: $\texttt{w\_unorm\_i}^{(\texttt{k})} \leftarrow \texttt{w\_unorm\_(i-1)}^{(\texttt{k})} \times \texttt{exp}(\Delta\texttt{log w\_i}^{(\texttt{k})})$
    Normalize weights: $\texttt{w\_i}^{(\texttt{k})} \leftarrow \texttt{w\_unorm\_i}^{(\texttt{k})} / \sum_{\texttt{j=1}}^{\texttt{n\_particles}} \texttt{w\_unorm\_i}^{(\texttt{j})}$
    Compute ESS: $\texttt{ESS = 1 } / \sum_{\texttt{k=1}}^{\texttt{n\_particles}} (\texttt{w\_i}^{(\texttt{k})})^2$
    if `ESS < n_particles / 2` **then** ▷ Resampling
        Resample particles: $\{\texttt{z\_i}^{(\texttt{k})}\}_{\texttt{k=1}}^{\texttt{n\_particles}} \sim \texttt{Multinomial}(\texttt{n\_particles}, \{\texttt{w\_i}^{(\texttt{k})}\}_{\texttt{k=1}}^{\texttt{n\_particles}})$
        Reset weights: $\texttt{w\_i}^{(\texttt{k})} = \texttt{1/n\_particles}$, $\texttt{w\_unorm\_i}^{(\texttt{k})} = \texttt{1}$ for all `k`
        Update `z_prev` accordingly
    end if
    `z_before_mutation` ← `z_i` (save current particles)
    for `k = 1` to `n_particles` **do** ▷ Mutating with Metropolis-Hasting
        Propose $\texttt{z}'^{(\texttt{k})} \sim \texttt{q}(\texttt{z}'^{(\texttt{k})} | \texttt{z\_i}^{(\texttt{k})})$ using specialized MH proposal
        Compute $\texttt{log(p\_current}^{(\texttt{k})}\texttt{)} = \texttt{- beta\_current} \times \texttt{loss}(\texttt{z\_i}^{(\texttt{k})}) + \texttt{sparsity\_prior}(\texttt{z\_i}^{(\texttt{k})})$
        Compute $\texttt{log(p\_proposed}^{(\texttt{k})}\texttt{)} = \texttt{- beta\_current} \times \texttt{loss}(\texttt{z}'^{(\texttt{k})}) + \texttt{sparsity\_prior}(\texttt{z}'^{(\texttt{k})})$
        $\texttt{log(alpha}^{(\texttt{k})}\texttt{)} \leftarrow \texttt{log(p\_proposed}^{(\texttt{k})}\texttt{) - log(p\_current}^{(\texttt{k})}\texttt{) + log(q\_ratio}^{(\texttt{k})}\texttt{)}$
        Generate $\texttt{u}^{(\texttt{k})} \sim \mathcal{U}(\texttt{0,1})$
        if $\texttt{u}^{(\texttt{k})} < \texttt{alpha}^{(\texttt{k})}$ **then**
            $\texttt{z\_i}^{(\texttt{k})} \leftarrow \texttt{z}'^{(\texttt{k})}$ (accept proposal)
        end if
    end for
    `acceptance_rate` ← fraction of accepted proposals
    Add `acceptance_rate` to `acceptance_history`
    `z_prev` ← `z_before_mutation` (update for next iteration)
end for
return `final particles, final losses, final weights`

Algorithm 3 Initialize Sparse Polynomials

Require: `n_particles` (number of particles), `max_degree` (maximum polynomial degree), `num_vars` (number of variables), `max_num_monomials` (maximum number of monomials per polynomial), `num_terms` (total number of possible terms)

`particles` $\leftarrow \mathbf{0}_{\texttt{n\_particles} \times \texttt{num\_terms}}$
`max_monomials` $\leftarrow$ `max_num_monomials`
`n_mon_to_max_order` $\leftarrow \binom{\texttt{max\_degree + num\_vars}}{\texttt{max\_degree}}$ ▷ Compute total number of possible monomials up to max degree
`n_mon_all_coeff` $\leftarrow \emptyset$
`n_tot_pol` $\leftarrow 0$
for `i = 1` to `max_num_monomials` **do** ▷ Compute probability distribution over number of monomials
    `n_mon_i_coeff` $\leftarrow \binom{\texttt{n\_mon\_to\_max\_order}}{i}$ ▷ Number of polynomials with exactly i monomials
    `n_tot_pol` $\leftarrow$ `n_tot_pol + n_mon_i_coeff`
    Append `n_mon_i_coeff` to `n_mon_all_coeff`
end for
`proba` $\leftarrow$ `n_mon_all_coeff / n_tot_pol` ▷ Normalize to get probabilities
for `i = 1` to `n_particles` **do** ▷ Generate particles
    `num_monomials` $\sim$ `Categorical({1, 2, ..., max_monomials}, proba)` ▷ Sample number of monomials according to computed distribution
    `nonzero_indices` $\leftarrow$ sample `num_monomials` indices from $\{$`1, 2, ...,num_terms`$\}$ without replacement ▷ Randomly select which monomials will be non-zero
    for `j` $\in$ `nonzero_indices` **do** ▷ Generate random coefficients
        `particles[i, j]` $\leftarrow \mathcal{U}(-2, 2)$ ▷ Uniform in $[-2, 2]$
    end for
end for
    return `particles`

[5] O. Aharony, M. Berkooz, and E. Silverstein, *Nonlocal string theories on AdS(3) x S**3 and stable nonsupersymmetric backgrounds*, *Phys. Rev. D* **65** (2002) 106007, [hep-th/0112178].

[6] X. Dong, D. Z. Freedman, and Y. Zhao, *Explicitly Broken Supersymmetry with Exactly Massless Moduli*, *JHEP* **06** (2016) 090, [arXiv:1410.2257].

[7] C. Eloy and G. Larios, *Charting the Conformal Manifold of Holographic CFT_2's*, arXiv:2405.17542.

[8] I. M. Comsa, M. Firsching, and T. Fischbacher, *SO(8) Supergravity and the Magic of Machine Learning*, *JHEP* **08** (2019) 057, [arXiv:1906.00207].

[9] D. Berman, T. Fischbacher, G. Inverso, B. Scellier, and B. Scellier, *Vacua of ω-deformed SO(8) supergravity*, *JHEP* **06** (2022) 133, [arXiv:2201.04173].

[10] A. Ashmore, Y.-H. He, and B. A. Ovrut, *Machine Learning Calabi–Yau Metrics*, *Fortsch. Phys.* **68** (2020), no. 9 2000068, [arXiv:1910.08605].

[11] D. S. Berman, Y.-H. He, and E. Hirst, *Machine learning Calabi-Yau hypersurfaces*, *Phys. Rev. D* **105** (2022), no. 6 066002, [arXiv:2112.06350].

[12] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, *Numerical metrics for complete intersection and Kreuzer–Skarke Calabi–Yau manifolds*, *Mach. Learn. Sci. Tech.* **3** (2022), no. 3 035014, [arXiv:2205.13408].

[13] P. Berglund, G. Butbaia, T. Hüubsch, V. Jejjala, D. Mayorga Peña, C. Mishra, and J. Tan, *Machine-learned Calabi–Yau metrics and curvature*, *Adv. Theor. Math. Phys.* **27** (2023), no. 4 1107–1158, [arXiv:2211.09801].

[14] V. Jejjala, D. K. Mayorga Pena, and C. Mishra, *Neural network approximations for Calabi-Yau metrics*, *JHEP* **08** (2022) 105, [arXiv:2012.15821].

[15] M. R. Douglas, R. L. Karp, S. Lukic, and R. Reinbacher, *Numerical Calabi-Yau metrics*, *J. Math. Phys.* **49** (2008) 032302, [hep-th/0612075].

[16] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, *Learning Size and Shape of Calabi-Yau Spaces*, arXiv:2111.01436.

[17] Y.-H. He, *The Calabi–Yau Landscape: From Geometry, to Physics, to Machine Learning*. Lecture Notes in Mathematics. 5, 2021.

[18] Y.-H. He, *Deep-Learning the Landscape*, arXiv:1706.02714.

[19] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, *Machine Learning in the String Landscape*, *JHEP* **09** (2017) 157, [arXiv:1707.00655].

[20] F. Ruehle, *Evolving neural networks with genetic algorithms to study the String Landscape*, *JHEP* **08** (2017) 038, [arXiv:1706.07024].

[21] H.-Y. Chen, Y.-H. He, S. Lal, and M. Z. Zaz, *Machine Learning Etudes in Conformal Field Theories*, arXiv:2006.16114.

[22] N. Brady, D. Tennyson, and T. Vandermeulen, *Machine Learning the 6d Supergravity Landscape*, arXiv:2505.16131.

[23] C. Krishnan, V. Mohan, and S. Ray, *Machine Learning $\mathcal{N} = 8, D = 5$ Gauged Supergravity*, *Fortsch. Phys.* **68** (2020), no. 5 2000027, [arXiv:2002.12927].

[24] K. Hashimoto, S. Sugishita, A. Tanaka, and A. Tomiya, *Deep learning and the AdS/CFT correspondence*, *Phys. Rev. D* **98** (2018), no. 4 046019, [arXiv:1802.08313].

[25] F. Ruehle, *Data science applications to string theory*, *Phys. Rept.* **839** (2020) 1–117.

[26] Y.-H. He, E. Heyes, and E. Hirst, *Machine learning in physics and geometry*, *Handbook of Statistics* **49** (2023) 47–81, [arXiv:2303.12626].

[27] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, *Hilbert series, machine learning, and applications to physics*, *Phys. Lett. B* **827** (2022) 136966, [arXiv:2103.13436].

[28] J. R. Koza, *Genetic programming as a means for programming computers by natural selection*, *Statistics and computing* **4** (1994) 87–112.

[29] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. Bosman, *Improving model-based genetic programming for symbolic regression of small expressions*, *Evolutionary computation* **29** (2021), no. 2 211–237.

[30] D. L. Randall, T. S. Townsend, J. D. Hochhalter, and G. F. Bomarito, *Bingo: a customizable framework for symbolic regression with genetic programming*, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2282–2288, 2022.

[31] B. Burlacu, G. Kronberger, M. Kommenda, and M. Affenzeller, *Parsimony measures in multi-objective genetic programming for symbolic regression*, in *Proceedings of the genetic and evolutionary computation conference companion*, pp. 338–339, 2019.

[32] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, *Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients*, *arXiv preprint arXiv:1912.04871* (2019).

[33] P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, *End-to-end symbolic regression with transformers*, *Advances in Neural Information Processing Systems* **35** (2022) 10269–10281.

[34] M. Valipour, B. You, M. Panju, and A. Ghodsi, *Symbolicgpt: A generative transformer model for symbolic regression*, *arXiv preprint arXiv:2106.14131* (2021).

[35] Z. Bastiani, R. M. Kirby, J. Hochhalter, and S. Zhe, *Diffusion-based symbolic regression*, *arXiv preprint arXiv:2505.24776* (2025).

[36] S. S. Sahoo, C. H. Lampert, and G. Martius, *Learning Equations for Extrapolation and Control*, *arXiv e-prints* (June, 2018) arXiv:1806.07259, [arXiv:1806.07259].

[37] S.-M. Udrescu and M. Tegmark, *AI Feynman: a Physics-Inspired Method for Symbolic Regression*, *Sci. Adv.* **6** (2020), no. 16 eaay2631, [arXiv:1905.11481].

[38] R. M. Neal, *Annealed importance sampling*, 1998.

[39] P. Del Moral, A. Doucet, and A. Jasra, *Sequential monte carlo samplers*, *Journal of the Royal Statistical Society Series B: Statistical Methodology* **68** (2006), no. 3 411–436.

[40] C. Eloy, G. Larios, and H. Samtleben, *Triality and the consistent reductions on $AdS_3 \times S^3$*, *JHEP* **01** (2022) 055, [arXiv:2111.01167].

[41] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[42] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, arXiv:1608.03983.

[43] R. J. G. B. Campello, D. Moulavi, and J. Sander, *Density-based clustering based on hierarchical density estimates*, in *Advances in Knowledge Discovery and Data Mining* (J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, eds.), (Berlin, Heidelberg), pp. 160–172, Springer Berlin Heidelberg, 2013.