MI-HET-???
May 21, 2025

Notes: Machine learning flat directions

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

2 Supergravity setup

- Context: 6d $\mathrm{AdS}_3 \times S^3$, truncation to 3d, potential, conformal manifold.
- Choice of truncation from 32 to 13 to 5 variables.

Potential The 22 scalars of the theory can be parametrised by ($22 = 32 - 10$, with 10 scalars gauge fixed using translations in the gauge group)

- $m = \nu\nu^T \in \mathrm{GL}(3,\mathbb{R})$ 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 $\tilde{\varphi}$.

We can further restrict ourselves to a set of 13 scalars (see ref. [**?**]), with

$$\xi = \begin{pmatrix} 0 & 0 & 0 & x_1 \\ 0 & 0 & 0 & x_2 \\ 0 & 0 & 0 & x_3 \end{pmatrix}, \quad \phi = \begin{pmatrix} 0 & x_4 & x_5 \\ -x_4 & 0 & x_6 \\ -x_5 & -x_6 & 0 \end{pmatrix}, \quad \tilde{\varphi} = x_{13}$$
$$\nu = e^{(6\,x_7+3\,x_8+\sqrt{3}\,x_9)/6} \begin{pmatrix} 1 & \frac{x_{10}}{\sqrt{2}} & \frac{x_{11}}{\sqrt{2}} + \frac{x_{10}x_{12}}{4} \\ 0 & e^{-x_8} & \frac{e^{-x_8}\,x_{12}}{\sqrt{2}} \\ 0 & 0 & e^{-(x_8+\sqrt{3}\,x_9)/2} \end{pmatrix}. \tag{2.1}$$

Scalar potential from ref. [**?**]:

$$
\begin{aligned}
V = 4\,e^{-4\tilde{\varphi}} + 2\,e^{-2\tilde{\varphi}}\Big[&- \mathrm{tr}\left(m + m^{-1}\right) + \mathrm{tr}\left(\phi m^{-1}\phi\right) - 2\ \mathrm{tr}\left(\phi m^{-1}\xi^2\right) - 2\ \mathrm{tr}\left(\xi^2\right) \\
&- \mathrm{tr}\left(\xi^2 m^{-1}\xi^2\right) + \frac{1}{2}\ \det\left(m^{-1}\right)\left(1 - \mathrm{tr}\left(\phi^2\right) - \mathrm{tr}\left(\xi^4\right) + \mathrm{tr}\left(\xi^2\right)^2\right) \\
&+ \frac{1}{2}\,\mathrm{T}\left(m^{-1}(\xi^2 - \phi), (\xi^2 + \phi)m^{-1}, m + (\xi^2 + \phi)m^{-1}(\xi^2 - \phi) + 2\,\xi^2\right) \\
&+ \frac{1}{4}\,\mathrm{T}\left(m^{-1}, m + (\xi^2 + \phi)m^{-1}(\xi^2 - \phi) + 2\,\xi^2, m + (\xi^2 + \phi)m^{-1}(\xi^2 - \phi) + 2\,\xi^2\right)\Big],
\end{aligned}
\tag{2.2}
$$

where $\mathrm{T}\,(A, B, C) = \varepsilon_{mnp}\,\varepsilon_{qrs}\,A^{mq}B^{nr}C^{ps}$.

As a first simplified example we have considered only the parameters x_1, x_2, x_4, x_8 and x_{10}. The potential (2.2) becomes

$$
\begin{aligned}
V = \frac{1}{8}\,e^{-2\,x_8}\Big[&4 + 4\,x_1^4 + e^{4\,x_8}\left(2 + x_{10}^2\right)^2\left(1 + x_2^4\right) \\
&- 4\,e^{3\,x_8}\left(2 + x_{10}^2\right)\left(2 - x_1^2 + \sqrt{2}\,x_1 x_{10} x_2^3 + x_2^4 + 2\,x_1 x_2 x_4 + x_4^2 - x_2^2\left(1 - x_1^2 + x_4^2\right)\right) \\
&- 8\,e^{x_8}\left(2 + x_1^4 + \sqrt{2}\,x_1^3 x_{10} x_2 - x_2^2 - 2\,x_1 x_2 x_4 + x_4^2 - x_1^2\left(1 - x_2^2 + x_4^2\right)\right) \\
&+ 4\,e^{2\,x_8}\left(2 - 4\,x_1^2 + x_1^4 - 4\,x_2^2 + 4\,x_1^2 x_2^2 + x_2^4 + x_{10}^2\left(1 + 3\,x_1^2 x_2^2\right) + 4\,x_4^2 + x_4^4\right) \\
&+ 8\sqrt{2}\,x_{10}\,e^{2\,x_8}\left(x_1^3 x_2 + x_1^2 x_4 - x_2^2 x_4 + x_1\left(x_2^3 - x_2 x_4^2\right)\right)\Big].
\end{aligned}
\tag{2.3}
$$

(ce: Change notation to keep x for the polynomial variables?) ⟸

(ce: Discuss here why we need numerical methods: to complicated for mathematica to simplify the gradient.) ⟸

3 Numerical analysis

3.1 Gradient descent and local analysis

Gradient descent: implementation, sampling, time needed, loss 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.

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 $[-1, 1]$. 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. The loss function is defined as:

$$
\mathcal{L} = \sum_{i=1}^{n_{points}} ||\nabla V(\vec{x}^{,i})||^2 \tag{3.1}
$$

where $\vec{x}^{,i} = (x_1^i, x_2^i, x_4^i, x_8^i, x_{10}^i)$ denotes the data points.

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 [**?**]. 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. 1. 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.

Although this strategy introduces the risk of desampling certain regions in favor of others, we did not observe any such issue in the remainder of our analysis.

The Adam optimizer was used throughout the gradient descent process. We reinitialized it at steps {250,500,750}, and fixing the learning rate at 10^{-2}. Then we reinitialized the optimizer at step 1000 but fix the learning rate at 10^{-3}. Finally we reinitialize the optimizer at step 1500 with learning rate 10^{-4} and let it run for 1000 more epochs. At the end of the run, we found that all points sit close to the value -4, and that x% of the data points are lying at a value with an error of xx. Similarly, we found that x% of the data has a gradient smaller than xx.

As a first visualization, we present a triangular plot of the data in Fig. 2. 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/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 alll directions appear to be well populated. Furthermore, we can notice that a large majority of the points converge to the origin $\vec{0}$. This is not surprising, as this just mean that the round sphere solution has a larger basin of attraction, which sounds reasonable. We do not however attempt to shift the points along flat directions to depopulate the origin or repopulate other regions.

Local PCA: graphs of number of patches with given local dimension, and with varying size of local patch, discuss optimal patch size (we want at least few points in each direction) Once the gradient descent has been completed and the flat direction 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 whether it consists of a single connected component or multiple disjoint components (e.g., two intersecting hyperplanes).

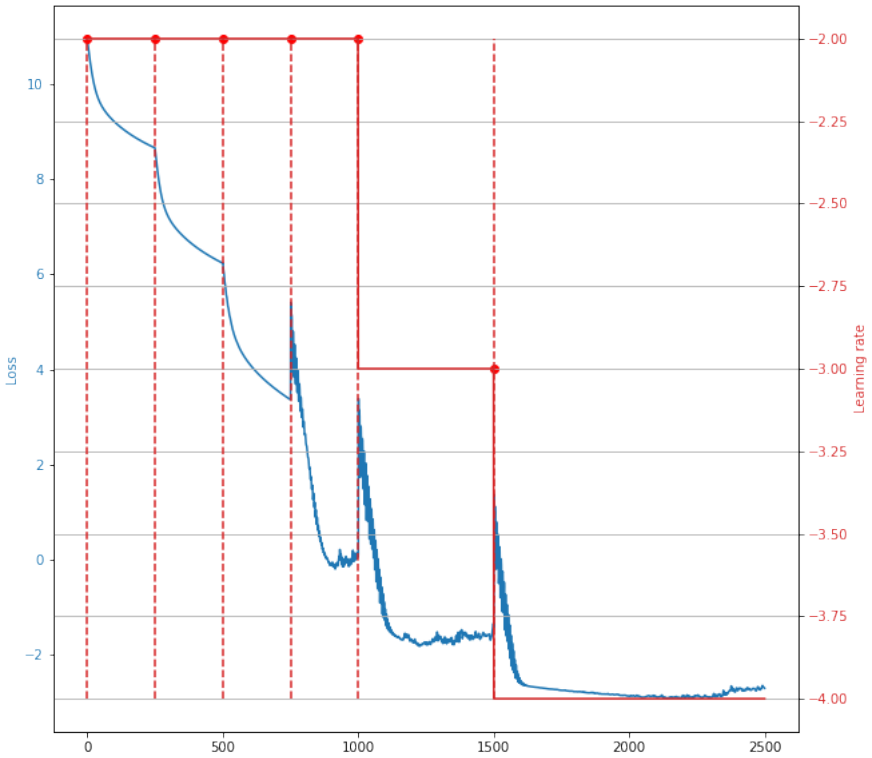

Fig. 1 Loss function for the gradient descent as well as the learning rate schedule, both in log scale. The red dashed lines correspond to epochs when the optimizer is reinitialized.

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 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. 3.

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.

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 conclude that the manifold under investigation has an intrinsic dimension of 3.

Clustering: HDBScan (only parameter: minimal number of points in a cluster), and graphs of 3d slices of the 5d space 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

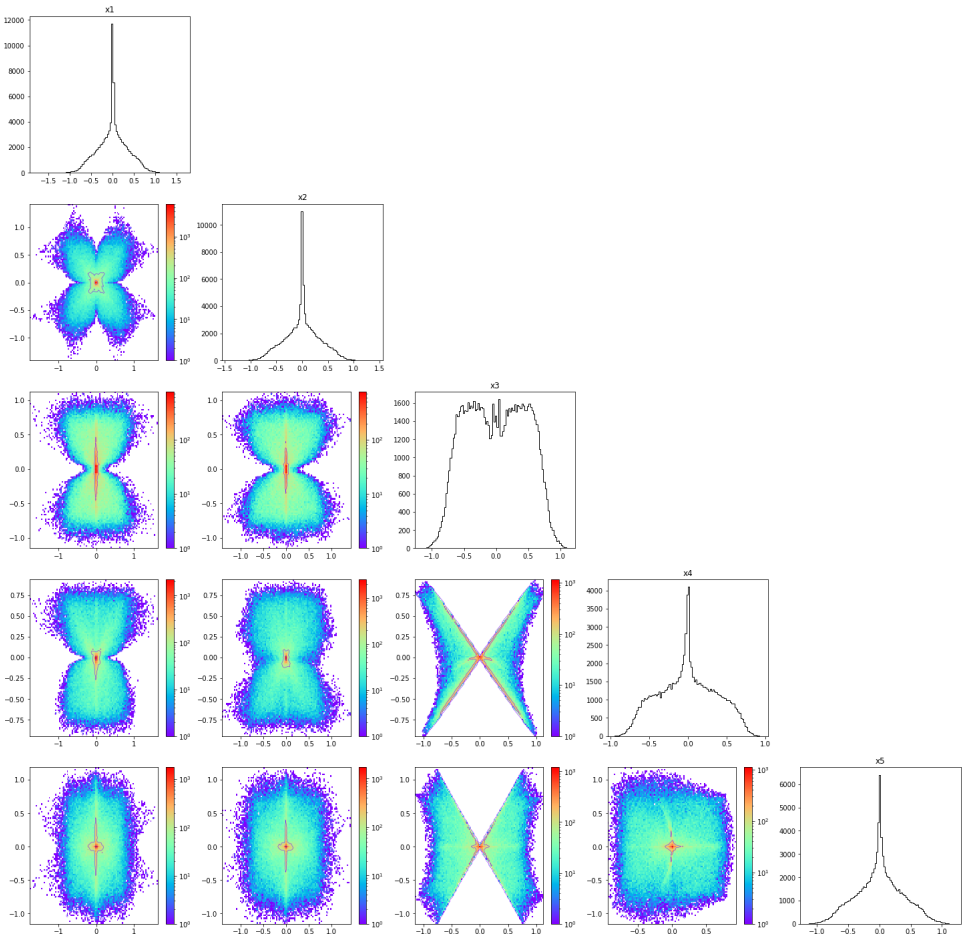

Fig. 2 Triangular plot showing $2d$ projections and $1d$ histograms of the data after the gradient descent

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. For the purpose of the clustering, we use the HDBSCAN algorithm[1].

We set the minimum cluster size to 10 points. Out of the 986,060 3-dimensional points, the algorithm identifies three clusters: one of size 10, one of size 91,747, and one of size 6,849. **(ce: Are those numbers right? We loose more than 800,000 points?)** ⟸

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. 4, 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

[1]Details of the algorithm can be found in the original HDBSCAN paper or documentation. 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.

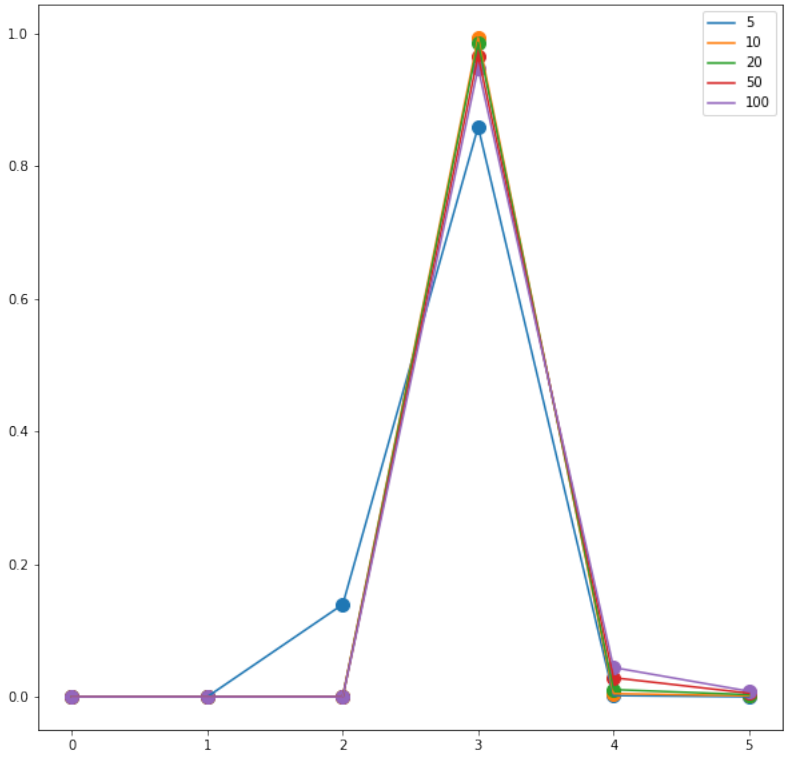

Fig. 3 Results of the local PCA analysis. The x-axis shows the dimension inferred by the algorithm, and the y-axis indicates the proportion of points for which that dimension was found. Each curve corresponds to a different value of k in the k-nearest neighbours.

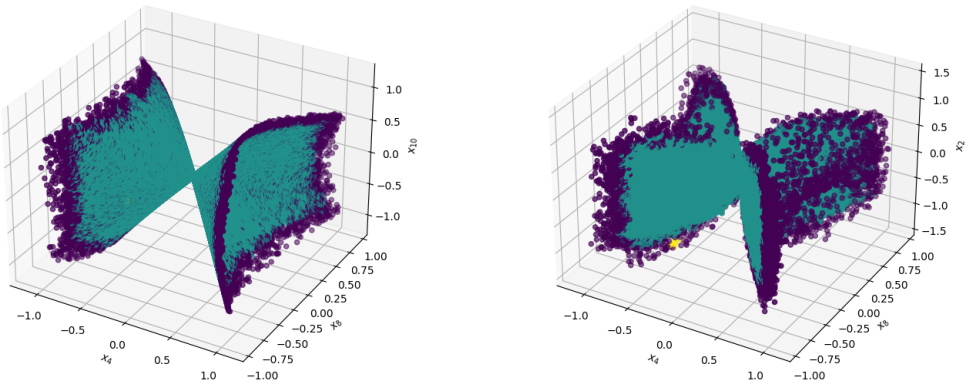

Fig. 4 3D plots of the data in selected coordinates. Left: (x_4, x_8, x_{10}). Right: (x_2, x_4, x_8).

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 single, connected, three-dimensional manifold.

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

3.2 Annealed Importance Sampling for polynomial symbolic regression

- We can convert the potential to a polynomial by converting the variables that appear in exponentials to logs.

- As the potential is a polynomial, the solutions satisfy polynomials equations. Discuss that fact that we could get polynomials directly by taking the gradient of the potential, but those will be to complicated to express the vacuum in a usable way. We search those using Annealed Importance Sampling.

  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 constraints on the embedding coordinates $\vec{x} = (x_1, x_2, x_4, x_8, x_{10})$. If we have a look at the form of the potential (2.3), we see that if we use $\tilde{x}_8 = e^{x_8}$, that, up to a global e^{-2x_8} factor, 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 are two of those. Of course if one takes directly the gradient of (2.3), 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. There are a number of way one can deal with it, using already existing methods such as AI Feynman methods [**?**], or ..., or by doing rough gradient descent on the most general polynomial with arbitrary coefficients and trying to minimize its value once evaluated on the data points. In the spirit of trying to build a generalisable method, we develop here our own method, which is based on symbolic regression using an Annealing Importance Sampling method.

- Annealed Importance Sampling: first explain general idea (construct density probability, role of temperature, links with Monte-Carlo), discuss β to control exploration and exploitation phases.

  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 $p(data) = 0$. 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.

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.

- Then more details: discuss hypotheses to compute the weights, choice of transformations, choice of β, choice of loss, prior, initial sampling, choice of representation for the polynomials.

  Let us now explain in more details how does this procedure goes. The goal is to reconstruct some distribution function $p(z)$, where here z in going to be some polynomial. **(ce: Notational conflict: $\Longleftarrow$ p for polynomials and p the distribution.)** We will try to reconstruct this density function by series of density function $\pi_n(z_n) = \gamma_n(z_n)/Z_n$ with $n = 1, \dots, N$ 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 $L(z)$:

$$\gamma_n(z) := p_0(z)\ \exp\left(-\beta_n L(z)\right). \tag{3.2}$$

  At each step, we have a set of particles $\{z_{n-1}^k, w_{n-1}^k\}$ 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 walculate 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 ($q(z_{n-1}^k|z_n^k)$). 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: It 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 his coefficient and modify it by a random noise from the guassian distribution $\mathcal{N}(0, \sigma^2)$. So for eg : $2x + 3y^2 \to 2.1x + 3y^2$.
- 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 : $2x + 3y^2 \to 2xy + 3y^2$.
- 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 : $2x + 3y^2 \to 2xy + 3y^2$. **(ce: Change the example.)** $\Longleftarrow$

This implementation allows us to straightforwardly calculate the forward and backward propagation kernel. For example, the case of the coefficient pertubation is symmetric. Therefore $q(z_n^k|z_{n-1}^k) = q(z_{n-1}^k|z_n^k)$, which allows to simplify the calculation of the incremental importance weight.

The acceptance ration for the MCMC algorithm is given by :

$$A(z_n, z_{n-1}) = \min\left(1, \frac{\gamma_n(z_n^k)q(z_{n-1}^k|z_n^k)}{\gamma_{n-1}(z_{n-1}^k)q(z_n^k|z_{n-1}^k)}\right) \tag{3.5}$$

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.

Before closing the section, we need to discuss what is the Loss function we choose. It is recquired 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 :

$$L(z) = \sum_i z(x_i)^2 + \frac{\lambda}{\sum_k c_k} \tag{3.6}$$

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 $\lambda \sim \mathcal{O}(1000)$.

- Discuss the fact that we allow float coefficients even though we know the coefficients will be only integers of square roots of integers?
- Analysis after AIS: select the best polynomials and do exploitation on the coefficients (without MC: we keep only the proposal of it betters the polynomial).

  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 coefficient 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.

- Results for naive choices of parameters (numbers of iteration and particles, probabilities, β) and motivate them (we want some exploration and then exploitation, not too long computations): for multiple runs we find multiple polynomials (good polynomial: after exploitation we convert the coefficients to integers and square roots, and recompute loss without regularisation, select with threshold). Quantify it nicely: success rates for each polynomials, and absolute number of success, failing rate. Total time needed, without cluster or fancy computers.

Results We have used the numerical analysis outlined in the previous sections to the search for extrema of the scalar potential (2.3). The code finds the 7 different following polynomials: **(ce: Mention that we "rounded" the coefficients to integer and square root of them.)** ⟸

$$p_1 = -\sqrt{2}\,x_1 + \sqrt{2}\,x_1\tilde{x}_8 + x_2\tilde{x}_8x_{10} - \sqrt{2}\,x_2x_4\tilde{x}_8, \tag{3.7a}$$
$$p_2 = 2\,x_2 - 2\,x_2\tilde{x}_8 + \sqrt{2}\,x_1x_{10} + 2\,x_1x_4 - x_2\tilde{x}_8x_{10}^2, \tag{3.7b}$$
$$p_3 = 2\,x_2 - 2\,x_2\tilde{x}_8 + \sqrt{2}\,x_1\tilde{x}_8x_{10} + 2\,x_1x_4\tilde{x}_8 - 2\,x_2x_4^2\tilde{x}_8, \tag{3.7c}$$
$$p_4 = \sqrt{2}\,x_2 - \sqrt{2}\,x_2\tilde{x}_8 + \sqrt{2}\,x_1\,x_4 + x_1\tilde{x}_8x_{10} - x_2x_4\tilde{x}_8x_{10}, \tag{3.7d}$$
$$p_5 = -2\,x_1 - 2\,x_2x_4 - 2\,x_1x_4^2 + 2\,x_1\tilde{x}_8 + \sqrt{2}\,x_2x_{10} + x_1\tilde{x}_8x_{10}^2, \tag{3.7e}$$
$$p_6 = -2\,x_2x_4 - 2\,x_1x_4^2 + 2\,x_2x_4\tilde{x}_8 + \sqrt{2}\,x_2x_{10} - \sqrt{2}\,x_2\tilde{x}_8x_{10} + x_1\tilde{x}_8x_{10}^2, \tag{3.7f}$$
$$p_7 = -\sqrt{2}\,x_1^2x_4 + \sqrt{2}\,x_2^2x_4 + \sqrt{2}\,x_1x_2x_4^2 - x_1^2x_{10} - x_2^2x_{10}, \tag{3.7g}$$

where $\tilde{x}_8 = e^{x_8}$. They are found with different frequencies, depending on the chosen parameters.

1. For 100 runs with parameters

$$\begin{gathered}\texttt{max degree} = 4, \quad \texttt{max num monomials} = 6, \quad n_{\text{iter}} = 1000, \quad n_{\text{particles}} = 1000,\\ \texttt{sample size} = 10000, \quad \text{reg} = 10^3, \quad \beta = 10^{-10} + \left(\frac{i}{n_{\text{iter}}}\right)^5, \quad \sigma = 0.5 - \frac{0.45}{1+\exp\left(10 - 20i/n_{\text{iter}}\right)},\\ p_{\text{add}} = 0, \quad p_{\text{remove}} = 0, \quad p_{\text{modify}} = 0.5, \quad p_{\text{multiply}} = 0.25, \quad p_{\text{divide}} = 0.25,\end{gathered} \tag{3.8}$$

the polynomials are found at the following frequencies

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$\varnothing$
Frequency	59%	25%	5%	9%	2%	4%	2%	13%

(3.9)

The sum of the percentages is higher than 100% because some runs find more than one polynomials. The code fails to find any polynomial in 13% of the time. The averaged time needed for a single run on a PC in $\sim 700\,$s. **(ce: This timing is with `analysis`, it may be less without.)** ⟸

Averaged number of particles reproducing a target in a successful run: 931 ± 230

Mean loss with regularisation for particles reproducing a target: 285 ± 108

Mean loss without regularisation for particles reproducing a target: 33 ± 24

2. For 100 runs with parameters

$$\begin{gathered}\texttt{max degree} = 4, \quad \texttt{max num monomials} = 6, \quad n_{\text{iter}} = 1000, \quad n_{\text{particles}} = 1000, \\ \texttt{sample size} = 10000, \quad \text{reg} = 10^3, \quad \beta = 10^{-10} + \left(\frac{i}{n_{\text{iter}}}\right)^5, \quad \sigma = 0.1, \\ p_{\text{add}} = 0, \quad p_{\text{remove}} = 0, \quad p_{\text{modify}} = 0.5, \quad p_{\text{multiply}} = 0.25, \quad p_{\text{divide}} = 0.25,\end{gathered} \tag{3.10}$$

the polynomials are found at the following frequencies

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$\varnothing$
Frequency	69%	11%	7%	12%	0%	4%	0%	13%

(3.11)

The dynamics of the appearance of the target polynomials is plotted in fig. 5.

Averaged number of particles reproducing a target in a successful run: 960 ± 175

Mean loss with regularisation for particles reproducing a target: 282 ± 91

Mean loss without regularisation for particles reproducing a target: 35 ± 24

3. For 100 runs with parameters

$$\begin{gathered}\texttt{max degree} = 4, \quad \texttt{max num monomials} = 6, \quad n_{\text{iter}} = 1000, \quad n_{\text{particles}} = 1000, \\ \texttt{sample size} = 10000, \quad \text{reg} = 10^3, \quad \beta = 10^{-10} + \left(\frac{i}{n_{\text{iter}}}\right)^3, \quad \sigma = 0.1, \\ p_{\text{add}} = 0, \quad p_{\text{remove}} = 0, \quad p_{\text{modify}} = 0.5, \quad p_{\text{multiply}} = 0.25, \quad p_{\text{divide}} = 0.25,\end{gathered} \tag{3.12}$$

the polynomials are found at the following frequencies

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$\varnothing$
Frequency	76%	18%	10%	6%	4%	1%	2%	6%

(3.13)

The dynamics of the appearance of the target polynomials is plotted in fig. 6.

Averaged number of particles reproducing a target in a successful run: 973 ± 103

Mean loss with regularisation for particles reproducing a target: 310 ± 115

Mean loss without regularisation for particles reproducing a target: 58 ± 40

4. For 100 runs with parameters

$$\begin{gathered}\texttt{max degree} = 4, \quad \texttt{max num monomials} = 6, \quad n_{\text{iter}} = 1000, \quad n_{\text{particles}} = 1000, \\ \texttt{sample size} = 1000, \quad \text{reg} = 10^3, \quad \beta = 10^{-10} + \left(\frac{i}{n_{\text{iter}}}\right)^5, \quad \sigma = 0.1, \\ p_{\text{add}} = 0, \quad p_{\text{remove}} = 0, \quad p_{\text{modify}} = 0.5, \quad p_{\text{multiply}} = 0.25, \quad p_{\text{divide}} = 0.25,\end{gathered} \tag{3.14}$$

the polynomials are found at the following frequencies

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$\varnothing$
Frequency	72%	27%	3%	19%	2%	0%	0%	12%

(3.15)

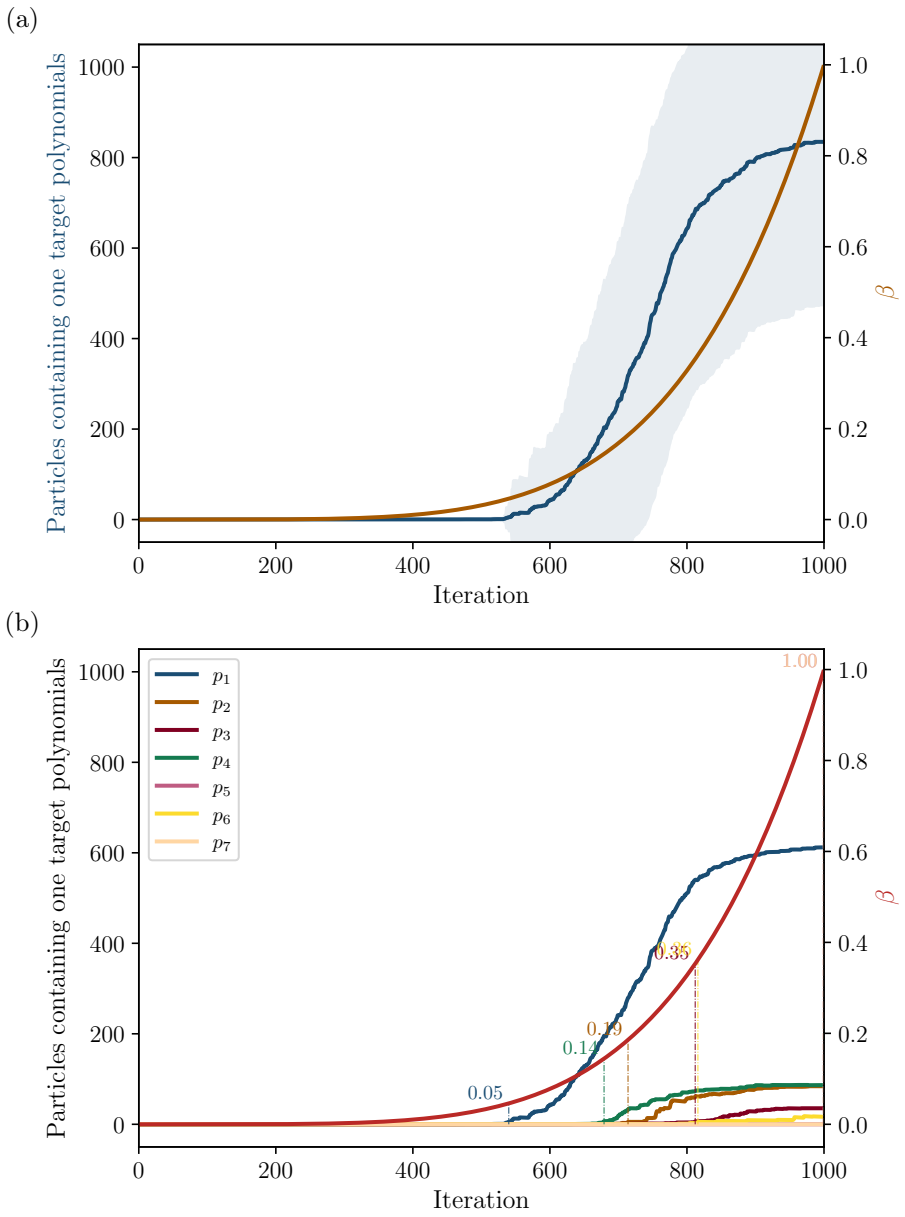

Fig. 5 (a) Evolution of the mean numbers of particles reproducing a target polynomials and its 1σ deviation for the parameters (3.8). (b) Evolution of the mean numbers of particles reproducing each targets polynomial. The vertical lines indicate when there are an average 5 particles reproducing a given polynomial, and the value of β at that moment.

The dynamics of the appearance of the target polynomials is plotted in fig. 7.

Averaged number of particles reproducing a target in a successful run: 401 ± 389

Mean loss with regularisation for particles reproducing a target: 406 ± 193 **(ce: Needs to be renormalized by the sample size.)** ⟸

Mean loss without regularisation for particles reproducing a target: 272 ± 192 **(ce: Needs to be renormalized by the sample size.)** ⟸

⟸

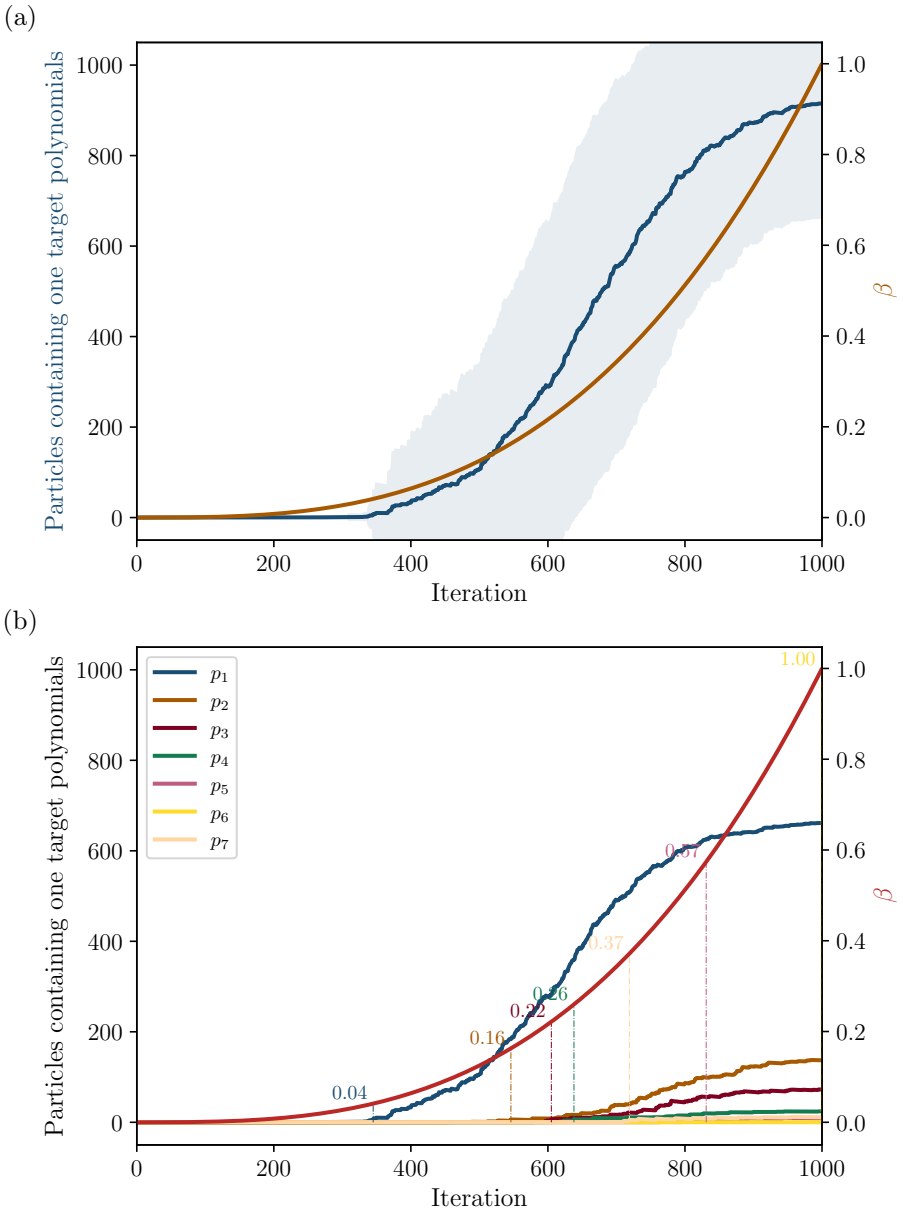

Fig. 6 Evolution of the mean numbers of particles reproducing a target polynomials and its 1σ deviation for the parameters (3.12). (b) Evolution of the mean numbers of particles reproducing each targets polynomial. The vertical lines indicate when there are an average 5 particles reproducing a given polynomial, and the value of β at that moment.

(ce: Draw histograms of the polynomials distribution to discuss runs with multiple outputs?)

4 Supergravity solutions

- For each couple of candidate polynomials, we get the same expression for the solution, and it indeed defines a unique vacuum of the 3d SUGRA.

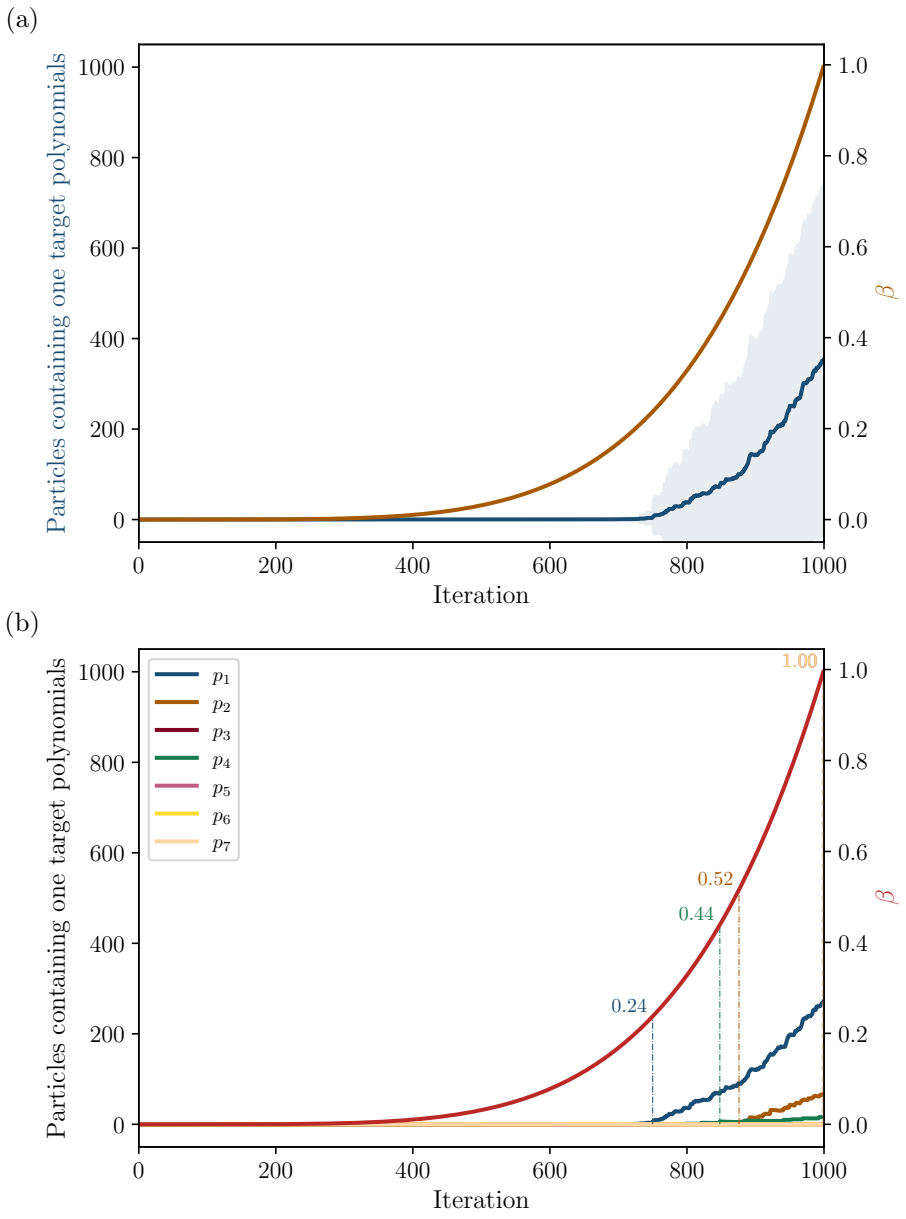

Fig. 7 Evolution of the mean numbers of particles reproducing a target polynomials and its 1σ deviation for the parameters (3.14). (b) Evolution of the mean numbers of particles reproducing each targets polynomial. The vertical lines indicate when there are an average 5 particles reproducing a given polynomial, and the value of β at that moment.

- Discussion of the vacuum: gauge group, Zham. metric, change of variables, 3d spectrum (and stability), spin 2 spectrum on S^3, one parameter seems to be a gauge parameter, uplift?

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. 3.7. We can now solve the system

$$p_i = 0, \quad \forall i \in 1, \dots, 7 \tag{4.1}$$

which leads to the following expressions:

$$\begin{cases} x_1 = \dfrac{x_2}{\sqrt{2}} \dfrac{e^{x_8/2}}{e^{x_8}-1} \Big(- x_5\, e^{x_8/2} + \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)}\Big), \\ x_4 = \dfrac{e^{-x_8/2}}{\sqrt{2}} \sqrt{2 - 4\, e^{x_8} + e^{2x_8}\big(2 + x_{10}^2\big)}. \end{cases} \tag{4.2}$$

Alternatively, the system can be recast as:

$$\begin{cases} \tilde{x}8 = \dfrac{x1^2 + x_2^2}{x_2^2 + \big(x_1 - x_2 x_4\big)^2}, \\ x_{10} = \sqrt{2}, x_4, \dfrac{x_2^2 - x_1^2 + x_1 x_2 x_4}{x_1^2 + x_2^2}. \end{cases} \tag{4.3}$$

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. 4.1 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.4.3—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.4.3.

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

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

$$x_1 = r\cos(\theta)\cos(\Phi), \quad x_2 = r\cos(\theta)\sin(\Phi) \quad \text{and} \quad x_4 = r\sin(\theta), \tag{4.4}$$

for which the Zamolodchikov metric reads

$$\mathrm{d}^2 s_{\text{Zam.}} = -\,\mathrm{d}r^2 - r^2 \left(\mathrm{d}\theta^2 - r\cos(\theta)\,\mathrm{d}\theta\mathrm{d}\Phi + \sin(\theta)\,\mathrm{d}r\mathrm{d}\Phi + \frac{1}{2}\big(3 + r^2 - \cos(2\theta)\big)\,\mathrm{d}\Phi^2 \right). \tag{4.5}$$

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

Bosonic spectrum Vector fields:

$$\begin{aligned} m_{(1)}\ell_{\text{AdS}}: \quad & 0\ [2], \quad -2\ [5], \quad 2\ [1], \\ & -1 \pm \sqrt{(1+r^2)^2 - 2\,r^2\,\cos(2\theta)}\ [2+2], \\ & 1 \pm \sqrt{1 + 4\,r^2 + r^4}\ [2+2]. \end{aligned} \tag{4.6}$$

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:

$$\begin{aligned}(m_{(0)}\ell_{\rm AdS})^2: \quad & 0\ [5], \quad 8\ [1], \quad r^2\ \left(4+r^2\right)\ [8], \\ & 2r\left(3\,r+r^3\pm(2+r^2)\,\sqrt{2+r^2-2\,\cos(2\theta)}-r\cos(2\theta)\right)\ [2+2].\end{aligned} \tag{4.7}$$

Gravitini:

$$m_{(3/2)}\ell_{\rm AdS}: \qquad \frac{1}{2}\left[1\pm\sqrt{4+2\,r^2+r^4-2r^2\cos(2\theta)}\right]\ [4+4], \tag{4.8}$$

no SUSY enhancements other than $r=0$.

No dependence on Φ.

5 Conclusion

- Conclusion: good prospects of improvement to be able to access higher dimensional cases. Classify flat directions.
- Appendix with some details on the code?