Optimization on Symplectic Embeddings

Alexander Gajewski, Eli Goldin, Jakwanul Safin,
Navtej Singh, Junhui Zhang

July 2019

1 Introduction

In this paper we develop new constrained optimization techniques for learning *symplectic* maps, like the maps that result from Hamiltonian flows in mechanical systems. We apply these techniques to the problem of finding symplectic embeddings of certain symplectic domains into the ball, a problem which has seen exciting theoretical developments recently, but about which little remains known. We show that our techniques can learn embeddings competitive with those constructed by hand in past theoretical works, and confirm a conjecture about embeddings of 8- and 10-dimensional simplices using novel simplices discovered by our algorithm.

1.1 Background

Symplectic maps originate from the Hamiltonian formulation of classical mechanics. In Hamiltonian mechanics, the configuration of a mechanical system is determined by a point in *phase space* $\mathcal{M}$, the space of all possible positions and momenta for the objects in the system.[1] Every dimension of position corresponds to a dimension of momentum, so phase space is always even-dimensional. For the purposes of this paper, phase space can be considered as any subdomain of $\mathbb{R}^{2n}$, with coordinates written as $(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ where each (x_i, y_i) pair corresponds to the position and momentum in the ith coordinate. $\mathbb{R}^{2n}$ can also be considered as $\mathbb{C}^n$ under the identification of $z_\ell = x_\ell + iy_\ell$. A path through phase space corresponds to a "movie" of the system playing out over time. Paths through phase space are governed by a system of differential equations, which in turn are determined by the *Hamiltonian* of the system, a real-valued function $H : \mathcal{M} \to \mathbb{R}$ that measures the total energy of each configuration. The Hamiltonian differential equations are then given by:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = J\nabla H \tag{1}$$

[1]This space is made mathematically rigorous as a *manifold*, which intuitively is a space that locally can be mapped to and from a subset of Euclidean space in a smooth way. For our purposes, it suffices to think of this as a closed subset of $\mathbb{R}^{2n}$ with nonempty interior, and whose boundary is "sufficiently smooth."

where x and y are vectors in $\mathbb{R}^n$ representing position and momentum, respectively, and J is the *canonical symplectic matrix*,

$$J = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}$$

When viewing phase space as $\mathbb{C}^n$, the canonical symplectic matrix is equivalent to rotating each coordinate by i. Recall that equation 1 determines the trajectory of the mechanical system given an initial configuration. In one sense, these equations yield a trajectory, which is a function $\tau_H^z(t) : [0, \infty) \to \mathcal{M}$ and outputs the configuration of the system after time t, starting at configuration $z = (x, y) \in \mathcal{M}$. However, one can also view a trajectory as a function of the initial configuration, fixing the time t that the "movie" will play for. In this interpretation, the Hamiltonian differential equations give a function $\Phi_H^t(z) : \mathcal{M} \to \mathcal{M}$, which is known as the time-t flow of the system. We often work with the time-1 flow, in which case we drop the t and write simply $\Phi_H(z)$.

One of the most important results of classical physics is that Hamiltonian flows preserve the canonical symplectic matrix. That is, if one considers the Jacobian $D(\Phi_H^t)$ of the Hamiltonian flow, then $D(\Phi_H^t)JD(\Phi_H^t)^T = J$. Any map satisfying this condition is called *symplectic*, and symplectic maps are central to the field of *symplectic geometry*. This is a remarkably strong condition, and it immediately follows that symplectic maps preserve volume, meaning that the volume of some domain is the same as the volume of its image under the symplectic map.[2]

In this paper, we concern ourselves with *symplectic embeddings*, i.e. a smooth embedding satisfying the symplecticity condition. Until recently, mathematicians and physicists believed that a symplectic embedding exists from any space to any other space of equal volume. When Mikhail Gromov proved his non-squeezing theorem in 1985, mathematicians realized that the community had a rather poor understanding of symplectic embeddings.[3] Even today, mathematicians know very little about when symplectic embeddings may or may not exist. A thorough understanding of when symplectic embeddings exist should help illuminate the nature of Hamiltonian mechanics itself.

We will focus on constructing symplectic embeddings, $\Phi : M \hookrightarrow N$, where M, N are convex subdomains of $\mathbb{R}^{2n}$. If such an embedding exists we say that M symplectically embeds into N, or $M \hookrightarrow N$.

1.2 Definitions

Definition 1.

$$\textit{disk: } D(a) = \left\{ z \in \mathbb{C} \mid \pi \|z\|^2 \leq a \right\}$$

[2]This result is known as Liouville's theorem, and is one of the earliest known results about Hamiltonian dynamics.

[3]Gromov proved that a symplectic ball cannot symplectically embed into an infinite cylinder with a smaller radius despite the cylinder having infinite volume. The formal statement says that $B(a')$ embeds into $Z(a)$ if and only if $a' \leq a$. Definitions in §1.2.

$$\textit{Ellipsoid: } E^{2n}(a_1, a_2, \ldots, a_n) = \{z \in \mathbb{C}^n \mid \sum_{i=1}^{n} \frac{\pi \|z_i\|^2}{a_i} \leq 1\}$$

$$\textit{Ball: } B^{2n}(a) = \big\{z \in \mathbb{C}^n \mid \pi \|z\|^2 \leq a\big\} = E^{2n}(\underbrace{a, a, \ldots, a}_{n})$$

$$\textit{Cylinder: } Z^{2n}(a) = \big\{z \in \mathbb{C}^n \mid \pi \|z_1\|^2 \leq a\big\} = D(a) \times \mathbb{C}^{n-1}$$

$$\textit{Polydisk: } P^{2n}(a_1, a_2, \ldots, a_n) = \big\{z \in \mathbb{C}^n \mid \pi \|z_i\|^2 \leq a_i, i = 1, 2, ..., n\big\} = D(a_1) \times \cdots \times D(a_n)$$

2 Methodology

2.1 Integrating Hamiltonians (symplectically)

In this work, we develop techniques for optimizing symplectic maps computationally. In order to do this, is it is necessary to first develop some parameterization of symplectic maps. While we experimented with several techniques (see sections A.2, **??**, A.3 for more details), our most successful technique consists of parameterizing time-dependent Hamiltonians as sequences of neural networks, and numerically integrating the resulting Hamiltonian differential equations with a symplectic integrator.

2.1.1 What is symplectic integration?

Recall that symplectic maps are a generalization of time-1 flows of arbitrary Hamiltonians (i.e. functions from $\mathbb{R}^{2n} \to \mathbb{R}$). In fact, every symplectomorphism can be approximated by the flow of some Hamiltonian. Thus, one way to parameterize symplectomorphisms is to represent Hamiltonians H with function approximators like neural networks or polynomials, and then numerically integrate the resulting flow with e.g. Euler's method:

$$\begin{aligned} x_{n+1} &= x_n + \nabla_y H(x_n, y_n)\delta \\ y_{n+1} &= y_n - \nabla_x H(x_n, y_n)\delta \end{aligned}$$

where x_n and y_n are vectors representing the position after the nth step of integration, and δ is the amount of time that passes during each step of integration. Although Hamiltonian flows are indeed symplectic, numerical integration can introduce a significant amount of error, making the resulting approximation *not* symplectic. Because we want to learn symplectic embeddings, it is important that the resulting maps be as close to symplectic as possible, and the error introduced by Euler's method is high enough to result in embeddings that are provably impossible, making this technique unacceptable for our purposes (see appendix A.3 for examples of such pathological cases).

One solution to this issue is symplectic integration. Symplectic integrators are rules for numerical integration, like Euler's method, but that guarantee that the resulting approximate flow is symplectic. Leapfrog integration is one such technique, and works for split Hamiltonians, i.e. Hamiltonians of the form

$H(x,y) = V(x) + T(y)$. Each step of leapfrog is defined by the following functions:

$$\begin{aligned} x_{n+1/2} &= x_n + \nabla_y T(y_n)\frac{\delta}{2} \\ y_{n+1} &= y_n + \nabla_x V(x_{n+1/2})\delta \\ x_{n+1} &= x_{n+1/2} + \nabla_y T(y_{n+1})\frac{\delta}{2} \end{aligned}$$

One step of leapfrog integration gives an approximation for the time-δ flow of the Hamiltonian, and so to approximate the time-1 flow of a Hamiltonian one can divide $[0,1]$ into chunks and repeatedly perform iterations of leapfrog with appropriately sized δs. Importantly, the resulting map is guaranteed to be symplectic regardless of the accuracy of the approximation. As our goal is to *learn* symplectomorphisms, and not to accurately compute time-1 flows of specific Hamiltonians, even a step size of $\delta = 1$ suffices.

2.1.2 Why/what neural networks?

Recall that with our method, we use leapfrog integration to symplectically integrate learned Hamiltonians. Thus, we need a way of parameterizing Hamiltonians, i.e. real-valued differentiable functions on $\mathbb{R}^{2n}$. Two such parameterizations are polynomials and neural networks. With polynomials, we represent split Hamiltonians as follows:

$$\begin{aligned} H(x,y) &= V(x) + T(y) \\ V(x) &= \theta_1^1 x_1 + \theta_2^1 x_2 + \cdots + \theta_n^1 x_n + \\ &\quad \theta^2_{(1,1)} x_1 x_1 + \theta^2_{(1,2)} x_1 x_2 + \cdots + \theta^2_{(1,n)} x_1 x_n + \cdots + \theta^2_{(n,n)} x_n x_n + \\ &\quad \vdots \\ &\quad \theta^d_{(1,1,\dots,1)} x_1^d + \theta^d_{(1,1,\dots,2)} x_1^{d-1} x_2 + \cdots + \theta^d_{(n,n,\dots,n)} x_n^d \end{aligned}$$

and likewise $T(y)$ as a polynomial in the variables $y_1, \dots, y_n$. Here, θ represents the parameters of the model, to be optimized during training. By increasing the degree d of the polynomial representation, it is possible to approximate any continuous function to arbitrary precision. However, the degree may need to be very high, and the number of parameters is $\Theta(n^d)$, exponential in d, making polynomials inefficient when $n > 1$.

Neural networks are alternating compositions of arbitrary affine and elementwise nonlinear maps. Like polynomials, they can theoretically approximate any continuous function to arbitrary precision. However, empirically neural networks have been found to efficiently approximate many functions of interest, and importantly to be efficiently trainable. With neural networks, we represent split Hamiltonians as follows:

$$\begin{aligned} H(x,y) &= V(x) + T(y) \\ V(x) &= \sigma(b_d + W_d \sigma(b_{d-1} + W_{d-1}\sigma(\cdots b_1 + W_1 x))) \end{aligned}$$

and likewise $T(y)$, where the W_k are matrices, the b_k are bias vectors, and σ is an elementwise nonlinear function. Because H is meant to be scalar-valued, the matrix W_d has only a single row, and the vector b_d only a single entry. The $W_1, \ldots, W_d$ and $b_1, \ldots, b_d$ together form the parameters θ to be optimized during training.

We were able to achieve results under both models, but in dimensions greater than 2, higher degree polynomials took significantly longer to compute than neural networks of similar expressivity. For this reason, we decided to perform our experiments primarily with neural networks.

2.1.3 Universal approximation by Leapfrog integration (Henon-like maps)

In this section, we outline a proof that iterated Leapfrog integration of time-dependent neural network Hamiltonians can approximate any symplectomorphism. At first, this may seem obvious, because any symplectomorphism of $\mathbb{R}^{2n}$ can be approximated by the time-1 flow of some Hamiltonian, and any Hamiltonian can be approximated by some sufficiently large neural network. Thus, the argument would continue, sufficiently many steps of numerical integration of these neural network Hamiltonians should be able to approximate any symplectomorphism.

However, recall that leapfrog integration only works for *split* Hamiltonians, i.e. Hamiltonians of the form $H(x, y) = V(x) + T(y)$, and a priori it is unclear whether all Hamiltonian flows can be approximated by split Hamiltonian flows. Fortunately, prior work Turaev [2003] has shown that, in fact, any symplectomorphism of $\mathbb{R}^{2n}$ can be approximated by the time-1 flow of some split Hamiltonian.

2.2 Learning with gradient descent

Armed with theoretical results that our models can approximate all symplectomorphisms, we can turn our attention to optimization. Our main optimization technique is stochastic gradient descent, chosen for its ability to quickly and efficiently optimize deep neural networks. More precisely, we have some source domain, say the 4-dimensional ellipsoid $E(1, 5)$, and we would like to symplectically embed this domain into the smallest possible ball $B^4(c)$.

We represent our map $F_\theta : \mathbb{R}^4 \to \mathbb{R}^4$ as a sequence of layers, each layer performing one step of leapfrog integration on a neural network Hamiltonian:

$$F_\theta(x, y) = \mathcal{L}(H_{\theta_k}) \circ \cdots \circ \mathcal{L}(H_{\theta_1})(x, y) \tag{2}$$

where each $H_{\theta_i} : \mathbb{R}^4 \to \mathbb{R}$ is a neural network Hamiltonian with parameters θ_i, and $\mathcal{L}$ applies one step of leapfrog integration under the Hamiltonian in its argument.

Learning is an iterative process. At each step, points (x_i, y_i) are sampled from the boundary of the source domain, here $E(1, 5)$, and mapped to $(\tilde{x}_i, \tilde{y}_i) =$

$F_\theta(x_i, y_i)$. We then compute our loss function, which in this case we take to be the area of the smallest ball containing each of the $(\tilde{x}_i, \tilde{y}_i)$:

$$Q(\theta) = 2\pi \max_i \|(\tilde{x}_i, \tilde{y}_i)\|^2 \tag{3}$$

Because each layer of F_θ is differentiable with respect to θ, we can compute the gradient ∇Q, and apply a step of gradient descent towards minimizing Q. Over many iterations, the parameters θ will improve, resulting in embeddings into smaller and smaller balls. See sections **??** and **??** for details on how we sample from different domains, and for examples of different loss functions.

2.3 Pathological embeddings and resampling

One remaining fear is that because our algorithm works with only finitely many points, it may be that these points do indeed map to a certain target domain, but that other points that were not sampled map outside the target domain. This problem is exacerbated if there are very few sample points, and if the sample points remain fixed for all iterations of training. Indeed, in this case the embeddings learn at first, but after several iterations, they start overfitting to the specific sample points chosen at the beginning of learning. See appendix section A.4 for more details on these experiments. To compensate, we *resample* training points at every iteration of learning. Indeed, with resampling, even as few as 100 sample points suffice with minimal sampling error, giving similar enclosing areas to verifications with upwards of $10^4 - 10^6$ sample points.

2.4 Φ_0 and folding

While theoretically true that any symplectomorphism can be approximated by a sequence of Leapfrog-integrated Hamiltonians, many existing constructions of symplectic embeddngs use a completely different technique. *Symplectic folding* is a family of symplectic embeddings that works particularly well for toric domains like ellipsoids and polydisks. Folding consists of a sequence of symplectic transformations, each given either as an explicit map or as a flow generated by some Hamiltonian. The complete folding construction is fairly complex, so we will not describe all of the steps (see Schlenk [1999] for details).

One of the key aspects of folding is that it takes place in a transformed coordinate space. That is, the first step of folding is a symplectic coordinate transformation, which we call Φ_0, and the last step of folding undoes this coordinate transformation via Φ_0^{-1} (see section A.5 for details on the construction of Φ_0). Because of the initially poor performance of the leapfrog approach on the polydisk problem, we decided to try learning directly in this transformed space, as nontrivial embeddings might be simpler to describe there.

There are two ways we tried to learn in this transformed space. One was to theoretically apply Φ_0 to our source domain, say the polydisk $P(1, 5)$, and compute its image under the map, in this case the domain $\tilde{E}(1, 5)$. Then during training, instead of sampling points (x_i, y_i) from the boundary of $E(1, 5)$, we sample from the boundary of $\tilde{E}(1,5)$. Correspondingly, as a loss function, instead of computing the area of the smallest ball containing the $(\tilde{x}_i, \tilde{y}_i)$, we compute the area of the smallest transformed ball $\tilde{B}^4(c)$ containing them, where $\tilde{B}^4(c)$ is the image of the ball under Φ_0. Unfortunately, this method did not manage to give results better than the identity, so we did not pursue it further. However, it might be possible to do better with more hyperparameter tuning.

The second way was inspired by standard, unconstrained neural networks, which are alternating compositions of linear and elementwise nonlinear maps. Making an analogy with neural networks, we represent symplectic maps as alternating compositions of *symplectic* linear and *symplectic* nonlinear maps. As symplectic nonlinear maps, we alternate between Φ_0 and Φ_0^{-1}. Thus, our map is given by

$$F_\theta(x,y) = \Phi_0^{-1} \circ A_{\theta_k} \circ \Phi_0 \circ \cdots \circ A_{\theta_2} \circ \Phi_0^{-1} \circ A_{\theta_1} \circ \Phi_0(x,y) \tag{4}$$

where each A_{θ_i} is a linear symplectic map with parameters θ_i. We also experimented with other nonlinearities, described in section A.2. We derived and implemented a differentiable approximation of Φ_0, and parameterized linear symplectic maps with G-reflectors (see sections A.5 and **??** for details). Unfortunately, due to numerical instability of the Φ_0 map, this method often failed to converge, and we were unable to learn embeddings on any complex problems. We hope that future work may further explore this direction of research.

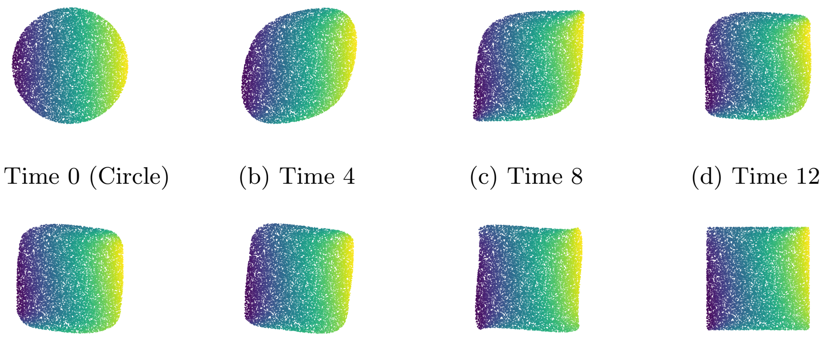
3 Experiments

3.1 2 dimensional experiments

In two dimensions, any two domains with the same volume are symplectomorphic. This makes two dimensional embedding problems a good testing ground for learning symplectomorphisms. We experimented with a pair of two dimensional embedding problems: that of embedding a ball into the smallest possible square, and that of embedding four side-by-side balls into the smallest single ball.

3.1.1 Ball into square

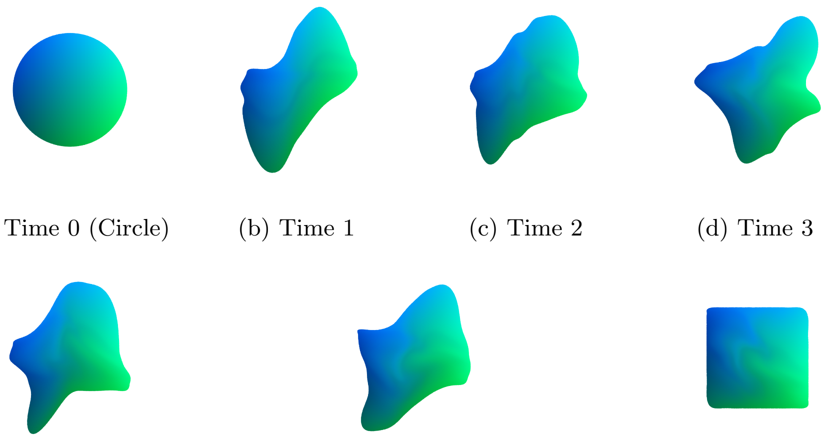
A simple example to understand our algorithms is embedding a ball into a square. As polynomials are sufficiently efficient to compute in the 2-dimensional case, we applied our algorithm to this problem using both polynomial and neural network Hamiltonians. Figure 1 shows the flow that our algorithm gives when using degree 10 polynomial Hamiltonians with 10 macro-steps and 10 micro-steps. It embeds the circle of area π into the square of area 3.1451, an error of about 0.004. Figure 2 shows the flow that the same algorithm gives when using neural networks. It embeds the circle of area π into the square of area 3.1677.



(a) Time 0 (Circle) (b) Time 4 (c) Time 8 (d) Time 12

(e) Time 18 (f) Time 22 (g) Time 26 (h) Time 30

Figure 1: Eight frames from the flow of a circle into a square under our learned polynomial map.



(a) Time 0 (Circle) (b) Time 1 (c) Time 2 (d) Time 3

(e) Time 4 (f) Time 5 (g) Time 6

Figure 2: Seven frames from the flow of a circle into a square under our learned polynomial map.

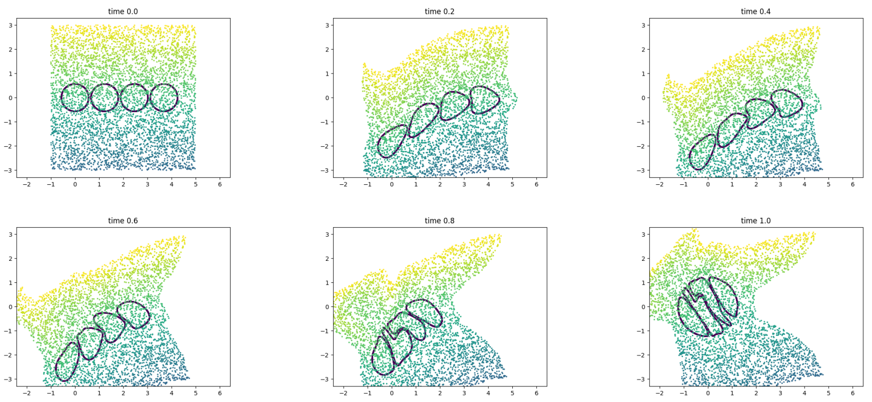


Figure 3: four balls into one

3.1.2 Four balls into one

We take four balls each of area 1 and try embedding them into one ball of the smallest area(radius) possible. We were not able to get arbitrarily close to the known optimal embedding: since these are 2 dimensional balls, we should be able to pack four balls of area 1 into a large ball of area 4. In the experiment, we use split neural network of width 200 with 2 hidden layer, 10 macro-steps and 2 micro-steps. Based on the initial centers of the four balls, after 15000 iterations, the algorithm can embed 4 balls into a larger ball of area 4.7889 (initial centers of balls on x axis, closest distance between 2 nearby balls is 0.1), and 5.24 (initial centers of balls are around a circle on the $x-y$ plane, closest distance between 2 nearby ball is 0.1). Figure 3 shows the flow of the Hamiltonian for four balls initially centered on the x axis (applied to both the four balls as well as random sample points in $[-1,5] \times [-3,3]$).

We posit some reasons for this failure. One issue rests on the fact that this problem is somewhat sensitive to the initial configuration of balls; that is, their relative positions leave more or fewer gaps of varying size with some easier to fill with symplectic transformations than others. Another issue is that we are using the same Hamiltonian for all four balls. However, it might be the case that four ball packing requires different Hamiltonians for different balls, and it might be hard to use the flow generated by just one Hamiltonian to approximate the flow generated by four different Hamiltonians.

3.2 Fibonacci Staircase

As a more complex test of our algorithm, we apply our method to the well-studied problem of embedding a 4-dimensional ellipsoid into the smallest possible ball. Because of scaling symmetry, this reduces to the problem of embedding an ellipsoid $E(1,a)$ into a ball $B^4(c)$. Thus the optimal c can be seen as a function of a single variable, a. In McDuff and Schlenk [2009], Mcduff and Schlenk compute this function exactly, building on prior work characterizing ellipsoid embeddings in 4 dimensions. This function shows a surprising amount of structure.

For a between 1 and τ^4, the function is piecewise linear with infinitely many endpoints at ratios of certain Fibonacci numbers, where τ is the golden ratio. For this reason, the function is often called the *Fibonacci staircase*. Between τ^4 and 7, $c(a) = \frac{a+1}{3}$. Between 7 and $8 + \frac{1}{36}$, $c(a) = \sqrt{a}$ except on 8 disjoint intervals, in which the function is piecewise linear. Past $8+\frac{1}{36}$, volume is the only constraint, giving $c = \sqrt{a}$. Between τ^4 and $8+\frac{1}{36}$, volume is the only constraint, giving $c = \sqrt{a}$, except for seven intervals where the function is piecewise linear. Past $8 + \frac{1}{36}$, volume is the only constraint everywhere, so $c = \sqrt{a}$.

We ran our algorithm on $E(1, a)$ with 30 evenly spaced values of a between 1 and 10. After training, we sampled 10^6 points from each ellipsoid and computed the area of the smallest ball containing the image of these points under the algorithms's learned embedding. On some parts of the staircase, our algorithm is very close to optimal. Between $a = 4$ and $a = 5.25$, there is an error of only about 0.1. However, between 5.25 and 6.25, the error continues to worsen even though the optimal embedding remains the same. We posit that the $E(1, 6.25)$ embedding is much harder than the $E(1, 5.25)$ embedding; the 6.25 ellipsoid has much larger volume, but theoretically it can be embedded into the same ball as the 5.25 ellipsoid. It should also be noted that all hyperparameters were tuned on the $E(1, 5)$ embedding problem, so better results might be possible given additional hyperparameter searching. Interestingly, our algorithm failed to produce embeddings better than the identity map between $a = 2$ and $a = 3.1$, even though better embeddings are theoretically possible. We hypothesize that these problems are hard because these ellipsoids cannot be compressed very much, only about 30% by area.

3.3 6D Ellipsoids $E(1, a, b) \hookrightarrow B^4(c)$

3.3.1 Volume and EH constraints

While the answer to the question of when $E(1, a) \hookrightarrow B^4(c)$ is entirely known, the same cannot be said of the 6-dimensional case $E(1, a, b) \hookrightarrow B^6(c)$. In fact, there are there are very few known obstructions to such embeddings, and even fewer known constructions.

3.3.2 Buse-Hind embeddings

One construction for $E(1, a, b) \hookrightarrow B^6(c)$ was discovered by Buse and Hind in a 2018 paper. By proposition 3.4 of that paper:

Proposition 1. *If $E(a, b) \hookrightarrow E(c, d)$, then $E(a, b, e_1, \ldots, e_n) \hookrightarrow E(c, d, e_1, \ldots, e_n)$.*

In the 6-dimensional case, this means that if there exists a 4-dimensional embedding $E(1, a) \hookrightarrow E(1, c)$, then there likewise exists a 6-dimensional embedding $E(1, a, b) \hookrightarrow E(1, c, b)$. To simplify this problem even further we decided to look more specifically at embeddings of $E(1, a, b) \hookrightarrow B^4(c)$. By this

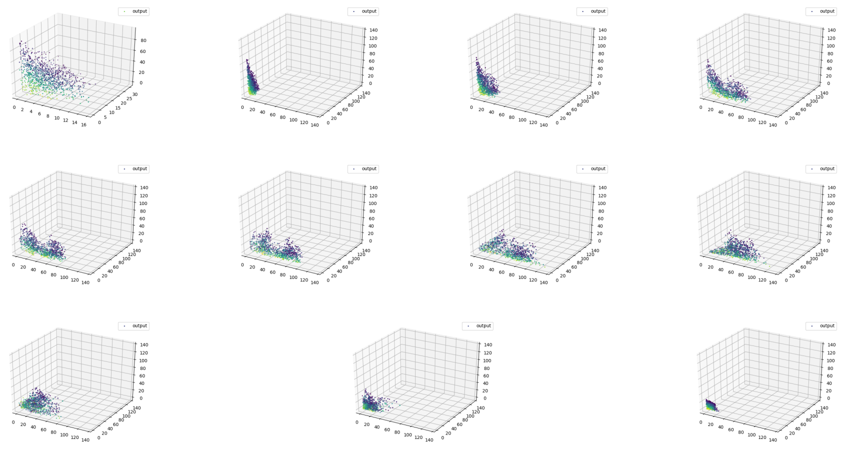


Figure 4: The moment map of the flow of $E(1, 27, 64) \hookrightarrow B^6(27.69)$ under our learned neural network Hamiltonians.

construction, whenever there is an embedding $E(1, a) \hookrightarrow B^4(c)$ there exists an embedding $E(1, a, b) \hookrightarrow B^6(\max(c, b))$ as $E(1, c, b) \subseteq B^6(\max(c, b))$. As the optimal embeddings for 4-dimensional ellipsoid into ball are fully known McDuff and Schlenk [2009], this construction can give a good upper bound on the minimal c for this question. However, this embedding is not known to be optimal, and so our goal for this problem was to find an embedding in any case that embeds into a smaller ball than this construction.

3.3.3 Challenges

The first issue encountered while attempting to apply this algorithm to the $E(1, a, b) \hookrightarrow B^6(c)$ problem was that of sampling error. Due to the curse of dimensionality, sampling points from a 6-dimensional ellipsoid gives significantly sparser results than sampling from a 4-dimensional ellipsoid. To alleviate this issue, we sampled from $20,000$ points for these experiments. However, this caused each experiment to take over 5 hours to run, and so we only were able to run our algorithm on a few sample ellipsoids. Nevertheless, these few samples were able to provide some promising results, and with more runs we hope that even more interesting embeddings could be found.

3.3.4 Our embeddings

Our techniques managed to embed the $E(1, 27, 64)$ ellipsoid into $B^4(27.69)$ (see fig. 4). For comparison, the Buse-Hind embedding in this case gives an embedding $E(1, 27, 64) \hookrightarrow B^6(27)$, while the volume constraint gives $c \geq 12$ for $E(1, 27, 64) \hookrightarrow B^6(c)$. Notably, our algorithm achieved an embedding within 3% of the best known embedding for this problem of $E(1, 27, 64) \hookrightarrow B^6(c)$. It is possible that in fact the Buse-Hind embedding is optimal, but it is certainly not guaranteed. However, this finding suggests that if there is a better embedding, it is in some real sense difficult to find. This also provides evidence that there may not be any such embedding at all. Meanwhile, in the case of the $E(1,5,50)$ ellipsoid, our techniques only managed to embed this into $B^6(14.1)$. In comparison, the Buse-Hind construction gives an embedding $E(1,5,50) \hookrightarrow B^6(7.1)$. Perhaps this is simply because our algorithm was not given enough time to learn, or perhaps skinnier ellipsoids simply take more time to learn.

3.4 4D Ball Packing

3.4.1 Background

Ball packing is another interesting problem that we try our algorithm. Recall that the Euclidean ball packing problem is to find compositions of translations (and rotations) of balls in order to fill as much as possible the volume of another shape. In a similar way, symplectic ball packing problem is to find symplectic embeddings from the disjoint union of k B^{2n} into another shape such as B^{2n} and C^{2n}. To test our algorithm, we mainly study the problem $\sqcup_{i=1}^{k} B^4(1) \hookrightarrow B^4(c)$ by minimizing the area $c = \max_{x,y} \pi(x_1^2 + y_1^2 + x_2^2 + y_2^2)$ of the ball we can embed k disjoint $B^4(1)$ into.

Like the ellipsoid case, in addition to the volume constraint that $c \geq \sqrt{k}$, there're other obstructions in the ball packing problem. One example is the following theorem from Gromov [1985]

Theorem 2 (Two Ball Theorem). *If $B^{2n}(a) \sqcup B^{2n}(a) \hookrightarrow B^{2n}(A)$, then $2a < A$.*

In fact, we can define the k's symplectic ball packing number (see Schlenk [2004] for details) p_k as

$$p_k = \sup_a \frac{k\ Vol(B^{2n}(a))}{B^{2n}(1)}$$

which means the maximum percent of volume of $B^{2n}(1)$ we can fill by symplectically embedding the disjoint union of k B^{2n} of equal area. It has been proven that in dimension 4, for $k \geq 9$, volume is the only constraint ($p_k = 1$), but for $1 < k < 9$, we can only fully fill $B^4(1)$ by 4 $B^4(\frac{1}{2})$. For other values of k, $p_k < 1$. See Table 1 for the exact values of these packing number. As a result, the optimal loss for our test should be $c = \sqrt{\frac{k}{p_k}}$.

Ball packing problem is also closely related to the "elliposid into ball" problem by the following theorem (see McDuff [2008] for details),

Theorem 3. *$E(1,k)$ symplectically embeds in the open ball $\mathring{B}^4(\mu)$ if and only if k disjoint balls $B^4(1)$ embed in $\mathring{B}^4(\mu)$.*

This means that the optimal embedding of $E(1,k), k \in \mathbb{N}$ into ball should give us $B^4(\sqrt{\frac{k}{p_k}})$. Indeed, this agrees with the Fibonacci Staircase.

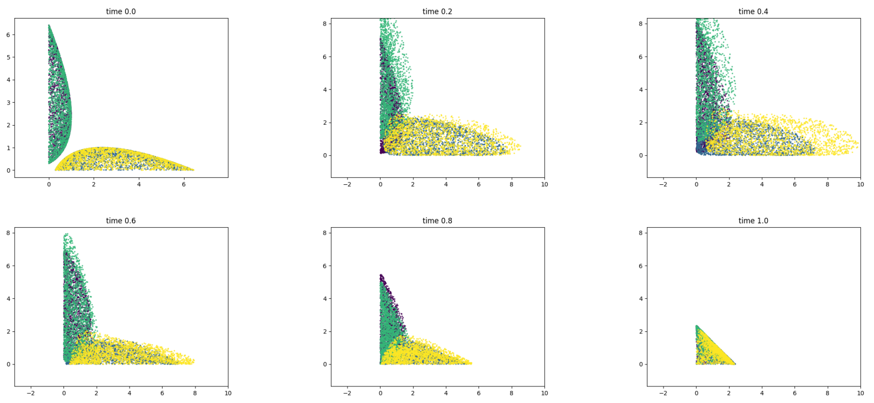


Figure 5: ball packing in 4 dimension, k=4

3.4.2 Experiment

Explicit realizations of the packing number use the moment map (see Schlenk [2004] and Traynor [1995] for details). Recall that the image of a B^4 under the moment map is $\triangle \times \square$, so instead of packing the ball directly, they work in the moment space and find the affine linear symplectomorphisms (different) for each ball (assuming that all balls are initially centered at the origin so that their images are isosceles right triangles) that transforms the image of the ball in the moment space to a desired shape and location.

This is different from what we do in this experiment: we sample directly from the boundaries of the k non-overlapping $B^4(1)$ centered at different points initially, and try to find the Hamiltonian/flow that embeds the disjoint balls so that they are contained in a larger ball of area as small as possible. The optimal packing constructions don't specify the initial centers of the balls, and since it involves translation and linear symplectomorphism in the moment space, the actual embedding might be hard to realize using neural network or polynomial.

In the experiment, we tried 4 dimensional k ball packing for $2 \leq k \leq 12$. The Hamiltonian we choose is the split Neural Network, with width 200, 2 hidden layers, macrostep 10, and microstep 2, with 15000 iteration. The initial center of the balls are either on a line (x_1 axis), or on a circle on the $x_1 - x_2$ plane. In both cases, the minimum distance between 2 nearby balls is set to be 0.1. See Table 1 for the experiment result and comparison with the fibonacci staircase experiment (to approximate $E(1, n)$ where n is integer, we choose $E(1, x)$ where x is the closest sample point to n in the staircase sample points), and Figure 5 for the flow for 4 balls packing in moment space.

It seems that the algorithm cannot give us embeddings that are close to optimal ones, and the error gets larger with more balls. In addition, the loss depends on the initial configurations of the balls. In dimension 4, placing the centers of the ball on a circle on $x_1 - x_2$ plane preforms better than putting the

k	p_k	optimal loss	line	circle	$(x,E(1,x))$
2	$\frac{1}{2}$	2	2.1343	2.1123	(2.02,2.0239)
3	$\frac{3}{4}$	2	3.4396	2.1825	(2.98,2.3358)
4	1	2	3.4584	2.3939	(4.0,2.3541)
5	$\frac{20}{25}$	2.5	3.8291	2.9495	(5.02,2.6835)
6	$\frac{24}{25}$	2.5	4.2895	3.4681	(5.98,3.0556)
7	$\frac{63}{64}$	2.6667	4.8727	4.1065	(7.0,3.3839)
8	$\frac{288}{289}$	2.8333	5.5498	4.7752	(8.02,3.6862)
9	1	3	6.0769	5.3350	(8.98,4.0177)
10	1	3.1623	7.4563	5.9137	(10.0,4.7760)
11	1	3.3166	8.2527	6.5748	/
12	1	3.4641	7.6419	7.1742	/

Table 1: 4 dimension ball packing

centers on a line (x_1 axis). This suggests that we can also learn the center of the balls. So, instead of setting the centers of balls to fixed values, we set the initial centers to be learnable variables. To avoid overlap after gradient descent (Adam), we either scale the gradient by multiplying it by a scalar between 0.01 and 1, or (if rescaling still gives overlap because the previous first momentum is large), we set the gradient to $\tilde{g} = -\frac{\beta_1}{1-\beta_1} m_{t-1}$ where m_{t-1} is the first momentum at $t-1$ and β_1 is the exponential decay rate for the first momentum. By this update rule, $m_t = \beta_1 * m_{t-1} + (1-\beta_1) * \tilde{g} = 0$, so the center will not change, and there will be no overlap. The initialization of the centers of the balls are on the circle on the $x_1 - x_2$ plane (s.t. the smallest distance between 2 nearby balls is 1), then we add random uniform noise in $[0.0, MAX\ NOISE)$, and if there is overlap, we move all initial centers away from the origin by multiplying the centers by a constant greater than 1 (s.t. the smallest distance between 2 nearby balls is 1).

We try learning centers in the $\sqcup_{i=1}^{4} B^4(1) \hookrightarrow B^4(c)$ problem. However, experiments with different settings of hyperparameters show that learning the centers cannot give us better result than fixing them initially around a circle. In addition, the loss depends on the hyperparameters heavily, suggesting that tuning parameters might give us better result. See Table 2 for the hyperparameters and results.

learn center	MAX NOISE	macro step	hidden layer	(decay rate, decay period)	loss
No	/	10	2	(0.98,100)	2.3939
Yes	1	10	2	(0.97,150)	3.5172
Yes	0.25	10	2	(0.97,150)	3.7962
Yes	1	10	2	(0.98,200)	3.2074
Yes	0.25	10	2	(0.98,200)	2.9209
Yes	0.25	5	5	(0.98,200)	2.4784

Table 2: learning centers, k=4

3.5 Simplices

Due to recent work from Haim-Kislev we can now compute the Ekeland-Hofer-Zehnder(EHZ) capacity of convex polytope.

Theorem 4. *(Haim-Kislev) Given a convex polytope $K \in \mathbb{R}^{2n}$ with non-empty interior, let F_k denote the number of $(2n-1)$ dimensional facets of K and $\{F_i\}_{i=1}^{F_K}$ be the set of these facets. Let n_i be the unit outer normal vector of F_i and define the height h_i of F_K as $\sup_{x \in K} < x, n_i >$. The EHZ capacity of K is,*

$$c_{EHZ}(K) = \frac{1}{2}\left[\max_{\sigma \in S_{F_K}, (\beta_i) \in M(K)} \sum_{1 \leq j < i \leq F_K} \beta_{\sigma(i)}\beta_{\sigma(j)} n_{\sigma(i)} J n_{\sigma(j)}\right]^{-1}$$

where

$$M(K) = \left\{(\beta_i)_{i=1}^{F_K} : \beta_i > 0, \sum_{i=1}^{F_K} \beta_i h_i = 1, \sum_{i=1}^{F_K} \beta_i n_i = 0\right\}$$

Rather than considering all polytopes we restrict our attention to simplices. Given the matrix $V \in GL_{2n}(\mathbb{R})$ with v_i denoting the i'th row we define the simplex

$$S(V) = \left\{\sum_{i=1}^{2n} \alpha_i v_i \mid \alpha_i \in \mathbb{R}, \sum_{i=1}^{2n} \alpha_i = 1\right\}$$

Every simplex is isometric to a simplex of this form for some set of vectors. For the simplex the formula for the EHZ capacity can be simplified. The i'th of this simplex is given by the i'th row of V. For the sake of notation we will say the origin is the $2n+1$'th vertex, v_{2n+1}, of $S(V)$. The facet of $S(V)$ opposite to v_i with be denoted F_i and referred to as the i'th facet.

Theorem 5. *Given the non-degenerate simplex $S(V)$, let U be V^{-1}, then*

$$c_{EHZ}(S(V)) = \frac{1}{2}\left[\max_{\sigma\in S_{2n}} \sum_{1\leq j<i\leq 2n} sign_\sigma(i,j)U_{ij}\right]^{-1}$$

$$c_{EHZ}(S(V)) = \frac{1}{2}\left[\max_{\sigma\in S_{2n}} \sum_{1\leq j<i\leq 2n} sign_\sigma(i,j)\beta_i\beta_j n_i J n_j + \sum_{1\leq i\leq 2n} \beta_i n_i J m\right]^{-1}$$

where

$$sign_\sigma(i,j) = \begin{cases} 1 & \sigma(i) > \sigma(j) \\ 0 & i = j \\ -1 & \sigma(i) < \sigma(j) \end{cases}$$

$$\beta = -N^{-T} m$$

$$m = V^{-1} 1_{2n\times 1}$$

Proof. The permutation in Haim-Kislev's formula for EHZ capacity corresponds to order in which facets are traversed by the closed loop $z(t)$, the Legendre dual to the reeb orbit. From proposition 3.3 we see that the summation is equivalent to an integral along z, which implies that for a given $\beta \in M(K)$ the sum is invariant up to cycling σ.

For the simplex there are $2n+1$ facets so we pick σ with a fixed point at $2n+1$. For a given $(\beta_i)_{i=1}^{2n+1}$ we have,

$$\begin{aligned} &\max_{\sigma\in S_{2n+1}} \sum_{1\leq j<i\leq 2n+1} \beta_{\sigma(i)}\beta_{\sigma(j)} n_{\sigma(i)} J n_{\sigma(j)} = \\ &\max_{\sigma\in S_{2n}} \sum_{1\leq j<i\leq 2n} \beta_{\sigma(i)}\beta_{\sigma(j)} n_{\sigma(i)} J n_{\sigma(j)} + \sum_{1\leq i\leq 2n} \beta_{\sigma(i)}\beta_{2n+1} n_{\sigma(i)} J n_{2n+1} \end{aligned} \tag{5}$$

We can rewrite the first summand as follows

$$\begin{aligned} &\max_{\sigma\in S_{2n}} \frac{1}{2} \sum_{1\leq j\leq 2n, 1\leq i\leq 2n} sign_{Id}(i,j)\beta_{\sigma(i)}\beta_{\sigma(j)} n_{\sigma(i)} J n_{\sigma(j)} = \\ &\max_{\sigma\in S_{2n}} \frac{1}{2} \sum_{1\leq j\leq 2n, 1\leq i\leq 2n} sign_{Id}(\sigma^{-1}(i),\sigma^{-1}(j))\beta_i\beta_j n_i J n_j = \\ &\max_{\sigma\in S_{2n}} \frac{1}{2} \sum_{1\leq j\leq 2n, 1\leq i\leq 2n} sign_{\sigma^{-1}}(i,j)\beta_i\beta_j n_i J n_j = \\ &\max_{\sigma\in S_{2n}} \sum_{1\leq j<i\leq 2n} sign_\sigma(i,j)\beta_i\beta_j n_i J n_j \end{aligned} \tag{6}$$

Now we show that for a simplex $M(K)$ is a singleton set. Note that all the heights to the facets containing the origin are 0 as the dot product of any point in the simplex with the outer normal can not be positive. Let h denote the height to the last face. Thus we have that $\beta_{2n+1} = \frac{1}{h}$ from the second constraint. The third constraint can be rewritten as

$$N^T \beta = -\frac{n_{2n+1}}{h}$$

where β is the column vector containing the first $2n$ values of beta in order.

An outer normal vector to the i'th facet, where $i \neq 2n+1$, is a vector that satisfies the equations $n_i \cdot v_j = -\lambda \delta_{ij}$ where λ is some positive scalar for all $1 \leq j \leq 2n$. This mean that

$$NV^T = -\Lambda$$

where Λ is a positive-definite diagonal matrix. Solving for N yields

$$N = -\Lambda V^{-T}$$

which implies that that N is non-singular. Therefore there can at most one solution the the constraints. However $M(K)$ is non-empty since every compact convex subdomain of R^{2n} has an EHZ capacity. n_{2n+1} is the vector such that

$$v_i \cdot n_{2n+1} = h$$

for all non-zero vertices.

$$\frac{n_{2n+1}}{h} = V^{-1} 1_{2n \times 1}$$

Denoting this value by m completes the proof.

□

Corollary 5.1. $c_{EHZ}(S(I_{2n})) = \frac{1}{2n}$

Proof.

$$N = -\Lambda V^{-T} = -I_{2n}$$

$$m = \beta = 1_{2n \times 1}$$

$n_i J n_j$ is 1 if $i = j + n$ and 0 otherwise. So the maxima is attained when σ is the identity. The second summand is 0 since $n_i J m$ is 1 for $1 \leq i \leq n$ and -1 for $n < i \leq 2n$. Thus the maximal value for the sum is n and so the EHZ capacity is $\frac{1}{2n}$. □

3.6 Viterbo's Conjecture

We will now consider the following conjecture of Viterbo [2000]

Conjecture 6. *(Viterbo, 2000) For a convex body $K \in \mathbb{R}^{2n}$ the following inequality holds*

$$\frac{n! c_{EHZ}(K)^n}{Vol(K)} \leq 1$$

We will refer to the right-hand-side as the systolic ratio. The rest of the section will be focused on maximizing this ratio. To do so is equivalent to solving the following mini-max equation

$$M_n = \min_{V \in SL_{2n}(\mathbb{R})} \max_{\sigma \in S_{2n}} \sum_{1 \leq j < i \leq 2n} sign_\sigma(i,j)\beta_i\beta_j n_i J n_j + \sum_{1 \leq i \leq 2n} \beta_i n_i J m$$

where β_i, n_i are those given in the equations of theorem 4 for the $S(V)$. Note that the value being minimized is $(2c_{EHZ}(S(V)))^{-1}$, which we will notate as M_V. Viterbo's conjecture implies that $M_n \geq \frac{1}{2}\sqrt[n]{\frac{(2n)!}{n!}}$. Using Sterling's approximation we have that $M_n \geq \frac{2n}{e}$. By Corollary 4.1 we have the upper bound $M_n \leq M_I = n$. The rest of this section will be on attempting to find bounds on M_n.

3.6.1 Simplex EHZ Capacity Algorithm

We can obtain upper bounds on M_n computationally by searching for matrices V that give large values. A natural method for computing the EHZ capacity of a given simplex is to iterate over all permutations of σ, compute the summation and take the largest value. However this approach is $\mathcal{O}((2n)!)$ in all cases. So instead we propose an alternative algorithm in which maximal permutation is built up successfully through the addition of relations. A constraint set, C, is a collection of tuples (i,j) such that i and j are integers ranging from 1 to n. We say that $\sigma \in S_{2n}$ satisfies C if for all $(i,j) \in$ C we have that $\sigma(j) > \sigma(i)$. C can be thought of as the basis of a partially ordering on $\{1, ...2n\}$, and can be implemented with a directed graph. The psuedocode for this algorithm is shown bellow. $sign_{C,M}(i,j)$ equals $sign_{M_{ij}}$ if there exists a σ following the constraints C such that $sign_\sigma = sign_{M_{ij}}$, otherwise it is $-sign_{M_{ij}}$. This ensures that the heuristic being used is admissible.

Algorithm 1: Simplex EHZ capacity

```
$M_{ij} := \beta_i \beta_j n_i J n_j$ for all i,j such that $1 \leq i < j \leq 2n$;
constraints := Empty Constraints;
H(C) := $\sum_{1 \leq i < j \leq 2n} sign_{C,M}(i,j) M_{ij}$;
pq = contraints;
max = $-\infty$;
while pq is non-empty do
    c $\rightarrow$ pop $argmin_{c \in pq} H(c)$ from pq;
    if H(C) < max then
        break;
    end
    s = $\sigma$ that satisfies c;
    max = maximum(max, $\sum_{1 \leq i < j \leq 2n} sign_s(i,j) M_{ij}$);
    k, l = values such that $\sigma(k)$, $\sigma(l)$ are not related by c;
    if there exits such a k, l then
        leftConstraint := c + relation($\sigma(k) < \sigma(l)$);
        rightConstraint := c + relation($\sigma(k) > \sigma(l)$);
        add leftConstraint, rightConstraint to pq;
    end
end
$c_{EHZ}(S(V)) = \frac{1}{2}[M_{ij} + \sum_{1 \leq i \leq 2n} \beta_i n_i J m]^{-1}$;
```

Using this algorithm to compute EHZ capacity we can search for simplicies that maximize the systolic ratio by applying gradient descent on V.

Viterbo's conjecture trivially provides the sharpest bounds when $n = 1$. For $n = 2$ no simplices were found whose systolic ratio was greater than that of the orthosimplex. However the data suggests that there are simplicies not linearly symplectomorphic to the orthosimplex with systolic ratios equal to the orthosimplex. We can check if two simplices, $S(V)$ and $S(U)$, are linearly symplectomorphic by checking if the linear transformation $V^{-1}U$ is a symplectomorphism. For n from 3 to 5 simplices were found with systolic ratios greater than that of the orthosimplex. The results of this experimentation are collected in the following table.

4 Discussion and Future Work

A Appendix

A.1 Hyperparameters

Hyperparameter	Setting
Initial Learning Rate	0.01
Learning Rate Decay Period	5000
Learning Rate Decay Factor	0.5

Table 3: **Leapfrog** $E(1,5)$ **Hyperparameters**

A.2 Alternative nonlinearities

Another parameterization of symplectic maps that we tried was inspired by standard neural networks. One such construction was described in section 2.4, but we will here describe a more general construction.

A.3 Euler's methods pathologies

To get the exact flow of a Hamiltonian, we need to solve Equation 1. However, the hamiltonians (high degree polynomial and neural network) are too complicated to be solved exactly, suggesting that we should use numerical integration to approximate the actual flow. Euler's method is a widely used integrator to solve PDE, but it is not symplectic after each step of integration. To see this, let $H(x,y) : \mathbb{R}^{2n} \to \mathbb{R}$ be the Hamiltonian, Euler's method updates the positions in the following way,

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \delta J \nabla H$$

so, the Jacobian of this transformation is

$$\frac{\partial(\hat{x},\hat{y})}{\partial(x,y)} = I + \delta J D^2 H(x,y)$$

where $D^2H(x,y)$ is the Hessian of $H(x,y)$. However, there is no guarantee that $\frac{\partial(\hat{x},\hat{y})}{\partial(x,y)}$ is symplectic (i.e. satisfies $(\frac{\partial(\hat{x},\hat{y})}{\partial(x,y)})J(\frac{\partial(\hat{x},\hat{y})}{\partial(x,y)})^T = J$).

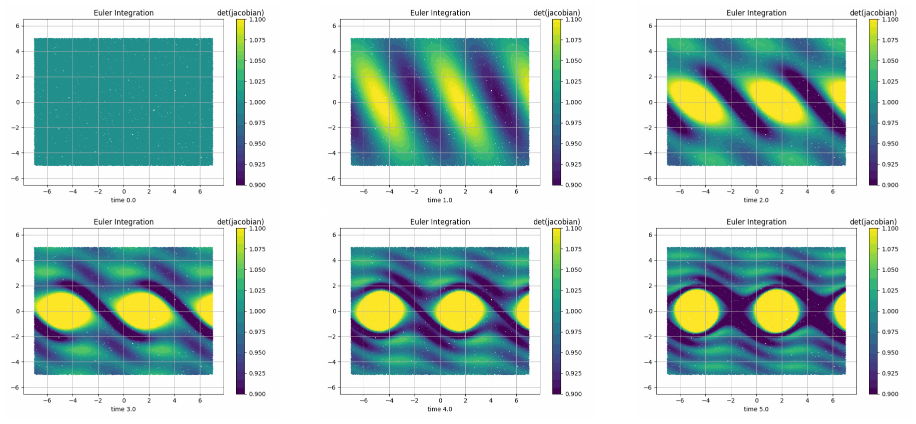


Figure 6: determinant of Jacobian for the pendulum

Take the 2 dimensional pendulum as an example. The Hamiltonian for the pendulum is $H(x, y) = \frac{1}{2}y^2 - \cos x$, where $x, y \in \mathbb{R}$. In 2d, being symplectic is equivalent to area preserving, which means that the determinant of the Jacobian of the transformation should be 1. However, Figure 6 shows that the Jacobian is not constant. (In this example, we calculate the determinant of the Jacobian from time 0 to time 5, with $\delta = 0.1$.)

Indeed, if we use Euler's method to integrate the Hamiltonian, we can get ”embeddings” that break volume constraints. For example, in the $E(1, 5) \hookrightarrow B$ problem, after 1000 iterations, Euter's method gives a embedding into $B(0.0241)$, which is way below the volume constraint $B(\sqrt{5})$. Figure 7 shows the loss during the training.

A.4 Resampling pathologies

When using only one set of sample points, our algorithm manages to learn an embedding significantly better than optimal. This is because the loss function only takes into account the points that have been sampled, and so as our symplectomorphism has overfitted to these points it is possible to take them within a smaller ball than the ellipsoid they border can possibly squeeze into. When validating these flows by resampling with more new points, the recalculated loss is significantly higher. However, when resampling every step, even with the same number of points, the validated loss and the sampled loss both converge. Figure 8 shows the algorithm running on $E(1, 4) \hookrightarrow B^4(c)$ with and without resampling using 200 sample points for learning and 2000 sample points for validation.

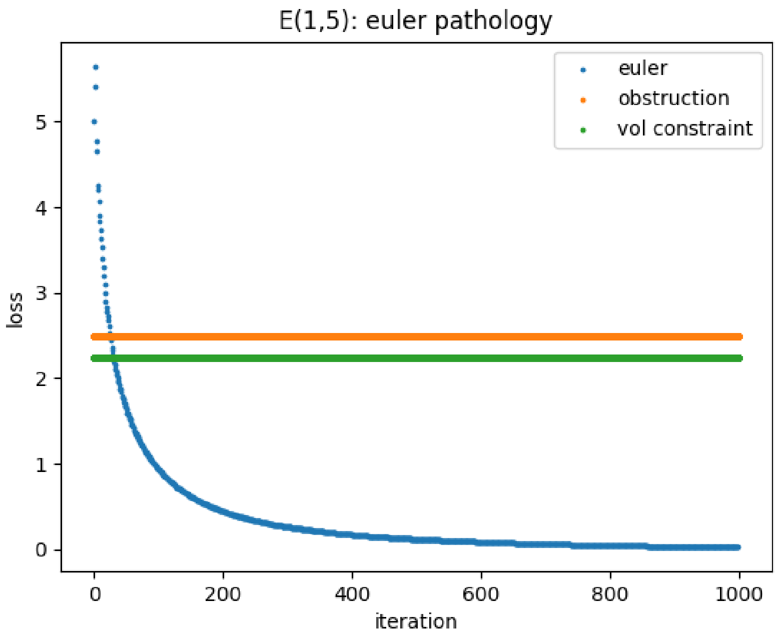
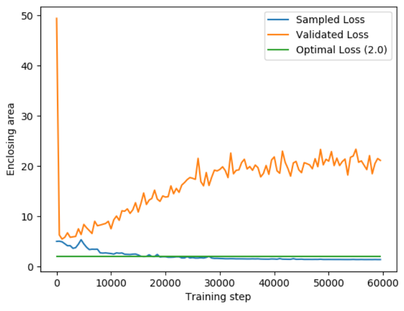
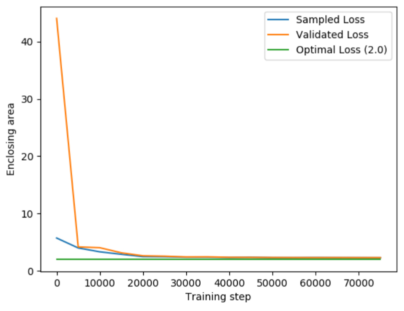


Figure 7: $E(1,5)$ and Euler's method



(a) No resampling



(b) Resampling every step

Figure 8: $E(1,4) \hookrightarrow B^4(c)$

A.5 Φ_0 and folding

In the chapter 3 of Schlenk [1999], Schlenk describes a technique to construct embeddings known as folding. Folding embedding have been proven to be sharp for certain polydiscs and ellipsoids being embedded into balls (see Schlenk [1999]) for details.

Due to the poor performance of our computational approach on polydiscs we adapted parts of the folding construction in our algorithm. The goal was to transform the domain into a different coordinate space which may be more conducive to learning.

Let ϵ be a positive real number. Denote by $D(a) \in \mathbb{R}^2$ the disc of area a and $R(a) \in \mathbb{R}^2$ the rectangle $[-\frac{1}{2}, \frac{1}{2}] \times [0, a]$. The folding construction begins with a symplectomorphism $\alpha : D(a) \hookrightarrow R(a)$ such that

$$\alpha(z)_x \leq \pi(\|z\|^2) + 2\epsilon, \forall z \in D(a) \tag{7}$$

where $\alpha(z)_x$ is the x coordinate of the image under α.

The existence of such symplectomorphism is easy to prove (using the flexibility of dimension 2); however, to get the explicit symplectomorphism is hard. The general idea is to map loops of circle in $D(a)$ to loops of rounded rectangle (or circle) in $R(a)$ by fixing the angles ϕ in the polar coordinate, and expanding or contracting the distances r to turn the circle into the rounded rectangle/circle (this gives us an area preserving diffeomorphism β). Then, we can solve an initial value problem, which gives us the corresponding angle change $h(r, \phi)$ such that β composite with this angle change ($re^{i\phi} \to \beta(re^{ih(r,\phi)})$ is the symplectomorphism we need.

Below is an approximate construction of this map α.

First, we construct the area preserving diffeomorphism $\hat{\beta} : \mathbb{R}^2 \to \mathbb{R}^2$ by specifying the loops of rounded rectangles (or circle) that each circle of radius r is mapped to:

a) for $\pi r^2 \leq \epsilon'$, $\hat{\beta}$ is the identity map, and circles are mapped to circles;

b) for $\epsilon' < \pi r^2 \leq \epsilon(1 - \epsilon/a)$, circle of radius r is mapped to a rounded rectangle $[-\frac{\epsilon}{2}s, \frac{\epsilon}{2}s] \times [-\frac{1-\epsilon/a}{2}s, \frac{1-\epsilon/a}{2}s]$, where $\delta < s \leq 1$ is a parameter to make sure that the map is area preserving, and δ is a small quantity that depends on ϵ';

c) for $\epsilon(1 - \epsilon/a) < \pi r^2 \leq a$, circle of radius r is mapped to a rounded rectangle $[-\frac{\epsilon}{2}(1 + s), \frac{\epsilon}{2} + (a - \frac{3\epsilon}{2})s] \times [-\frac{1-\epsilon/a}{2} - \frac{\epsilon}{2a}s, \frac{1-\epsilon/a}{2} + \frac{\epsilon}{2a}s]$, where $0 < s \leq 1$ is a parameter to make sure that the map is area preserving;

d) for $a < \pi r^2 \leq 4k^2a^2$, circle of radius r is mapped to a rounded rectangle $[-\epsilon - (ka - \epsilon)s, a - \epsilon + (ka - a + \epsilon)s] \times [-\frac{1}{2} - (ka - \frac{1}{2})s, \frac{1}{2} + (ka - \frac{1}{2})s]$, where $0 < s \leq 1$ is a parameter to make sure that the map is area preserving;

e) for $\pi r^2 > 4k^2a^2$, circle of radius r is mapped to a rounded square of area πr^2 centered at the origin.

After applying translation $(-\epsilon, \frac{1}{2})$ to the above area preserving diffeomorphism $\hat{\beta}$, we get the diffeomorphism β that can map $D(a)$ to $R(a)$. Notice that β is already symplectic for $\pi r^2 \leq \epsilon'$, we only need to find the angle change

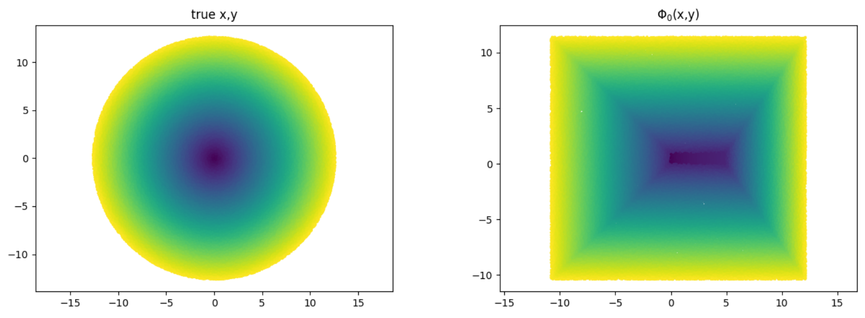


Figure 9: Φ_0

$h(r, \phi)$ for maps from circles to rounded rectangles. Since we can make the rounded corner arbitrarily small, we can assume that the rounded rectangles are rectangles. Without loss of generality, we find such $h(r, \phi)$ for the map from circle of radius R to rectangle of shape $[-wl, wr] \times [-h, h]$, where wl and wr represents the left width and right width respectively and depends on r.

After solving Equation 3.1.3 from Schlenk [1999], we get:
a) $h(r, \phi) = \arctan \frac{r}{wr'wr}\phi$ for $0 \le \phi < \frac{wr'h}{r}$
b) $h(r, \phi) = \frac{\pi}{2} - \arctan \frac{-r\phi + c_1}{h'h}$ for $\frac{wr'h}{r} \le \phi < \pi - \frac{wl'h}{r}$
c) $h(r, \phi) = \pi + \arctan \frac{r(\phi - \pi)}{wl'wl}$ for $\pi - \frac{wl'h}{r} \le \phi < \pi + \frac{wl'h}{r}$
d) $h(r, \phi) = \frac{3}{2}\pi - \arctan -\frac{r\phi + c_3}{h'h}\phi$ for $\pi + \frac{wl'h}{r} \le \phi < 2\pi - \frac{wr'h}{r}$
e) $h(r, \phi) = 2\pi + \arctan \frac{r(\phi - 2\pi)}{wr'wr}$ for $2\pi - \frac{wr'h}{r} \le \phi < 2\pi$
where $wr' = \frac{\partial wr}{\partial r}$, $wl' = \frac{\partial wl}{\partial r}$, and $h' = \frac{\partial h}{\partial r}$, $c_1 = wr * h' + wr' * h$, and $c_3 = 2\pi r - c_1$.

The above construction of β and h gives us the desired map satisfying Equation 7. See Figure 9 and Figure 10 for the map and its inverse($a = 5$, $k = 3$, $\epsilon = 0.1$).

References

Mikhail Leonidovich Gromov. Pseudo-holomorphic curves in symplectic manifolds, 1985.

Dusa McDuff. Symplectic embeddings of 4-dimensional ellipsoids, 2008.

Dusa McDuff and Felix Schlenk. The embedding capacity of 4-dimensional symplectic ellipsoids, 2009.

Felix Schlenk. On symplectic folding, 1999.

Felix Schlenk. Packing symplectic manifolds by hand, 2004.

Lisa Traynor. Symplectic packing constructions, 1995.

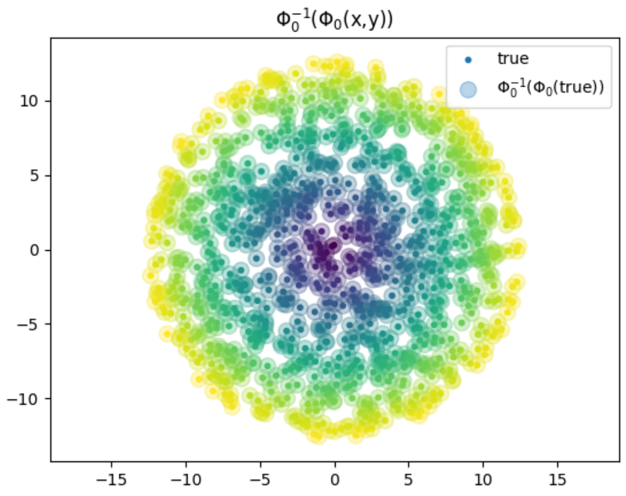


Figure 10: $\Phi_0^{-1} \circ \Phi$

D. V. Turaev. Polynomial approximations of symplectic dynamics and richness of chaos in nonhyperbolic area-preserving maps, 2003.

Claude Viterbo. Metric and isoperimetric problems in symplectic geometry. 2000.