Searching for Optimal Symplectic Maps

IML Scholars: Eli Berry, Tianyang Ma, Alex Ware, Eleven Yan

Project Leaders: Aline Leite, Yefei Zhang

Faculty Mentor: Ely Kerman

December 24, 2025

1 Introduction and Background

1.1 Project Goal

The goal of our projects is to search for symplectic maps that can embed regions into the smallest possible ball. We start by defining symplectic maps in $\mathbb{R}^{2n}$.

Definition 1.1.1. A smooth invertible map $F : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is **symplectic** if

$$F^*\left(\sum_{i=1}^{n} dx_i \wedge dy_i\right) = \sum_{i=1}^{n} dx_i \wedge dy_i.$$

Remark (Meaning of the symplectic condition). Let $F : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ be a smooth map and write the standard symplectic form as

$$\omega = \sum_{i=1}^{n} dx_i \wedge dy_i.$$

Given tangent vectors $X, \bar{X} \in T\mathbb{R}^{2n}$, the form $\omega(X, \bar{X})$ can be written in matrix form as

$$\omega(X, \bar{X}) = X^\top J\,\bar{X}, \qquad J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}.$$

The condition that F is symplectic, namely $F^*\omega = \omega$, means that

$$\omega\big(dF(X), dF(\bar{X})\big) = \omega(X, \bar{X}) \quad \text{for all } X, \bar{X}.$$

Equivalently, in coordinates, this condition is

$$dF^\top J dF = J.$$

Thus, F is symplectic if and only if its Jacobian matrix preserves the bilinear form defined by J. This provides a convenient criterion for checking symplecticity in computations.

1.2 Symplectic Maps in $\mathbb{R}^2$

Before we attempt this problem in higher dimensions, we first consider the case in $\mathbb{R}^2$. From our definition, we can see that any symplectic map preserves volume (or area in $\mathbb{R}^2$). However, only in 2 dimensions do we have that the converse is true.

Proposition 1.2.1. *Every symplectic map $F : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^{2n}$ preserves volume. In particular, symplectic maps preserve area in $\mathbb{R}^2$.*

Theorem 1.2.2. *A smooth invertible map $F : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ is symplectic if and only if it preserves area.*

1.3 Rigidity of Symplectic Maps

Theorem 1.3.1 (Gromov)**.** *If $r > 1$, then there is no symplectic map of $\mathbb{R}^{2n}$ that takes $B^{2n}(r)$, the ball of radius r, into $Z(1) = \{x_1^2 + y_1^2 \leq 1\} \subset \mathbb{R}^{2n}$.*

Even though there exist volume-preserving maps that could accomplish this embedding, no symplectic map can, which is a clear counter-example showing that not every volume-preserving map is symplectic. This rigidity phenomenon motivates our focus on explicit constructions of symplectic maps in higher dimensions, which we now specialize to $\mathbb{R}^4$.

1.4 Symplectic Maps in $\mathbb{R}^4$

We work on $\mathbb{R}^4$ with canonical coordinates (x^1, x^2, y^1, y^2), where x plays the role of "position" and y plays the role of "momentum". The *standard symplectic form* is

$$\omega \;=\; dx^1 \wedge dy^1 \;+\; dx^2 \wedge dy^2.$$

Smooth maps and Jacobians. Let $F : \mathbb{R}^4 \longrightarrow \mathbb{R}^4$ be a smooth map,

$$F(x^1, x^2, y^1, y^2) = \big(f_1(x,y),\ f_2(x,y),\ g_1(x,y),\ g_2(x,y)\big),$$

with component functions $f_1, f_2, g_1, g_2 : \mathbb{R}^4 \longrightarrow \mathbb{R}$. Its Jacobian (differential) dF at a point is the 4×4 matrix

$$dF \;=\; \begin{pmatrix} \frac{\partial f_1}{\partial x^1} & \frac{\partial f_1}{\partial x^2} & \frac{\partial f_1}{\partial y^1} & \frac{\partial f_1}{\partial y^2} \\ \frac{\partial f_2}{\partial x^1} & \frac{\partial f_2}{\partial x^2} & \frac{\partial f_2}{\partial y^1} & \frac{\partial f_2}{\partial y^2} \\ \frac{\partial g_1}{\partial x^1} & \frac{\partial g_1}{\partial x^2} & \frac{\partial g_1}{\partial y^1} & \frac{\partial g_1}{\partial y^2} \\ \frac{\partial g_2}{\partial x^1} & \frac{\partial g_2}{\partial x^2} & \frac{\partial g_2}{\partial y^1} & \frac{\partial g_2}{\partial y^2} \end{pmatrix}.$$

Symplectic condition in $\mathbb{R}^4$ (recall). In $\mathbb{R}^4$ the standard symplectic matrix is

$$J = \begin{pmatrix} 0 & I_2 \\ -I_2 & 0 \end{pmatrix}.$$

By the general criterion above, F is symplectic if and only if

$$dF^{\top}J\,dF = J.$$

Groups of symplectic and Hamiltonian maps. Define

$$\mathrm{Symp}(\mathbb{R}^4,\omega) := \big\{F : \mathbb{R}^4 \to \mathbb{R}^4 \text{ smooth} \,\big|\, F^*\omega = \omega\big\}.$$

This is a group under composition. We will later see that *Hamiltonian flows* (time-1 maps of Hamiltonian vector fields) belong to this group.

1.5 Hamiltonian vector fields and flows

Hamiltonian vector field. Given a Hamiltonian $H : [0,1]\times\mathbb{R}^4 \to \mathbb{R}$, its Hamiltonian vector field $X_H(t,\cdot)$ is defined at each time t by

$$\iota_{X_H(t,\cdot)}\omega \;=\; dH_t, \qquad H_t(\cdot) := H(t,\cdot).$$

In coordinates this means

$$\begin{pmatrix} X_H^1 & X_H^2 & Y_H^1 & Y_H^2 \end{pmatrix} J \;=\; \begin{pmatrix} \frac{\partial H}{\partial x^1} & \frac{\partial H}{\partial x^2} & \frac{\partial H}{\partial y^1} & \frac{\partial H}{\partial y^2} \end{pmatrix},$$

or equivalently

$$X_H \;=\; \frac{\partial H}{\partial y^1}\frac{\partial}{\partial x^1} + \frac{\partial H}{\partial y^2}\frac{\partial}{\partial x^2} - \frac{\partial H}{\partial x^1}\frac{\partial}{\partial y^1} - \frac{\partial H}{\partial x^2}\frac{\partial}{\partial y^2}.$$

where all partial derivatives are taken with respect to the spatial variables, with time t held fixed.

Thus the corresponding ODEs (Hamilton's equations) are

$$\dot{x}^i = \frac{\partial H}{\partial y^i}, \qquad \dot{y}^i = -\frac{\partial H}{\partial x^i}, \qquad i = 1,2.$$

Flow of a vector field. The (time-dependent) flow $\phi_X^t : \mathbb{R}^4 \to \mathbb{R}^4$ of a vector field X is the solution operator of

$$\frac{d}{dt}\,\phi_X^t(z) \;=\; X\big(\phi_X^t(z)\big), \qquad \phi_X^0 = \mathrm{id}.$$

For a Hamiltonian vector field X_H, we write ϕ_H^t.

Hamiltonian flows are symplectic. One shows (e.g. via Cartan's magic formula $L_{X_H}\omega = d(\iota_{X_H}\omega) + \iota_{X_H}d\omega = d(dH_t) + \iota_{X_H}(0) = 0$) that

$$(\phi_H^t)^*\omega = \omega \quad \text{for all } t.$$

Hence every time-t map ϕ_H^t is symplectic; in particular, $\phi_H^1 \in \mathrm{Symp}(\mathbb{R}^4, \omega)$. We denote the set of all such time-1 maps by

$$\mathrm{Ham}(\mathbb{R}^4, \omega) := \{ \, \phi_H^1 \mid H \in C_c^\infty([0,1] \times \mathbb{R}^4) \, \},$$

the (compactly supported) Hamiltonian diffeomorphisms. This shows that $\mathrm{Ham}(\mathbb{R}^4, \omega) \subset \mathrm{Symp}(\mathbb{R}^4, \omega)$.

1.6 Two elementary symplectic building blocks

The following maps are time-1 flows of simple Hamiltonians, hence symplectic. We also verify the Jacobian criterion $dF^\top J dF = J$ explicitly.

(1) Shears in the x-coordinates

Let $F : \mathbb{R}^2 \to \mathbb{R}$ be smooth in (y^1, y^2) and define

$$S_X(F) : \; (x^1, x^2, y^1, y^2) \longmapsto \big(x^1 + \partial_{y^1} F(y), \;\; x^2 + \partial_{y^2} F(y), \;\; y^1, \;\; y^2\big).$$

Hamiltonian: $H(y) = F(y)$. Then $\dot{x}^i = \partial_{y^i} F$, $\dot{y}^i = 0$, so time-1 gives $S_X(F)$.

Jacobian check:

$$dS_X(F) = \begin{pmatrix} I_2 & H \\ 0 & I_2 \end{pmatrix}, \qquad H = \nabla_y^2 F = \begin{pmatrix} F_{y^1 y^1} & F_{y^1 y^2} \\ F_{y^2 y^1} & F_{y^2 y^2} \end{pmatrix}.$$

Since $H^\top = H$ (Hessian is symmetric), $dS_X(F)^\top J \, dS_X(F) = J$.

(2) Cross-couplers between x^1 and y^2

Let $f, g : \mathbb{R} \to \mathbb{R}$ be smooth and define

$$F_{XY}(f, g) : \; (x^1, x^2, y^1, y^2) \longmapsto \big(x^1, \;\; x^2 + f(x^1) g'(y^2), \;\; y^1 - g(y^2) f'(x^1), \;\; y^2\big).$$

Hamiltonian: $H(x^1, y^2) = f(x^1) g(y^2)$. Then

$$\dot{y}^1 = -f'(x^1) g(y^2), \quad \dot{x}^2 = f(x^1) g'(y^2), \quad \dot{x}^1 = \dot{y}^2 = 0,$$

so time-1 gives $F_{XY}(f, g)$.

Jacobian check: with $a = -g f''$, $b = -f' g'$, $c = f' g'$, $d = f g''$ (all at (x^1, y^2)),

$$dF_{XY} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ c & 1 & 0 & d \\ a & 0 & 1 & b \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A direct multiplication shows $dF_{XY}^\top J \, dF_{XY} = J$ (the only potentially nonzero off-block term is proportional to $b + c$, which vanishes since $b = -c$).

Why composition preserves symplecticity If Φ_1 and Φ_2 are symplectic, then so is their composition:

$$(\Phi_2 \circ \Phi_1)^* \omega = \Phi_1^*(\Phi_2^* \omega) = \Phi_1^*(\omega) = \omega.$$

Thus any finite concatenation of our building blocks is automatically symplectic. This simple fact underlies both our parameterisation strategy and the construction of symplectic numerical methods such as the leapfrog (Störmer–Verlet) integrator.

Proposition 1.6.1 (Construction of Hénon-like maps)**.** *Maps of the form*

$$(x, y) \longmapsto (y, -x + \nabla V(y)), \qquad V : \mathbb{R}^n \longrightarrow \mathbb{R},$$

can be obtained as finite compositions of the symplectic building blocks introduced above.

1.7 Approximation by Hénon-like maps

The following is an important theorem used for approximating the space of symplectic maps.

Theorem 1.7.1. (*Turaev*) *Every symplectic map of $\mathbb{R}^{2n}$ can be approximated on any compact set by the composition of maps of the form $(x, y) \mapsto (y + \eta, -x + \nabla V(y))$ where $x, y \in \mathbb{R}^n$, $V : \mathbb{R}^n \longrightarrow \mathbb{R}$ a polynomial and $\eta \in \mathbb{R}^n$ a constant vector.*

We will refer to these types of maps as Hénon-like maps for the rest of this paper. It is important to note that Hénon-like maps are actually of the form

$$(x, y) \mapsto (y, -x + \nabla V(y))$$

where V is any smooth function $\mathbb{R}^n \longrightarrow \mathbb{R}$. It can be shown that Hénon like maps in this form can be constructed from our previous set of maps by composing them in nice ways. If we take the smooth functions to be restricted to polynomials, then we do not immediately have that this approximates the space of symplectic maps. Turaev's proof of this theorem demonstrates why we need the constant vector η in the Hénon-like maps.

1.8 Regions of Interest

In $\mathbb{R}^4$ we focused on the following regions:

Ellipsoids $\quad E(1, a) = \left\{\pi(x_1^2 + y_1^2) + \frac{\pi(x_2^2+y_2^2)}{a}) \leq 1\right\}$

Polydisks $\quad P(1, a) = \left\{\pi(x_1^2 + y_1^2) \leq 1, \pi(x_2^2 + y_2^2) \leq a\right\}$

Lagrangian tori $\quad L(1, a) = \left\{\pi(x_1^2 + y_1^2) = 1, \pi(x_2^2 + y_2^2) = a\right\}$

Now we define the embedding capacity as follows. Consider the open ball

$$B(c) = \left\{ \pi(x_1^2 + y_1^2 + x_2^2 + y_2^2) < c \right\}$$

Then for a region R,

$$c(R) := \inf \left\{ c \;\middle|\; \exists \text{ symplectic } \phi : \mathbb{R}^4 \to \mathbb{R}^4 \text{ with } \phi(R \subset B(c) \right\}.$$

The values of this function for the ellipsoids $E(1, a)$ are completely known.

Theorem 1.8.1. (*McDuff-Schlenk,* [3]) *The embedding capacity for the ellipsoid* $E(1, a)$ *follows the Fibonnaci stairs sequence. In particular,* $c(E(1, 2)) = c(E(1, 3) = c(1, 4) = 2$ *and* $c(E(1, 6.25) = 2.5$

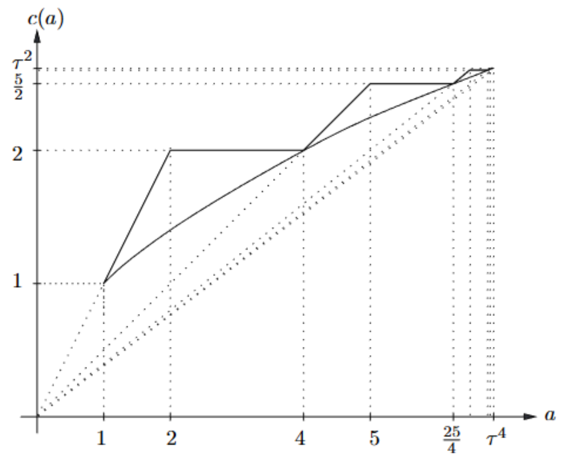

Figure: Fibonacci stairs sequence

As we can see, there exist volume-preserving embeddings of $E(1, 2)$ into $B(\sqrt{2})$, but the best symplectic embedding is $B(2)$, which it is trivially embedded into. The interesting regions are $E(1, 4)$ and $E(1, 6.25)$ in which there exist symplectic embeddings into their equal volume balls.

2 Algorithm

2.1 Optimization Framework

We emphasize that our optimization strategy differs between $\mathbb{R}^2$ and $\mathbb{R}^4$. In $\mathbb{R}^2$, symplectic maps are exactly the smooth invertible area-preserving maps, which allows for a relatively flexible parameterization. In contrast, in $\mathbb{R}^4$ symplectic rigidity prevents arbitrary volume-preserving maps from being symplectic. As a result, in four dimensions we restrict our search space to compositions of Hénon-like maps, which are guaranteed to be symplectic. This restriction is essential for making the optimization problem tractable in higher dimensions.

For a fixed k and fixed degree d we consider the landscape of symplectic maps that are compositions of k Hénon maps whose polynomial functions have degree d. The constants of the maps and coefficients of their polynomials are the weights we train, Given a region in $\mathbb{R}^{2n}$ we take N points on the boundary, apply a map in our landscape and evaluate a loss function which measures how close the points are to the origin. We then train the weights using the Adams optimization algorithm.

In 4D we made several refinements to the search algorithm. We still used the Adam Optimizer, but we rewrote it in the machine learning library JAX. We also redefined the loss function. Given a map Φ in our landscape and a sample of N points p_i on the boundary of D we replaced the loss function

$$L(\Phi) = \max_i \|\Phi(p_i)\|$$

with the function

$$LSE(\Phi) = \frac{1}{\tau}\log\left(\sum_{i=1}^{n}\exp\big(\tau||\Phi(p_i)||\big)\right)$$

for some fixed $\tau > 0$. This is a smoothing of the max function which provides stable, usable gradients for optimization. We have the following bounds on this function :

$$\max_i \|\Phi(p_i)\| \le LSE(\Phi) \le \max_i \|\Phi(p_i)\| + \frac{1}{\tau}\log(N)$$

In many of our simulations in $\mathbb{R}^4$, we sample as many as 10^6 points. Thus we take τ to be between 20 to 100 to give us a good upper bound on the LSE, as for most of the simulations, the hard max is around 2 to 6.

The reason we use so many points for the $\mathbb{R}^4$ algorithm is due to switching the optimization to being written in the JAX library. Before, to find the gradient we used methods such as finite difference, which were slow and often had too much error. However, switching to JAX we simply used its auto-differentiation method to precompute the gradient on a set of points before training. Thus a lot of the work being done in the algorithm is being done at the start of the program, which leads to each iteration taking considerably less time when the training starts. The downside to this method is that since we compute the gradient on the set of points, we would need to repeat that process if we wanted to resample the points during training. Thus we solve this by taking large amounts of points on the boundary, on the magnitude of 10^5 to 10^6 to avoid overfitting and to represent the regions with enough accuracy.

We contrast this with our $\mathbb{R}^2$ algorithm which does resample points after every iteration, meaning we can take anywhere from 10^2 to 10^5 points to cut down on computation. Of course the downside of this is that it has to use finite difference gradient calculation at every step, which adds a lot of time to each iteration, even though we save computation on the number of points.

2.2 Adam Optimization

In order to optimize our loss function, we use the Adam optimizer from the machine learning library Jax. Adam (Adaptive Moment Estimation) is a gradient descent algorithm that uses both the concepts of momentum and root mean square propagation. Momentum works by accounting for previous gradients to allow for smoother convergence. The momentum term is updated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla L$$

where m_t is the momentum term at time t, β_1 is the momentum decay rate, and ∇L is the gradient of the loss function. Essentially this is a weighted moving average of past gradients. The RMS (root mean square) term update rule looks similar:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L)^2$$

where v_t is the RMS term at time t, and β_2 is the RMS decay rate. As opposed to momentum, RMS propogation is a moving weighted average of the gradients squared. Next, Adam uses bias correction since the momentum and RMS terms are initially set to 0:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Thus we have the final update rule for our parameters in the loss function:

$$\theta_t = \theta_{t-1} - \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}\alpha$$

Where θ_t are the parameters of our symplectic map at time t, ϵ is a small number to prevent division by zero, and α is the initial learning rate. For most of the simulations, we used $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, and $\alpha = 10^{-3}$, although we later discuss how our algorithm will adjust our learning rate.

Advantages of Adam

Compared to other algorithms for gradient descent, Adam seems to have the best performance. Some observations about Adam are that it is less likely to get stuck in local minima, it is more stable, and it converges faster than other algorithms. There are a few reasons for why this could be the case. By using momentum and RMS propogation, it can use data of past gradients to determine the step size for each iteration. This means it is less likely to oscillate about local minima. Some algorithms have a hard time getting started, and Adam addresses this with its bias correction, leading to better stability. Lastly, the hyperparameters of Adam are only the initial learning rate and the weight decay rates, meaning it is easy to adjust the model.

2.3 Methods for Stability

Using Adam in practice worked very well, but there were a few issues with stability, where the algorithm was prone to making some parameters diverge. Thus we implemented a few safeguards to ensure the algorithm would stay stable.

Gradient Clipping

One issue that came up often was the gradients of the loss function becoming too large, leading to the algorithm becoming unstable. Since we are evaluating points on the composition of polynomials, the higher degree terms can get large very fast. The solution to this is to take the norm of the gradient and test check it is above some threshold c. If the norm is greater than the threshold, we scale the whole gradient by $c/||\nabla L||$ to ensure that the norm is below the threshold. This way, the direction of each step is still preserved. We chose to use $c = 10$, which helped keep the algorithm more stable.

Learning Rate Rollback

One phenomenon we observed in the algorithm was that at times, the loss function would start increasing and the optimizer would essentially stop optimizing it. One reason this could have happened was that once it took a step in the wrong direction, the momentum term could have affected it to keep going in that direction. The reason for why it could have increased the loss in the first place is by taking too large of a step size.

Thus we attempted to fix this by implementing a learning rate rollback if the loss ever started increasing by too much. What the algorithm does is it tests what the new loss would be by proceeding through the optimization as normal, and if the loss happened to have diverged, it resets the momentum of the optimizer, as well as decreases the initial learning rate, and retries the iteration. This way, the algorithm naturally finds a good learning rate to work with, as if the initial learning rate is too high, it will decrease it until it is stable.

Centering and Regularization

Lastly, we want to make sure that the center of the mapped region is near the origin, and that parameters of the symplectic map do not get too large. Thus we introduce centering and regularization terms into our loss function:

$$L(\theta) = LSE(\Phi_\theta) + w_c||C||^2 + w_r||\theta||^2$$

where w_c, w_r are the centering and regularization weights and C is the center of the region. In practice, these added terms are small, so they do not overpower the LSE.

3 Simulation and Results

3.1 Toy Examples in $\mathbb{R}^2$

Since symplectic maps in $\mathbb{R}^2$ are exactly the smooth invertible area-preserving maps, we expected that our algorithm would be good at optimizing regions in 2 dimensions. That is, for every region in $\mathbb{R}^2$, there exists a symplectic map that can embed the region into the ball of the same area around the origin. In the following paragraphs, we discuss the parameters used to accomplish each embedding: d, the degree of each polynomial, k, the number of compositions of maps, and N, the number of training iterations.

Embedding a Square into a Circle

The first region we look at is a square.

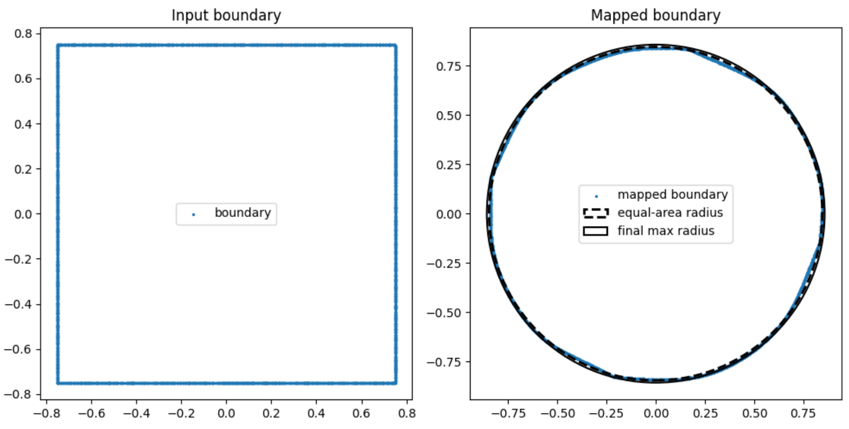

Figure: Embedding a square into a circle

The parameters we used for this embedding were $d = 5, k = 5, N = 3000$. As we can see, the algorithm can accomplish this embedding fairly easily.

Embedding a Keyhole into a Circle

Next we take a look at the Keyhole region. Since this shape is concave, one may expect that the algorithm will have a harder time optimizing it.

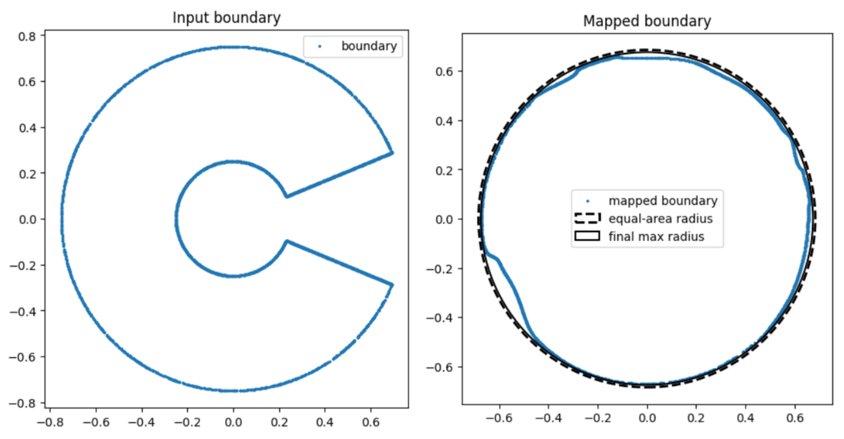

Figure: Embedding a keyhole into a circle

The parameters we used for this embedding were $d = 5, k = 7, N = 15000$. Even with a more complex shape the algorithm is still able to reach an embedding close to the equal area circle.

Embedding 3 Circles into a Circle

Lastly, we consider 3 disjoint circles to see how the algorithm works with disconnected regions. One should note that since this is not a connected region, the optimal embedding cannot be exactly the equal area circle, but the infimum of the embeddings is the equal area circle. Thus we should be able to get arbitrarily close to it.

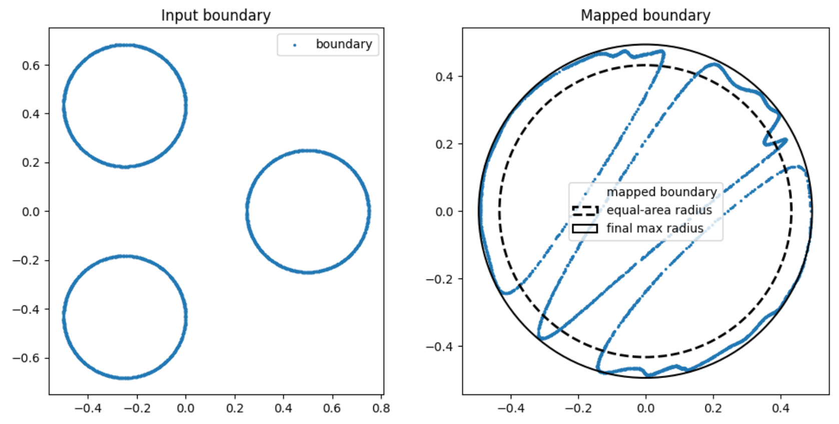

Figure: Embedding 3 circles into a circle

The parameters used for this embedding were $d = 5, k = 8, N = 30000$. Something to notes is that for many arrangements of the circles, we still converged to the same arrangements of circles in the end, where they lie on a diagonal line. As we expected, the algorithm had a bit of a harder time optimizing this region, likely due to there being multiple disconnected regions.

Overall, we were happy with our $2D$ results, as they showed that Adam Optimization was a good strategy to use for this problem. As we discussed earlier in section 2, we did need to make refinements to the algorithm, but this showed that the general idea was correct, and we could begin work in higher dimensions, where there are more interesting results.

3.2 Results in $\mathbb{R}^4$

The numbers $c(E(1,a))$ were computed in a famous paper of McDuff and Schlenk. In particular, they showed that $c(E(1,4)) = 2$ and $c(E(1,6.25)) = 2.5$. Our algorithm found concrete symplectic maps which achieved values for these cases of 2.19 and 3.05, respectively. (The maps of McDuff and Schlenk are far from explicit.)

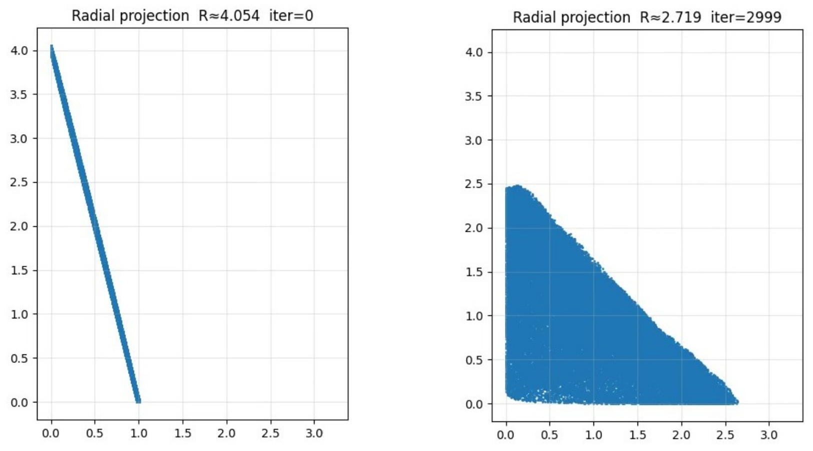

Figure: Embedding an ellipsoid into a ball

For the $E(1,4)$ embedding, we used the parameters $d = 4$, $k = 60$, and $N = 150000$. For the $E(1,6.25)$ embedding, we had parameters $d = 6, k = 60$, and $N = 50000$. In the above figure we represent the boundary of the ellipsoid on the radial projection, $x_1^2 + y_1^2$ vs $x_2^2 + y_2^2$.

There are some ways we suspect would help improve our results further. One way could be to let the algorithm run for a long time to see what happens, although we notice that after enough time, it pretty much stays around the same embedding.

We also ran simulations to verify that the algorithm did not optimize any further than it should be able to. With $E(1,2)$, it is already embedded within the optimal ball $B(2)$, so we tested if it could optimize below that. Over many simulations, the algorithm never did better than the trivial embedding, so that reassured us that the algorithm was working correctly.

4 Future Work

Even though we had success with the ellpisoids, our algorithm needs to be refined in order to approximate optimal symplectic embeddings of Lagrangian tori and polydisks.

Polydisks

For polydisks we have the following theorem as guidance.

Theorem 4.0.1 (Schlenk). *$c(P(1,a)) \leq \frac{a}{2} + 2$ for $2 \leq a \leq 6$.*

Our current algorithm struggles to get close to these bounds.
$P(1,6) \hookrightarrow B(6.18)$.

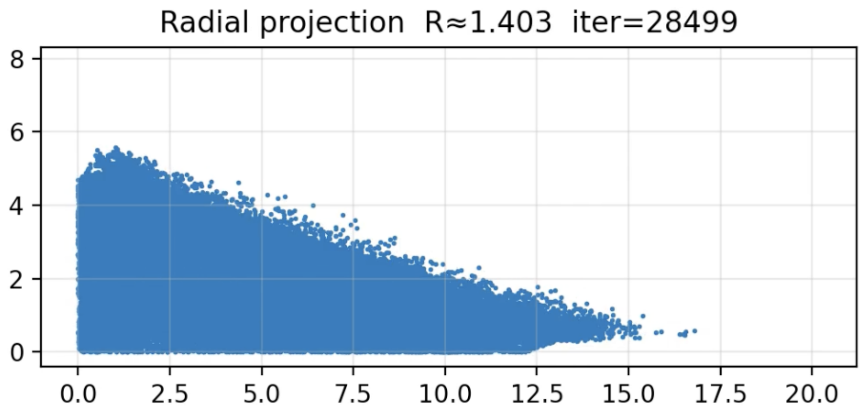

Figure: Radial projection for $P(1,6)$

The polydisk is trivially embedded into $B(7)$, so it is able to make some improvements, although we cannot be sure that this embedding is correct. Looking at the figure, there are disconnected points which likely implies that it is discounting some parts of the boundary to optimize the parts it can see, given we are using a fixed amount of points. The solution to this could be either to use a lot more boundary points, or rerunning the algorithm with the a new set of boundary points, but starting with the map we have just ended with.

Lagrangian Tori

For Lagrangian we have the following complete theorem to shoot for.

Theorem 4.0.2 (Hind-Opshtein)**.** *$c(L(1,a)) = 3$ for all $a \geq 2$.*

Again, our current algorithm struggles to get close to these bounds.
$L(1,6) \hookrightarrow B(7.02)$.

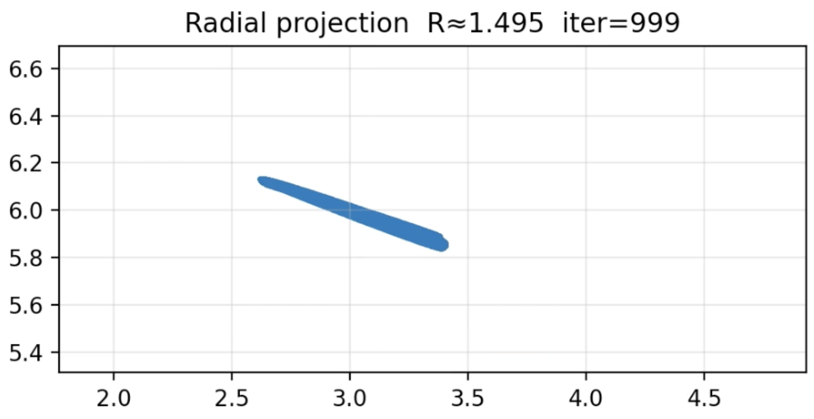

Figure: Radial projection for $L(1,6)$

Interestingly enough, the Lagrangian Tori should be able to embed at least as well as the polydisk, yet our algorithm is currently not robust enough to do so. Some future strategies to try out could be to start the optimization with the map we obtained from the polydisk, and then see what it does from there.

Future goals: Since it is not proven that the current bounds on the polydisk embeddings are the best, our future goals include searching for maps which take $P(1,6)$ into $B(\lambda)$ for λ below the known upper bound of 5. We also wish to improve the algorithm so that it comes close to matching the performance of the Hind-Opshtein map. Lastly, even though our results for the ellipsoid are satisfactory, it would be good to see if we can get even closer to the lower bound.

References

[1] M. Gromov, Pseudo-holomorphic curves in symplectic manifolds, *Inventiones Mathematicae*, 82 (1985), 307–347.

[2] R. Hind, E. Opshtein, Squeezing Lagrangian tori in dimension 4. Comment. Math. Helv. 95 (2020), no. 3, 535—567.

[3] D. McDuff and F. Schlenk, The embedding capacity of 4-dimensional symplectic ellipsoids, *Ann. of Math.*, 175 (2012), 1191–1282.

[4] F. Schlenk, Embedding problems in symplectic geometry De Gruyter Expositions in Mathematics 40. Walter de Gruyter Verlag, Berlin. 2005.