Manifold Learning for Extracting Conservation Laws

Ruba Houssami

under the direction of

Mr. Peter Y Lu
Mr. Rumen R Dangovski
Professor Marin Soljačić
Massachusetts Institute of Technology

Research Science Institute
August 2, 2021

Abstract

Conservation laws are one of the most powerful features that govern physical systems. However, it may get difficult to identify and extract these laws in complex systems and high dimensional ones. Several approaches have been suggested to automate the process of extracting these laws, including the Koopman operator, siamese neural networks (SNNs), and the AI Poincaŕe method, but these methods rely on dynamical quantities and training neural networks, which can make this process even harder. We will, instead, use a non-parametric manifold learning approach that implements the Wasserstein distance to directly extract conservation laws into a lower dimensional manifold. We will be testing the several manifold learning methods that perform dimensionality reduction on two systems: a simple pendulum system given in two versions and a system of an orbiting planet. Our results show that spectral embedding was the only method able to extract all the conserved quantities in our tests, proving its efficiency, reliability, and superiority over other methods.

Summary

Physical systems rely heavily on physics conservation laws. A pendulum, for example, has one conserved quantity—energy. We intend to discover and extract this conservation law using a machine learning method that aims to reduce dimensions. This method would make extracting conservation laws easier and quicker due to the minimal data it requires. Therefore, we have tested this method on a pendulum system and on a planet orbits system.

1 Introduction

Conservation laws, such as the conservation of energy or momentum, form one of the most important concepts in physics. These laws are crucial for many applications in science and engineering because physical systems are governed by them. We can use conservation laws to successfully predict the future state of a dynamical system and its evolution through time. Traditionally, to figure out a conservation law, we would use a mathematical approach by noticing a pattern in the dynamic equations of this system then applying these equations to calculate the conserved quantity. Additionally, based on Noether's theorem [1] which informally stated reads as "when an action has a symmetry, we can derive a conserved quantity," we would find a symmetry in a certain system then deduce the conservation laws. Although the conservation laws of a lot of systems are known and have been applied for years, there remain several cases where determining the conservation laws of a system is difficult and requires significant theoretical and computational effort. Therefore, another approach must be taken to ease the extraction of these conservation laws.

Previous work done by Brunton et al., 2016 [2], included using sparse identification to discover dynamical equations of conservation laws for nonlinear dynamical systems. Another approach has been tried by Mauroy et al., 2020 [3], which relies on constructing the dynamical system in question and using the Koopman operator to extract the conserved quantity. This method uses detailed dynamical information to observe how the system changes over time to discover the conservation laws, producing a linear embedding of nonlinear dynamics. A similar research was done by Kaiser et al., 2018 [4], using a data-driven method with the Koopman operator to discover the total energy and angular momentum in a system.

However, what these methods aim to do is build a physical system to record their dynamical properties. This dynamical approach would require very detailed time recordings and precise measures to obtain a sufficient amount of data. If we look at how conservation

laws work and what they are, we notice that these laws do not actually rely on the dynamics of a system because no matter the duration that a system stays in a certain parameter, the conserved quantity is still going to be conserved. Therefore, we should not need dynamical information to discover conservation laws because detailed dynamical information just makes it difficult to extract the conserved quantity in complicated systems. We might have random datasets of unknown dimensions (the curse of dimensionality) or a new physical system where it is hard to record and deduce this symmetry. Our approach will not focus on the dynamics of a system but on its geometrical phase space and how the system moves in this space.

Unlike previous methods, our focus will be on the shape of the trajectories and orbits of the system using manifold learning. Each trajectory taken by the system corresponds to a set of conserved quantities, and every time this trajectory changes, the conserved quantities change as well. Therefore, the change in the shape of the trajectories can directly tell us the conserved quantity. The trajectories of the system can provide a bunch of points that fill out an abstract manifold which these points move in, and we parameterize all the possible shapes of the orbits which tell us all the possible ways the conserved quantities can change. We finally obtain a manifold consisting of the conserved quantity.

Manifold learning is not a totally new method, and several people have tried similar methods. One of those methods was tested by Wetzel et al., 2020 [5], where the authors experiment with siamese neural networks (SNNs) to make these networks learn the symmetry in a system. However, it was very hard to make the SNNs work with more than one conservation law and needed careful hyperparameter tuning and repeated training on smaller and smaller subsets of the original dataset. Also, this method did not always work, and they had to run their system several times. Another approach done by Liu et al., 2020 [6], suggests using AI Poincaré to determine the dimensionality of a manifold. AI Poincaré tries all the possible ways of fitting the data into an equation and then guesses the exact equation of the

conserved quantity. This method was only able to count how many conservation laws are present in a system and did not actually extract the conservation laws, requiring a separate algorithm to actually discover the laws. Also, it did not perform well upon testing it with complicated systems and did not always work.

However, these methods also need the dynamical information of the system despite using manifold learning and did not work that well. While saimese neural networks is a parametric method that requires fitting, classical manifold learning is a non-parametric method that would execute the job without the need for a model of the system or any fitting. We will use the Wasserstein metric from optimal transport to give the necessary metric structure for the abstract manifold that will be produced by the manifold learning methods. Manifold learning [7] is an unsupervised machine learning dimensionality reduction method. Manifold learning is simply reducing a high dimensional space (or manifold), such as a 4-dimensional space (which we cannot visualize), to a lower dimensional one, such as a 2-dimensional space, in order to observe the most important changes that are occurring and to be able to visualize these changes on a graph. The final extracted manifold corresponds directly to the space of conserved quantities. Manifold learning can also help us study chaotic systems, such as the double pendulum, where it is too complicated to perform dynamical tests on our system. Lu et al., 2021 [8], proposed using the diffusion maps method, which is one of the manifold learning methods, to extract the conserved quantities in a system.

We will be testing the manifold learning methods on two physical systems: a simple pendulum system with conserved energy and an orbital system with conserved energy, angular momentum, and orientation angle. We will also take an image version of the same pendulum system to see if the manifold learning methods can work with images. Because previous methods did not really work, we propose using the Wasserstein distance metric to represent the geometry of different orbits and then use classical manifold learning methods on top of that to directly extract the manifold of conserved quantities, and thus the conservation laws.

2 Datasets

Pendulum Dataset

The pendulum is a very simple and easy physical system to begin our test on the manifold learning methods. We can clearly visualize a pendulum, and it can be demonstrated easily. The pendulum is a 2-dimensional phase space consisting of a string of length r and a mass m connected to the string. We generate our data according to the following equations of motion, such that θ is the angular position and L is the angular momentum:

$$\frac{d(\theta)}{d(t)} = L \qquad\qquad \frac{d(L)}{d(t)} = -\sin\theta.$$

The pendulum system has one scalar conserved quantity which is the conservation of energy:

$$E = KE + GPE = \frac{1}{2}L^2 + (1 - \cos\theta).$$

E corresponds to the total energy of the pendulum which is the sum of the kinetic energy KE and the gravitational potential energy GPE. We consider and record the two quantities L and θ of 100 generated trajectories of the pendulum system and plot the points on a graph to study the conserved quantity (Figure 1):

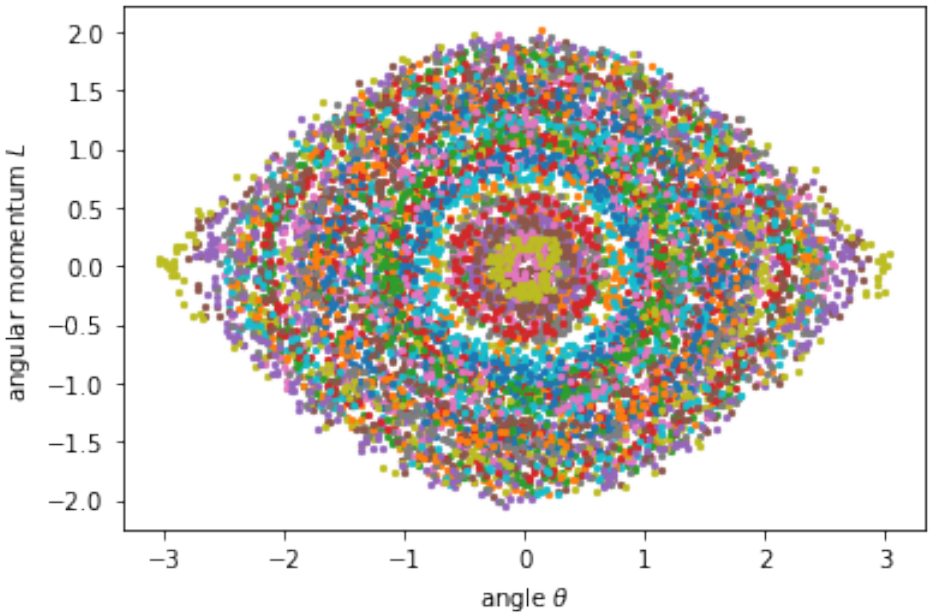


Figure 1: A noisy sample of the generated data in our pendulum system made of 100 trajectories, consisting of 100 points each, plotted according to position and angular momentum

Pendulum Image Dataset

We were interested in investigating whether our approach would work on raw images. This dataset is a very high dimensional one, and instead of providing the values of θ and L, we are only providing images of the pendulum, making it harder to extract the energy. This is the same physical system as the previous dataset, but we are providing different data. Therefore, we generated images of this pendulum system consisting of 100 trajectories and 100 sample points to experiment on, adding together 2 frames at a time:

Figure 2: A sample of the generated images in our pendulum system consisting of the position of the pendulum 0.5 time units apart

Planet Orbits Dataset

For a more complicated approach, we chose the planet orbits dataset because it has three independent conserved quantities instead of one, making the manifold of conserved quantities much more interesting. We want to test whether the manifold learning methods would be able to extract these quantities since this dataset is of higher dimensions than the pendulum dataset. It consists of one planet orbiting a star and forming fixed elliptical orbits of 2-dimensional spaces each. This dataset is 4-dimensional, consisting of 2 position dimensions and 2 momentum dimensions. We use the following equations of motion to generate our data, where r is the radius (or the relative position between the star and the planet) and p is the momentum:

$$\frac{d(r)}{d(t)} = p \qquad\qquad \frac{d(p)}{d(t)} = -\frac{\hat{r}}{|r|^2}.$$

This system has three vector conserved quantities:

$$E = \frac{p^2}{2} - \frac{1}{|r|}, \qquad L = r \times p, \qquad A = p \times L - \hat{r}.$$

In our system, these vector quantities correspond to 3 scalar conserved quantities: the total energy of the orbits E, the angular momentum L, and the orbital orientation angle ϕ which is the angle of the Laplace–Runge–Lenz (LRL) vector A relative to the x-axis [9]. To study the orientation angle, we decompose it into sine and cosine, because it has two components, in order to be able to plot each component on a 2D graph. We generate 100 planet orbits and plot the points according to position then momentum:

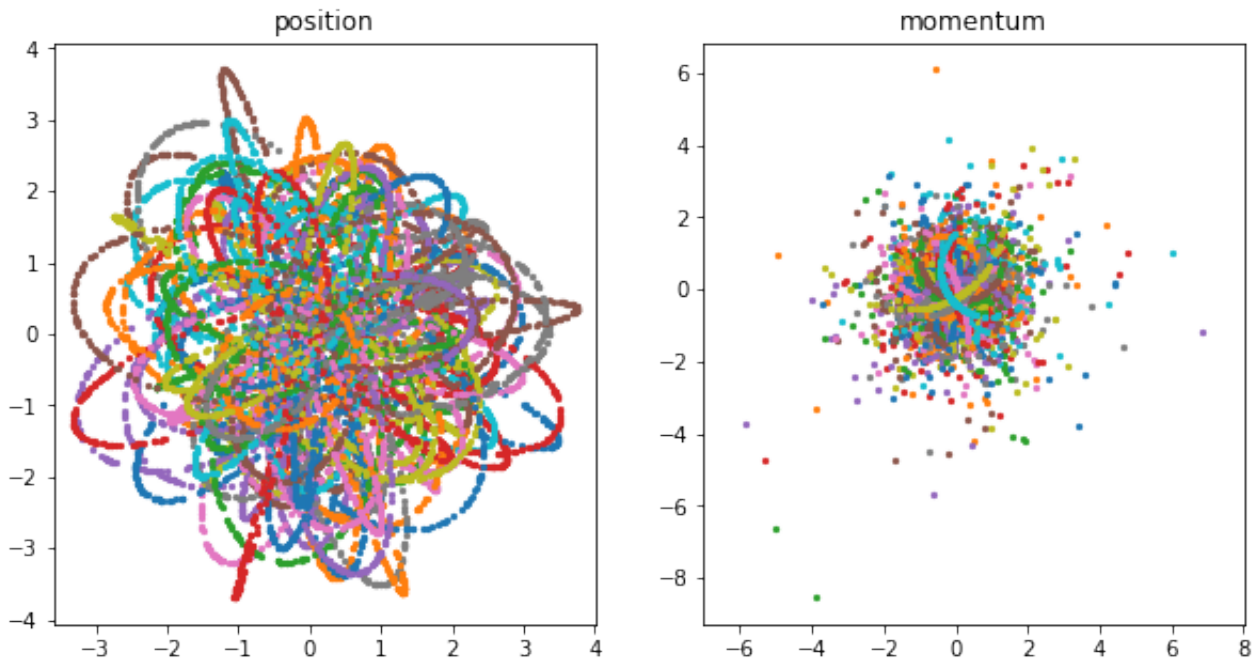


Figure 3: A sample of the generated data in our planet orbits system made of 100 trajectories, consisting of 100 points each, plotted according to position and angular momentum

3 Methods

The Wasserstein Distance

In order to use manifold learning on the set of shapes of orbits, we need some measurement of the distances between these orbits. A good choice of measurement is the Wasserstein distance [10] due to its unique properties. It is a very popular and useful metric that is based on optimal transport. The Wasserstein distance can be defined as the minimum distance that can be crossed from one point to another, or from one function to another, while implementing the least amount of work or effort possible based on continuous probability distribution. This method has been used in machine learning and deep learning applications,

for computer graphics applications, and several other applications [11, 12]. In our case, we are going to calculate the Wasserstein distances between every pair of points in our orbits, and we are going to use them as our input. Therefore, this Wasserstein distance tells us how far apart the orbits of the system are, how the points are distributed on each orbit, and how the points move with respect to each other. The manifold learning methods take this distance and construct a manifold that would be processed differently by each of them.

The Sinkhorn algorithm [13] approximates the Wasserstein distance, using the Sampleloss function [14] found in the Geomloss library [15], according to the following equation:

$$\frac{1}{2}W_2^2 = \min_T \sum_{i,j} T_{ij}C_{ij},$$

$$\sum_i T_{ij} = 1 \qquad\qquad \sum_j T_{ij} = 1,$$

where W_2 is the 2-Wasserstein distance and $T_{ij}C_{ij}$ is the product of the distance and minimum cost (effort) of the optimal transport plan of probabilities between the orbits (points) i and j such that $T \geqslant 0$ (since we cannot have a negative transport plan) and $C_{ij} = \frac{1}{2}||x_i - y_j||^2$. The Wasserstein distance between the orbits can be calculated using the square root of the 2-Wasserstein distance. Then, we fit these distances into a similarity matrix to feed it to the manifold learning methods since a matrix is a very convenient way to represent our data.

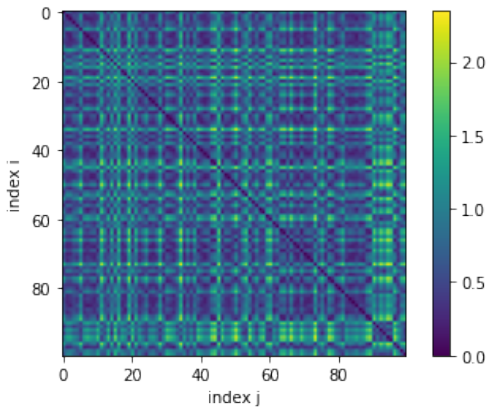


Figure 4: Pendulum dataset similarity matrix between the orbits plotted according to trajectory index with color representing the matrix values of all the Wasserstein distances between each pair of points

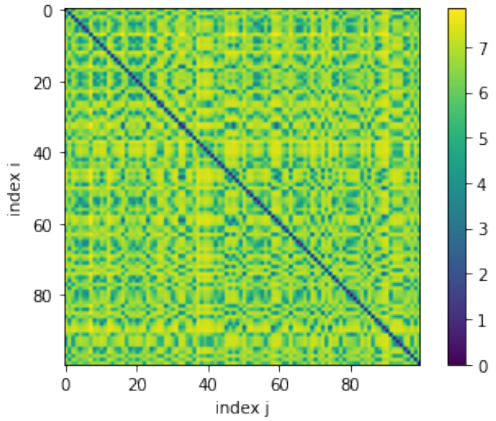


Figure 5: Pendulum image dataset similarity matrix between the orbits plotted according to trajectory index with color representing the matrix values of all the Wasserstein distances between each pair of points

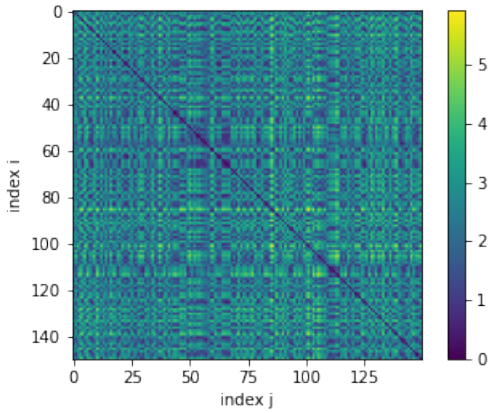


Figure 6: Planet orbits dataset similarity matrix between the orbits plotted according to trajectory index with color representing the matrix values of all the Wasserstein distances between each pair of points

Now that we have the similarity matrices of the distances, we need to extract the low dimensional manifold of the conserved quantities using the manifold learning methods. There is no randomness in these algorithms, and the libraries that we will be utilizing for each method and visualize our results are Numpy [16], Pytorch [17], Scikit Learn [18], and Matplotlib [19].

Primary Non-Linear Dimensionality Reduction Methods

These methods can directly work with the similarity matrix of the precomputed pairwise distances and do not need any extra information about our data. This makes them very easy and reliable methods, and our main comparison will be between these methods.

Spectral Embedding (SE): Spectral embedding is the broad term for diffusion maps or Laplacian eigenmaps, so let us consider diffusion maps as the method we are testing. Diffusion maps takes the manifold constructed from the Wasserstein distance and uses the Gaussian kernel [20] to determine the closest points to each point within a calculated Gaussian bubble around the point and the shortest distance between each point and its closest points, according to the following equation:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{\epsilon}\right),$$

such that k is the Gaussian kernel, and $||x - y||^2$ is the euclidean distance between the two points. This Gaussian kernel enables diffusion maps to build graphs between these points, only considering neighbor points and ignoring far points. It is as if diffusion maps take a walk on the manifold to calculate the shortest distance from one point to another within a specified range. After building the graphs, diffusion maps consider each graph as a spring. Therefore, we now have a vibrating sheet or surface of springs that vibrates in different ways. Then, diffusion maps observe how this sheet can vibrate, and each 'way' of vibration is called a Laplacian mode. Diffusion maps use linear algebra to find the linear transformations of this manifold in order to construct them. We can consider the stiffness k of each spring as the eigenvalue of the Laplacian mode and the direction of vibration of that spring as the eigenvector. Usually, we do not think of eigenvalues and eigenvectors in this manner but in

a more mathematical way such that:

$$A\vec{x} = \lambda\vec{x},$$

where A is the matrix of n coefficients, $\vec{x}$ is the eigenvector of A, and λ is the eigenvalue of the eigenvector. In our case, it is very similar to this interpretation except that we're dealing with springs:

$$A\vec{v_n} = \lambda_n\vec{v_n} \qquad\qquad \vec{F} = -k\vec{x}.$$

So A, here, represents the matrix that tells us how to map the positions that are subjected to the force F, $\vec{v}$ is the mode or pattern of color that tells us the patters of vibrations, n is the number of modes obtained, $\vec{F}$ is the force of the springs, k is the matrix of constants for every point (spring), and $\vec{x}$ tells us the position of every point.

Then, we must choose the least eigenvalues out of the modes, or the vibration modes with the least frequencies, to obtain the greatest amount of information (since diffusion maps do not automatically choose the Laplacian modes). Usually, we choose the first 2 eigenvalues or modes or the first and fourth modes, but we must see which eigenvalues give us the most information about our points in order to plot them according to their color. [21] After identifying the needed Laplacian modes, diffusion maps map out the manifold of conserved quantities. [22]

Multi-Dimensional Scaling (MDS): MDS takes the similarity matrix (all possible Wasserstein distances between all points) and directly compresses all distances into an embedding of points while conserving the distances between these points. So MDS basically picks the points that respect these distances, and thus, a low dimensional embedding is obtained from the higher dimension such that pairwise distances between the points remain the same. However, MDS measures distances in 3D which is not really walking on the manifold

(2D) but just giving its general structure. MDS respects far distances (global method) and takes into account all distances, including the ones between small and large orbits which are less useful. Yet MDS remains a very easy and simple method that is supposed to be sufficient to get what we want and reach our goal. [23]

Isometric Mapping (Isomap): Isomap is basically an improved version of MDS because it converts MDS from a global method to a local one. Isomap uses the Wasserstein distances to create local neighborhood graphs for all points. Therefore, Isomap connects every point with a specific number of neighbors around it, building graphs between these points, and then walks around the edges of the graph. By doing so, Isomap computes approximations of the geodesic distances between these points to find the shortest path along the manifold by just adding very limited euclidean distances. After that, Isomap just runs MDS using the geodesic distances and not directly using the similarity matrix, which stops MDS from calculating the distances in 3D because we already have the distances. Isomap only takes into account the near points and works better than MDS since we have the actual geodesic distances between pairs of points. What we eventually get is a flat manifold of our data. [24]

Secondary Non-Linear Dimensionality Reduction Methods

These methods require an embedding of the data to work properly. If we directly feed them the similarity matrix, they would just consider it as an embedding and would work accordingly. Therefore, these methods are not really reliable. Our aim is to only use and trust the Wasserstein distance, but these methods need additional information. We will provide a euclidean embedding using MDS, and then these methods can process the embedding. It is not sufficient and pointless to test these methods on their own because they need an embedding input, so we will test them with MDS. Additional details and further explanation on these methods are in the appendix.

4 Metrics

To measure how good the embedding produced by each method is, we will use two metrics to evaluate their efficiency and how well the extracted values line up with the actual conserved quantity. We will compare the embedding to the known and expected conserved values using the linear regression test and the Spearman's rank correlation coefficient test to obtain the R^2, ρ, and p-value results. Detailed descriptions of the metrics in the appendix.

5 Results

Pendulum Dataset

We did not restrict the number of components for any method except for t-SNE because t-SNE is sensitive for that parameter. We expect to observe a gradual transition of the color of the points from purple to yellow or from yellow to purple, representing the gradual change in energy between orbits, and distributed on a linear line if the method performed well.

Methods	R^2	ρ	p-value
diffusion maps	0.9845	0.9985	8.471×10^{-126}
MDS	0.9583	-0.9869	1.652×10^{-79}
Isomap	0.9805	0.9929	1.138×10^{-92}
MDS + PCA	0.9832	0.9985	5.680×10^{-126}
MDS + LLE	0.5980	0.9882	9.240×10^{-82}
MDS + t-SNE, n_components=2	0.9686	0.9935	1.323×10^{-94}
MDS + t-SNE, n_components=3	0.3047	-0.5871	1.353×10^{-10}
MDS + UMAP	0.8067	0.8373	1.876×10^{-27}

Table 1: The R^2, ρ, and p-value results of the primary methods

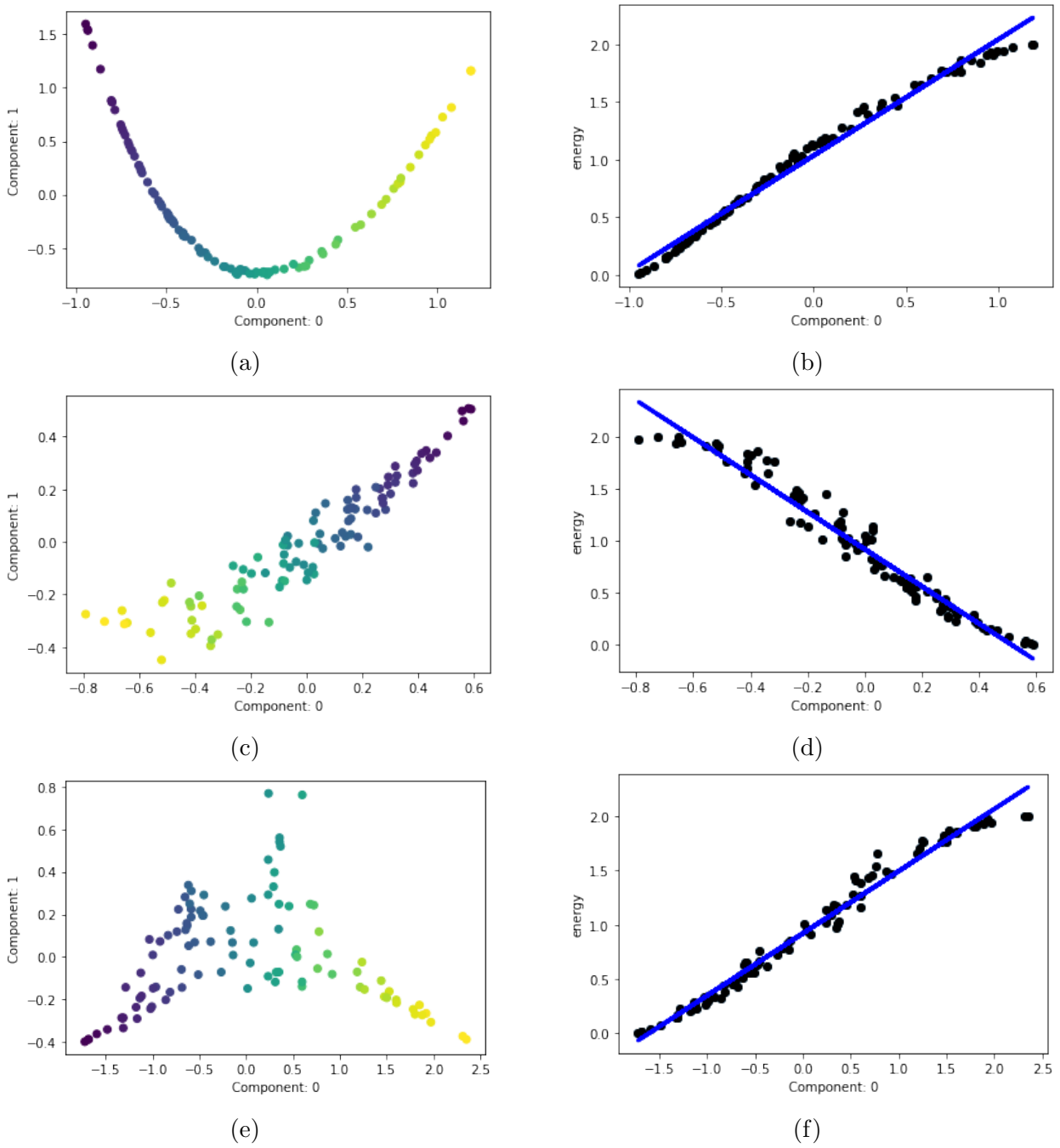


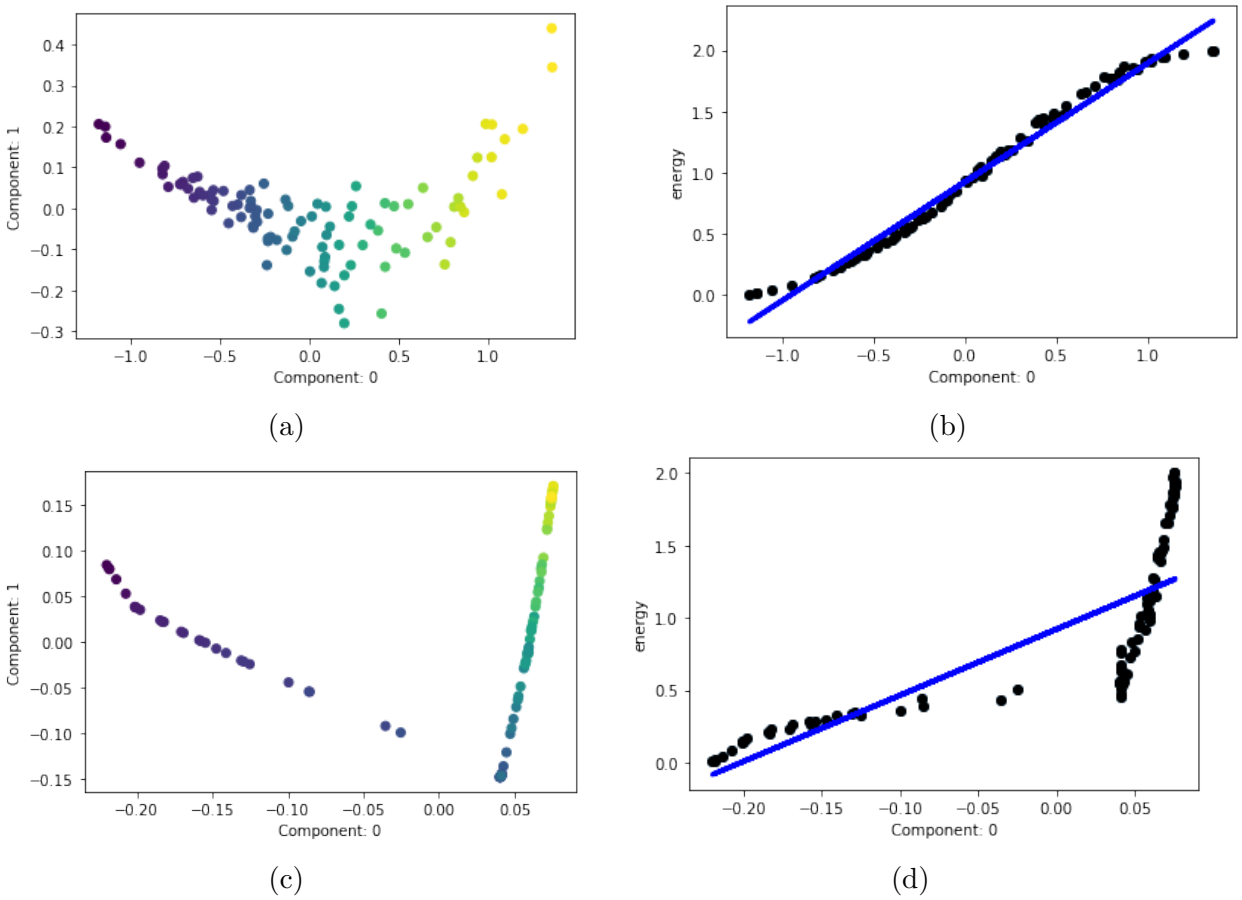
Figure 7: The graphs (a, c, e) contain the first 2 components produced by diffusion maps (a), MDS (c), and Isomap (e) and colored by the energy while the graphs (b, d, f) represent the linear regression results which compare the energy to the first component produced by diffusion maps (b), MDS (d), then Isomap (f) for the pendulum dataset

All the primary methods worked approximately equally well with diffusion maps being

slightly better than MDS and Isomap as judged by the R^2 value. By comparing the R^2, ρ, and p-value results of the methods in Table 1, we can clearly see how well the primary methods performed. There is not much variation in component 1, and most of the information is present in component 0 which tells us that the methods did a great job at extracting the energy. This is evident in the graphs they produced in Figure 7a, Figure 7c, and Figure 7e. Also, the embeddings produced by the primary methods fit fairly linearly to the energy as shown in the results of the linear regression in Figure 7b, Figure 7d, and Figure 7f. This means that diffusion maps, MDS, and Isomap were able to extract the energy successfully.

Upon providing an embedding by MDS, PCA was able to extract the energy and gave pretty good results, both in the Table 1, in the graph it produced in Figure 8a, and in the linear regression test in Figure 8b which shows a linear relationship. Similar to our primary methods, most of the information is present in component 0.

On the contrary, LLE did not perform well. Despite having fairly good results in Table 1, its produced graph and the linear regression test in Figure 8c and Figure 8d show how poorly LLE actually performed in extracting the energy. There is barely a linear relation between its embedding and the real energy. We can sort of see the attempt of embedding the energy in Figure 8c but its eventual failure. Also, LLE gave different results upon running it more than once, making it an unreliable method.



(a) (b)

(c) (d)

Figure 8: The graphs (a, c) contain the first 2 components produced by MDS followed by PCA (a) and MDS followed by LLE (e) and colored by the energy while the graphs (b, d) represent the linear regression results which compare the energy to the first component produced by MDS followed by PCA (b) then MDS followed by LLE (d) for the pendulum dataset

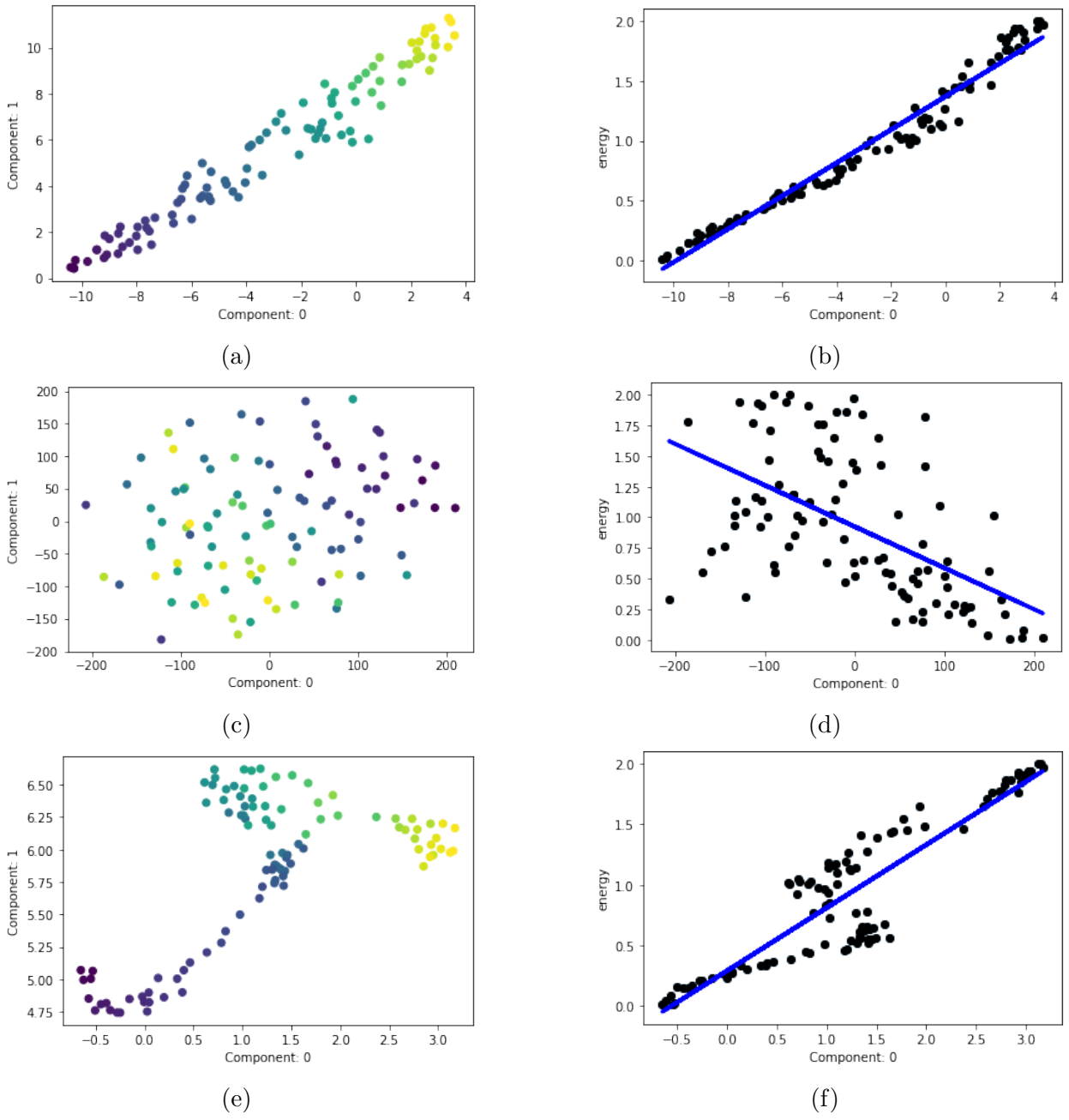


Figure 9: The graphs (a, c, e) contain the first 2 components produced by MDS followed by t-SNE of n_components=2 (a), MDS followed by t-SNE of n_components=3 (c), then MDS followed by UMAP (e) and colored by energy while the graphs (b, d, f) represent the linear regression results which compare the energy to the first component produced by MDS followed by t-SNE of n_components=2 (b), MDS followed by t-SNE of n_components=3 (d), then MDS followed by UMAP (f) for the pendulum dataset

t-SNE gave really great results, both graphically in Figure 9a and Figure 9b and in Table 1, upon setting the number of components (dimensions) to 2. However, upon increasing the number of components to 3, t-SNE's performance dropped, and it did not extract the conserved quantity. We can clearly see the poor performance of t-SNE in the R^2, ρ, and p-value results in Table 1, in the graph it produced Figure 9c, and in the linear regression test in Figure 9d. Therefore, t-SNE does not really work if we do not know the dimension we're reducing to, making it an unreliable method that doesn't work for everything. This is actually an expected result since t-SNE reduces to its given dimension.

Upon using it with MDS, UMAP gave good results in Table 1, in its graph in Figure 9e, and in the linear regression graph in Figure 9f. Just like LLE, UMAP gave different results every time we ran it. Therefore, despite having good R^2, ρ, and p-value results, UMAP does not serve our purpose and is not a dependable method.

Despite having good results, it is unnecessary to join these secondary methods with MDS because MDS works just fine on its own (the secondary methods did not improve MDS's performance noticeably), and the 3 primary methods already performed well. After testing the secondary methods with the simple pendulum dataset, we can exclude them from our next test with the pendulum image and the planet orbits dataset.

Pendulum Image

Methods	R^2	ρ	p-value
diffusion maps	0.9482	-0.9630	1.1528×10^{-57}
MDS	0.1476	0.3414	0.000
Isomap	0.9912	-0.9992	4.866×10^{-139}

Table 2: The R^2, ρ, and p-value results of every primary method

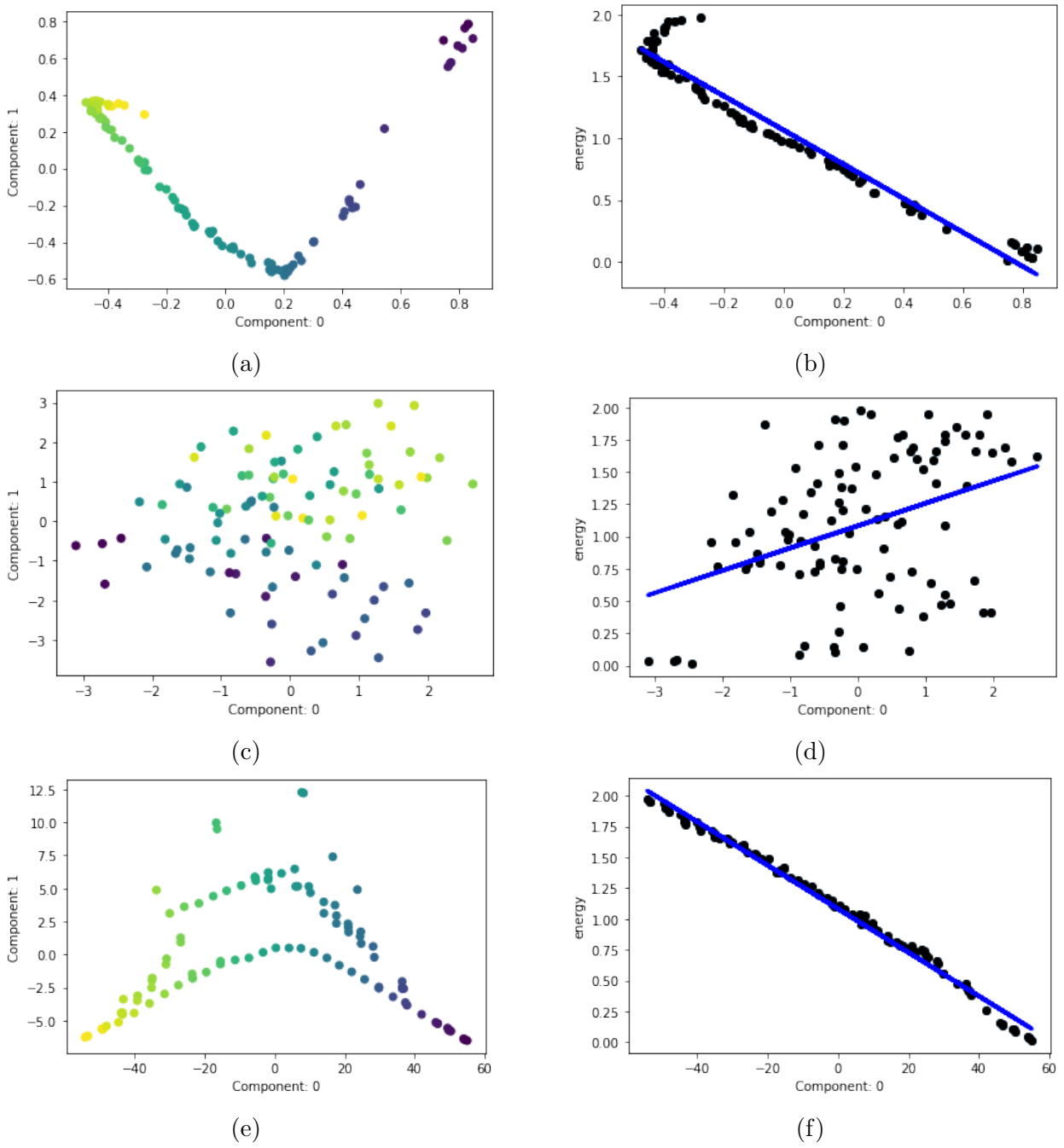


Figure 10: The graphs (a, c, e) contain the first 2 components produced by diffusion maps (a), MDS (c), and Isomap (e) and colored by the energy while the graphs (b, d, f) represent the linear regression results which compare the energy to the first component produced by diffusion maps (b), MDS (d), then Isomap (f) for the pendulum image dataset

After testing the image version of the pendulum, we see that diffusion maps and Isomap did good job at extracting the energy with Isomap having slightly better results both graphically and in Table 2. We can clearly observe their performance in their graphs in Figure 10a and Figure 10e and the linear relationship between their first component and the energy in Figure 10b and Figure 10f. As mentioned previously, most of the information is compacted in the first component which is a good indicator.

MDS did a very bad job and did not extract the energy this time. We can see that in the R^2, ρ, and p-value results in Table 2, in the graph produced which does not gradually change color in Figure 10c, and the very bad fit in Figure 10d.

Planet Orbits Dataset

We expect to observe a gradual transition of the color of the points from purple to yellow or from yellow to purple, representing the gradual change in the conserved quantity between orbits, and distributed on a 2-dimensional manifold if the method performed well.

Diffusion Maps

Conserved Quantity	R^2	ρ	p-value
energy	0.9621	0.9841	3.900×10^{-113}
angular momentum	0.9526	0.9767	7.703×10^{-101}
$\sin\phi$	0.9663	0.9650	5.207×10^{-88}
$\cos\phi$	0.9632	0.9782	4.573×10^{-103}

Table 3: The R^2, ρ, and p-value results of diffusion maps for E, L, and angle ϕ

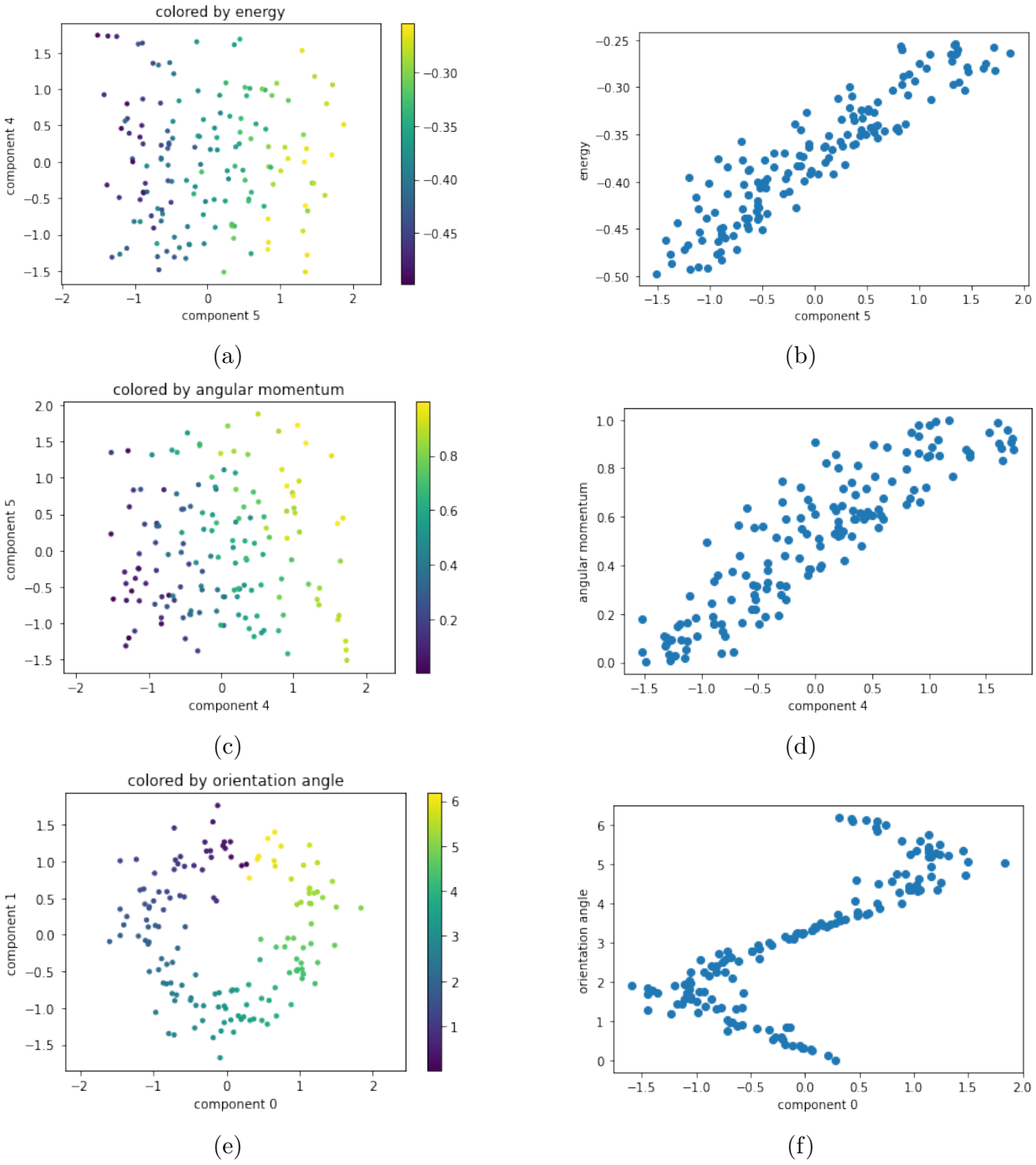


Figure 11: The graphs (a, c, e) contain the 2 chosen components produced by diffusion maps and colored by the conserved quantities while the graphs (b, d, f) represent the linear regression results which compare the conserved quantity to one of the chosen components for the planet orbits dataset

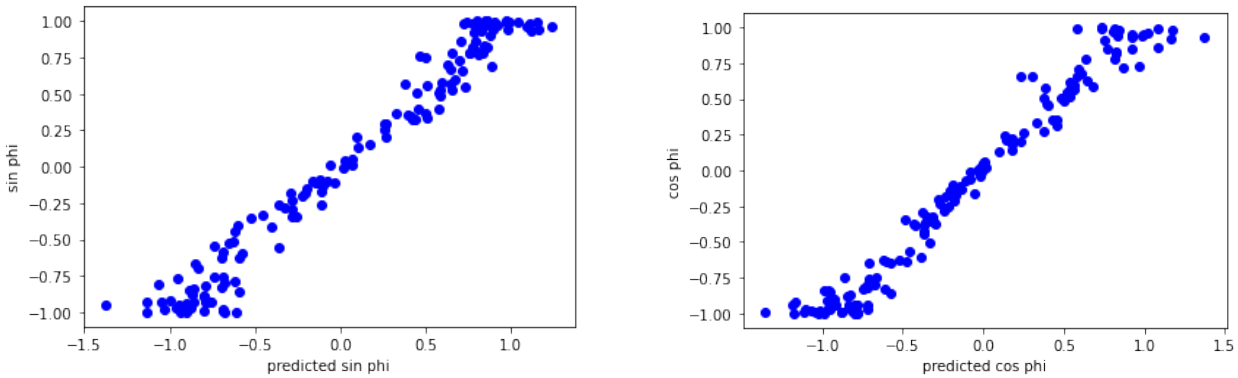


Figure 12: Representing the sine and cosine of the angle ϕ versus the predicted sine and cosine of ϕ produced by diffusion maps

Diffusion maps was able to successfully extract the three conserved quantities in this system. This is evident in diffusion maps' R^2, ρ, and p-value results in Table 3, in the graphs produced by diffusion maps which clearly show the 3 conserved quantities in Figure 11a, Figure 11c, and Figure 11e, and in the linear regression results represented in Figure 11b, Figure 11d, and Figure 12 that show the good linear fit. Diffusion maps was able to pull the energy and angular momentum out into a separate embedding, meaning that it could detect these quantities in higher order modes (Laplacian).

MDS

Conserved Quantity	R^2	ρ	p-value
energy	0.4151	0.6728	4.099×10^{-21}
angular momentum	0.7269	0.8675	9.420×10^{-47}
$\sin\phi$	0.9483	0.9640	4.323×10^{-87}
$\cos\phi$	0.9583	0.9724	1.719×10^{-95}

Table 4: The R^2, ρ, and p-value results of MDS for E, L, and angle ϕ

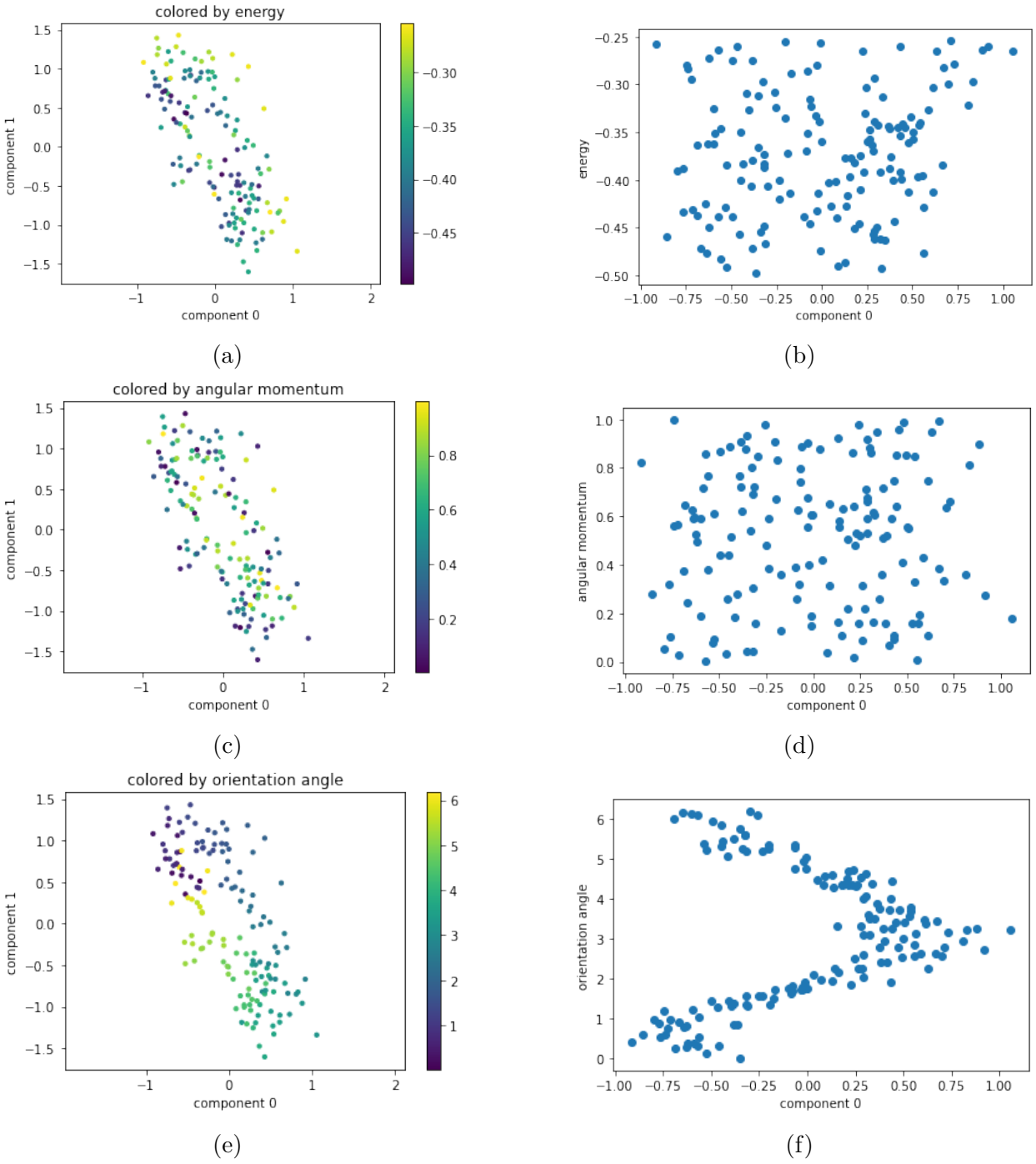


Figure 13: The graphs (a, c, e) contain the 2 chosen components produced by MDS and colored by the conserved quantities while the graphs (b,d,f) represent the linear regression results which compare the conserved quantity to one of the chosen components for the planet orbits dataset

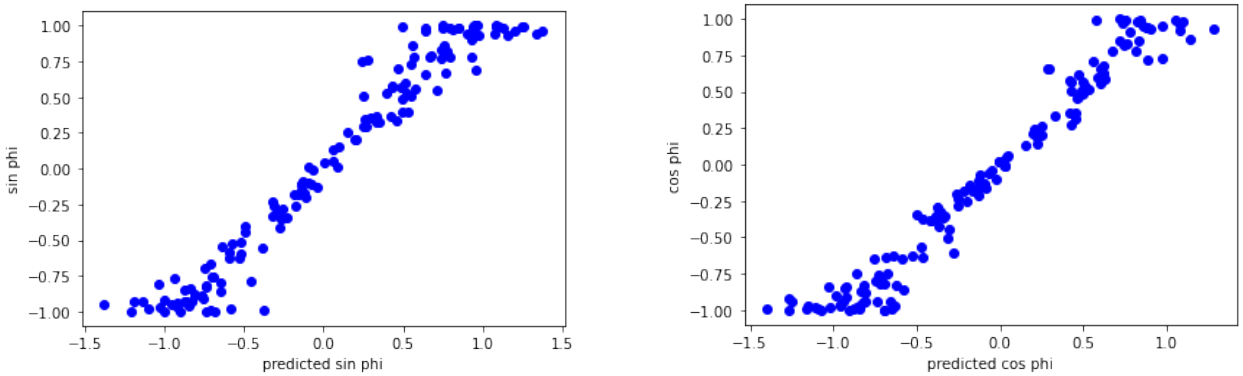


Figure 14: Representing the sine and cosine of the angle ϕ versus the predicted sine and cosine of ϕ produced by MDS

MDS was only able to detect the orientation angle observed in Figure 13e, Figure 13f, and Figure 14 and failed to extract the energy and angular momentum in Figure 13a, Figure 13b, Figure 13c, and Figure 13d. We can see some sort of pattern or combination between energy and momentum in the graphs, meaning that MDS may have recognized the presence of the conserved quantities but didn't embed them well.

Isomap

Conserved Quantity	R^2	ρ	p-value
energy	0.2305	0.4611	2.881×10^{-9}
angular momentum	0.2512	0.5086	3.017×10^{-11}
$\sin\phi$	0.9722	0.9728	5.295×10^{-96}
$\cos\phi$	0.9732	0.9790	3.312×10^{-104}

Table 5: The R^2, ρ, and p-value results of Isomap for E, L, and angle ϕ

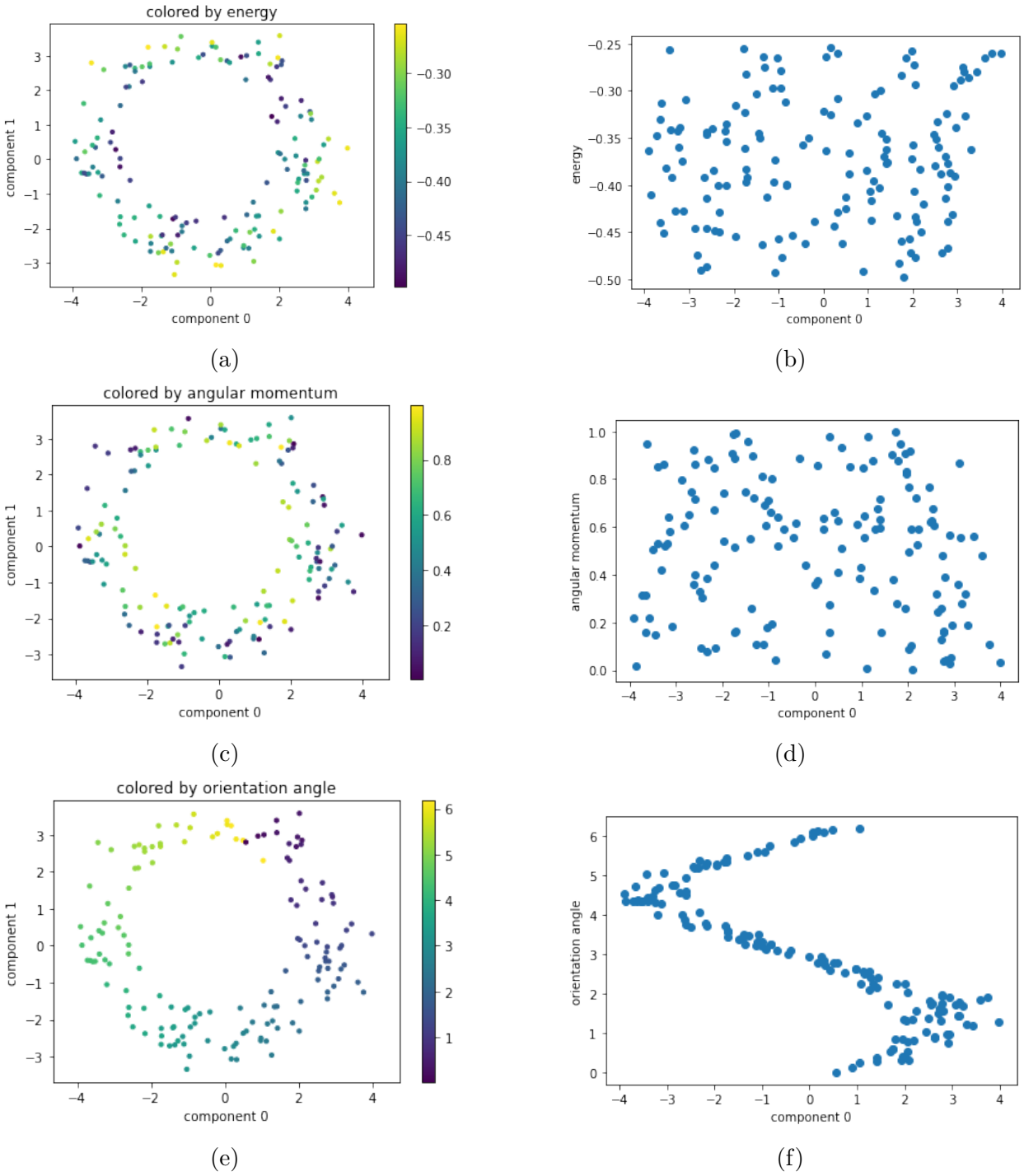


Figure 15: The graphs (a, c, e) contain the 2 chosen components produced by Isomap and colored by the conserved quantities while the graphs (b, d, f) represent the linear regression results which compare the conserved quantity to one of the chosen components for the planet orbits dataset

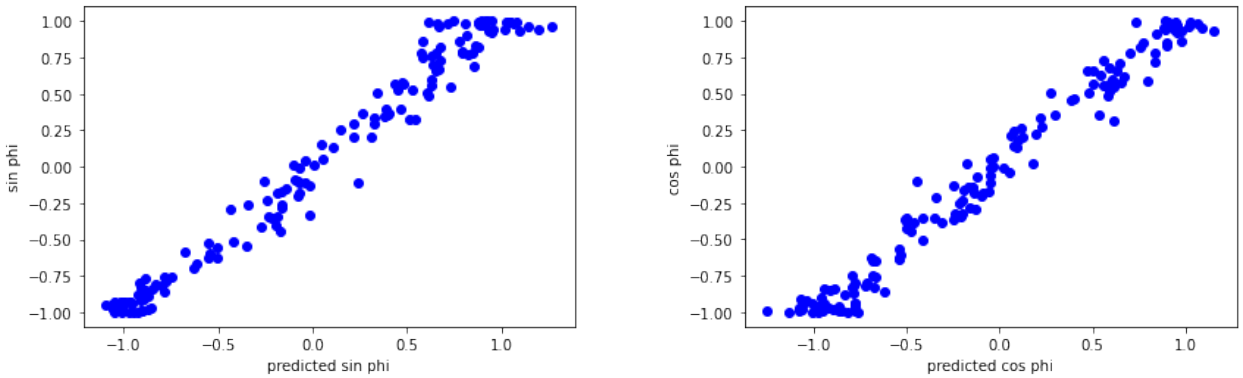


Figure 16: Representing the sine and cosine of the angle ϕ versus the predicted sine and cosine of ϕ produced by Isomap

Similar to MDS, Isomap was only able to extract the orientation angle as observed in Figure 15e, Figure 15f, and Figure 16 and failed to extract the energy and angular momentum in Figure 15a, Figure 15b, Figure 15c, and Figure 15d. We tried changing the number of neighbors parameter, but Isomap did not work. We can see some sort of pattern or combination between energy and momentum in the graphs, meaning that Isomap may have recognized the presence of the conserved quantities but did not embed them well.

As we see in Figure 11a and Figure 11c of the diffusion maps graphs, we chose the fifth and sixth Laplacian modes for energy and angular momentum and the first and second modes for the orientation angle (the third and fourth modes are redundant parameters). Since diffusion maps pushed the energy and momentum to higher eigenvalues, then we can deduce that the change of these quantities is very small. They do not change the orbit of the planet as much as the angle does, causing them to be trickier to extract. To further understand why diffusion maps worked while MDS and Isomap did not, we conducted a small experiment with a toy dataset in the appendix.

6 Discussion

Our results show that spectral embedding (diffusion maps) worked really well and is one of the best manifold learning methods that was capable of extracting all conservation laws. It works with simple, complicated, and images datasets. It worked on the two versions of the pendulum system and the planet orbits system. It was able to work with images and extract three conserved quantities, making spectral embedding a very convenient method to work with.

The Koopman approach [3, 4] requires a model of the physical system and detailed dynamical information about this system, making it a harder and time-consuming method. However, by just plotting the positions and momenta of our systems and computing Wasserstein distances between the orbits, we were able to directly extract the conservation laws without the need for any dynamics. This just shows how easy and convenient spectral embedding can be.

Unlike the parametric siamese neural networks method that was able to extract only one conserved quantity at a time [5], our non-parametric approach, spectral embedding, was able to extract three conserved quantities together. Also, spectral embedding did not require very much tuning. It directly worked upon running it, making it a very robust method that works every time. The AI Poincaré method [6] did not really work with complicated systems and could only count the number of conserved quantities found in a system. Our aim is not to use these methods on simple systems that we already know the conservation laws for, or even count the number of laws, but for complicated systems of unknown laws. That is what spectral embedding has proven to accomplish, making it a very useful method.

7 Conclusion and Future Work

Spectral embedding is the optimal method to directly extract conservation laws because it was able to do so in both the simple pendulum system and the planet orbits one. It is a just a better method overall because it is very robust and does not require a lot of effort to work, unlike previous methods. Since spectral embedding worked with the set of images of the pendulum system, we can basically take a video or several snapshots of a pendulum and use this method to directly extract the conservation of energy, for example.

We can now use spectral embedding for future predictions about the state of a system which could be directly applied in robotics, for example. Spectral embedding can also be applied in various science fields and to further understand those fields. New physical systems can be tested using spectral embedding and possibly new conservation laws can be detected. We can also try applying this method with fluid systems or wave systems.

8 Acknowledgements

Firstly, I would like to express my sincerest gratitude to my mentors Mr. Peter Y Lu and Mr. Rumen R Dangovski for guiding me through these past weeks, teaching me what I needed for this project, and helping me through this journey. I also thank professor Marin Soljačić for all his contributions. I thank my tutor Dr. Jenny Sendova for helping me overcome the problems I faced, her insightful advice and edits, and, most importantly, her greatness. I thank professor Justin Solomon and professor Emma Schmidgall for reading and editing my paper and for their advice. I thank my counselor Angelin Mathew who was always there for me and my fellow Rickoids. I thank Lucy Cai, Ishan Khare, Dimitar Chakarov, Yoland Gao, and Edward Hu for their help and all the effort they put for this program. I also thank my family for being there for me whenever I needed anything. I thank the Research Science

Institute (RSI), Massachusetts Institute of Technology (MIT), and the Center for Excellence in Education (CEE) for giving me this opportunity. Finally, I thank my sponsor Mr. Rafic A. Bizri, the president, CEO, and Chairman of the Board of the Hariri Foundation for his contributions that allowed me to partake this unforgettable experience.

References

[1] D. E. Neuenschwander. *Emmy Noether's wonderful theorem*. JHU Press, 2017.

[2] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[3] A. Mauroy, Y. Susuki, and I. Mezic. Introduction to the koopman operator in dynamical systems and control theory. *The Koopman Operator in Systems and Control*, pages 3–33, 2020.

[4] E. Kaiser, J. N. Kutz, and S. L. Brunton. Discovering conservation laws from data for control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6415–6421. IEEE, 2018.

[5] S. J. Wetzel, R. G. Melko, J. Scott, M. Panju, and V. Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks. *Physical Review Research*, 2(3):033499, 2020.

[6] Z. Liu and M. Tegmark. Ai poincar\'e: Machine learning conservation laws from trajectories. *arXiv preprint arXiv:2011.04698*, 2020.

[7] Y. Ma and Y. Fu. *Manifold learning theory and applications*, volume 434. CRC press Boca Raton, FL, 2012.

[8] P. Lu. Discovering conservation laws using manifold learning and optimal transport. unpublished manuscript, 2021.

[9] P. G. Leach and G. P. Flessas. Generalisations of the laplace–runge–lenz vector. *Journal of Nonlinear Mathematical Physics*, 10(3):340–423, 2003.

[10] S. Vallender. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786, 1974.

[11] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.

[12] N. Zelesko, A. Moscovich, J. Kileel, and A. Singer. Earthmover-based manifold learning for analyzing molecular conformation spaces. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1715–1719. IEEE, 2020.

[13] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[14] P. Christoffersen and K. Jacobs. The importance of the loss function in option valuation. *Journal of Financial Economics*, 72(2):291–318, 2004.

[15] Geometric loss functions between sampled measures, images and volumes — geomloss. `https://www.kernel-operations.io/geomloss/`.

[16] C. R. Harris, K. J. Millman, S. J. der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[20] B. M. ter Haar Romeny. The gaussian kernel. *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematics*, pages 37–51, 2003.

[21] D. Pfau and C. P. Burgess. Minimally redundant laplacian eigenmaps. 2018.

[22] Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural computation*, 16(10):2197–2219, 2004.

[23] M. A. Cox and T. F. Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008.

[24] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[25] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[26] Local linear embeddding. `https://thegreedychoice.github.io/machinelearning/local-linear-embedding`, Jun 2017.

[27] L. K. Saul and S. T. Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

[28] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[29] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[30] G. A. Seber and A. J. Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[31] J. Hauke and T. Kossowski. Comparison of values of pearson's and spearman's correlation coefficient on the same sets of data. 2011.

A Appendix

Secondary Non-Linear Dimensionality Reduction Methods

Principal Component Analysis (PCA): PCA is one of the simplest and oldest manifold learning methods. PCA starts by calculating the center of the plotted data in an orthonormal system, the embedding in our case. Then, PCA shifts the data so that the center is aligned with the origin of the graph while preserving the positions of the points relative to each other. PCA begins to look for the dimensions (components) with the largest variants and the biggest spread then arranges the dimensions in a decreasing order according to their importance and presence of information such that the first dimension is the the dimension with the largest variant, the second dimension is the one with the second largest variant, etc. Therefore, all PCA does is rotate and project the input according to the largest varying components. It cannot unfold or stretch a manifold, so we do not expect it to perform really well. PCA also does a bad job at projecting non-linear manifolds (if the manifold is curved, it will not work), yet works really well if the manifold is flat. [25]

Locally Linear Embedding (LLE): Just like Isomap, LLE builds graphs between points, connecting each point to a specific number of neighbors. It then tries to preserve the non-linear global structure by working with local linear patches on the manifold and then later combines all its findings to build a global presentation or structure. [26] Therefore, LLE basically finds a flat approximation of the manifold and tries to embed it into a lower dimension. LLE can be possible, but it's probably not a very ideal way because it would require a lot of effort to work properly and flatten the manifold. [27]

Clustering Methods: Clustering methods, such as t-SNE (t-distributed Stochastic Neighbor Embedding) [28] and UMAP (Uniform Manifold Approximation and Projection) [29],

use an optimisation algorithm to cluster similar points, forcing the given data into the dimension we want. Therefore, these methods are not actually designed to target our goals but are used for visualizing data and mostly in biology. The problem with these methods is that they do not preserve the structure of our data and could cluster points that should not be clustered (imagined clusters) just to compress it to the dimension we want. We do not expect them to perform well.

Metrics

Linear Regression Test

Linear regression is a correlation test that performs a regression task. We are going to use linear regression to compare the embedding calculated by each method to the real conserved quantity in our orbits. Linear regression takes the embedding and predicts the conserved quantity then finds the linear relationship between them if present where the conserved quantity is the dependant variable value y and the embedding (or outcome of the methods) is the given independent variable x. Linear regression tries to fit the input x and output y onto a line according to this relationship. Then, we can extract the R^2 coefficient, or score, of each test performed on each method. The R^2 coefficient is the correlation coefficient that tells us how well the data points fit on this line, thus telling us how well each method performed. If $R^2 \approx 1$ then the method performed well and the results of the embedding and the conserved quantity are correlated, but if $R^2 \approx 0$ then the method performed poorly and the results and the conserved quantity are not correlated. [30]

Spearman's Rank Correlation Coefficient Test

The Spearman's rank correlation coefficient is a statistical test that tells us how well we can predict this y if we have our x, basically evaluating how well the relation is by comparing

the predicted value to the real value of the conserved quantity. This test also tells us how monotonic our function is. The Spearman coefficient ρ is a measure of rank correlation and the value that we need to know how good the correlation is ($-1 < \rho < 1$). If $\rho \approx 1$ or $\rho \approx -1$ then the relation is monotonic while if $\rho \approx 0$ then the relation is not monotonic. In addition to the coefficient, this test gives us the p-value which tells us the probability of getting uncorrelated or correlated by chance/accident results. In this case, if the p-value$< 3 \times 10^{-7}$ then there is an actual correlation between the results of the embedding and the conserved quantity. However, if the p-value$> 3 \times 10^{-7}$ then our results are not actually correlated, and if the graph shows a good correlation then it is just due to chance/accident. [31]

Test with Toy Dataset

We generate a dataset of 1000 scattered points, making the shape of a ring to simulate a similar case as the orbits dataset, as shown in the following figure:

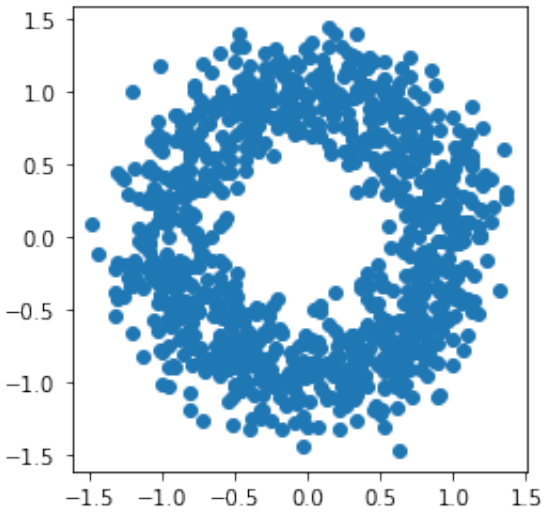
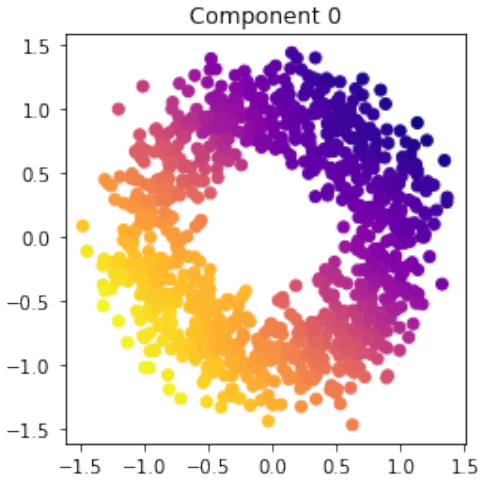


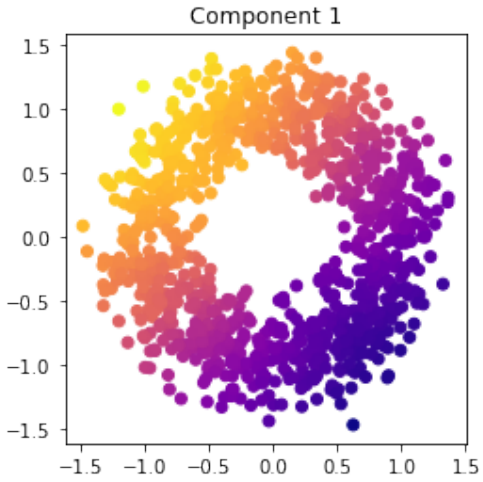
Figure 17: Plotting of the toy dataset having a ring shape

The quantities that should be detected by the methods are the angle and the radial direction. This dataset is very similar to the planet orbits one since we can interpret the angle

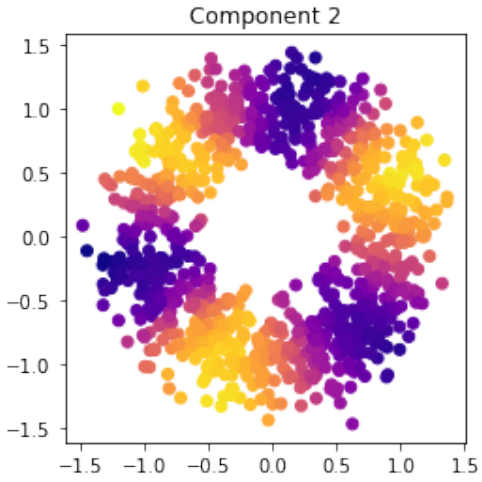
(dominant variable) as the orientation angle and the radial direction (harder to extract) as the energy and the angular momentum. We then perform Isomap and Spectral Embedding (diffusion maps) on our dataset (we didn't test MDS because it would give similar results to Isomap). We plot the dataset and color it with the components produced by the methods to visualize what is happening in each component and what each method was able to catch.



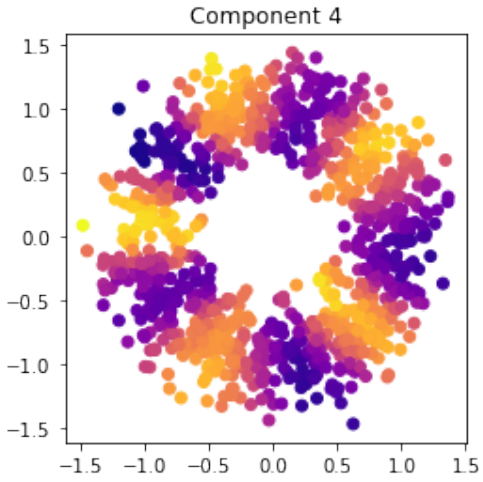
(a)



(b)

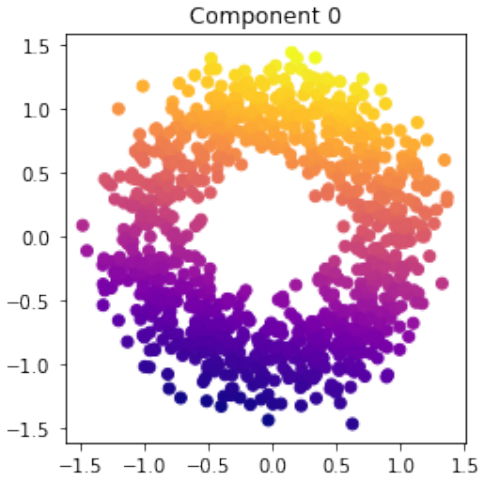


(a)

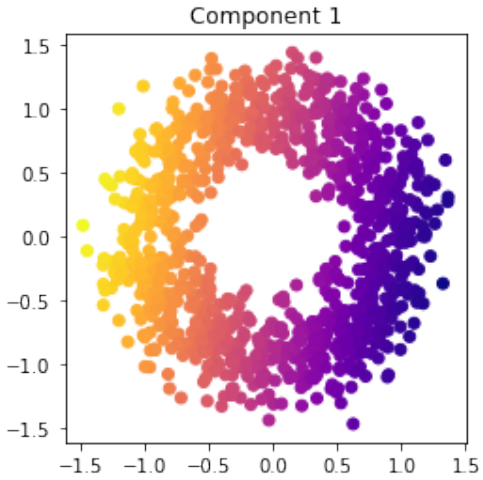


(b)

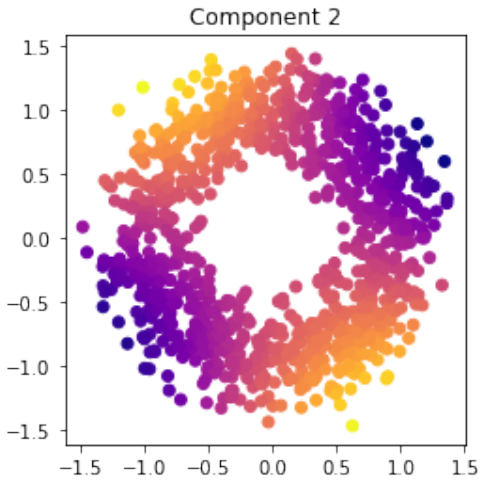
Figure 19: Graphs of toy dataset colored by the embedding produced by Isomap



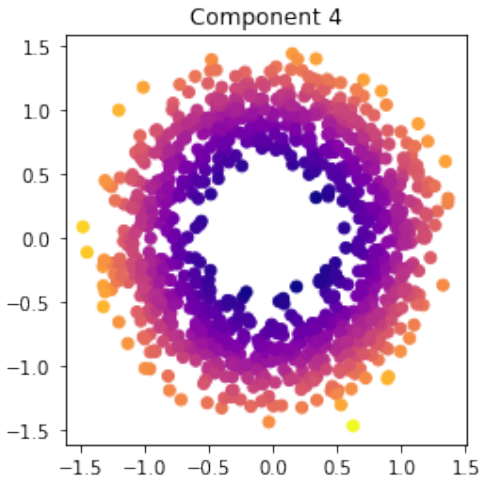
(a)



(b)



(a)



(b)

Figure 21: Graphs of toy dataset colored by the embedding produced by Spectral Embedding

We can see that components 0 and 1 for both methods are almost the same and detect the angle in Figure 18a, Figure 18b, Figure 20a, and Figure 20b. However, as we increase the number of components for Isomap to plot the other dimensions (more than the first 2), we only obtain redundant components of the angle as shown in Figure 19a and Figure 19b which don't give any new or useful information. However, if we look at the components produced by spectral embedding, we have 2 redundant components, 2 and 3, that can be represented

by Figure 21a, similar to Isomap, and then a component 4 which was able to directly embed the radial direction as observed in Figure 21b. This means that diffusion maps was able to detect these quantities in higher order modes (Laplacian).