A NUMERICAL INVESTIGATION OF LAPLACIAN EIGENMAPS

BOBITA ATKINS, ASHKA DALAL, NATALIE DININ, TESS MCGUINNESS

1. How to Construct an Eigenmap from a Set of Points

We consider a finite set of points and suppose that two points, x_i and x_j, are connected by an edge if the distance, d, between them is less than some given ε. The edges of the resulting graph can be embedded into a Laplacian Matrix, L, given by $L = D - W$ where W is the adjacency matrix and D is the degree matrix defined as follows.

We define connections between points to have weights w_{ij}.

$$w_{ij} = \begin{cases} 1 \text{ if } d(x_i - x_j) \leq \varepsilon, i \neq j \\ 0 \text{ else} \end{cases}$$

This graph is undirected, so $w_{ij} = w_{ji}$. Then we write the adjacency matrix, W, $W = (w_{ij})_{i,j=0}^{n-1}$.

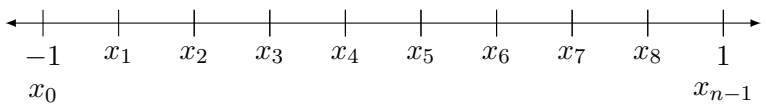
Let the degree matrix, D be defined as a mtrix where the diagonal elements are row sums of the adjacency matrix. That is,

$$D_i = \sum_{j=0}^{n-1} W_{ij}.$$

Calculate the eigenvalues and eigenvectors of the Laplacian Matrix. Call each eigenvalue and eigenvector pair λ_i and v_i respectively. Sort the eigenvalues such that $0 = \lambda_0 < \lambda_1 < \lambda_2 \ldots \lambda_n$. We then select a finite number of eigenvectors corresponding to non-zero eigenvalues and plot the vectors against each other. This plot is known as the eigenmap or eigencoordinates (these terms will be used interchangeably throughout the paper).

2. Evenly Spaced Points

We begin by choosing evenly spaced points off the interval $[-1, 1]$ as follows.



n is arbitrary, but for the sake of simplicity in the above diagram, we set $n = 10$

Note that we select n points, each a distance $\delta = \frac{2}{n-1}$ apart from each other. We also set an $\varepsilon = k\delta$ where $k = 1, \ldots, \lfloor \frac{n-1}{2} \rfloor$. Note that k essentially denotes the number of points that a given point is connected to on a single side. Depending on the value of n and k that we select, we construct eigencoordinates based on points that are evenly spaced in the interval $[-1, 1]$. Below, we display some pictures that we have generated from eigencoordinates based on points that are evenly spaced in the interval $[-1, 1]$ and compare them to Chebyshev polynomials.

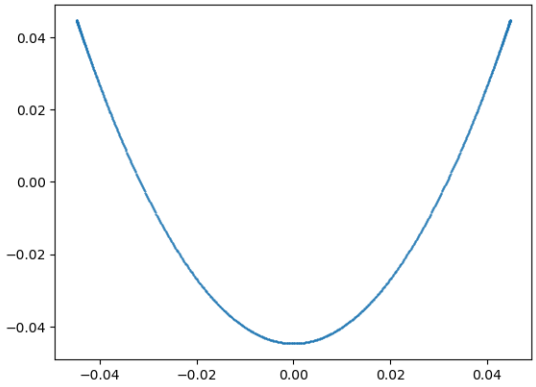


FIGURE 1. eigencoordinates for $n = 1000$, $k = 1$

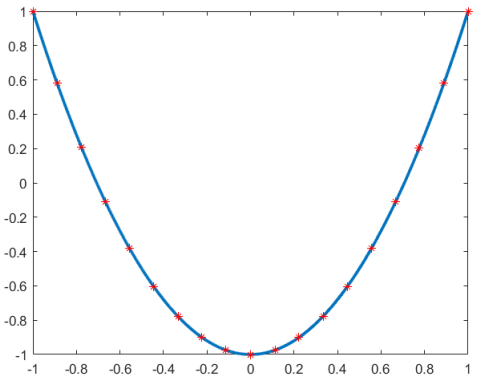


FIGURE 2. In blue: eigencoordinates for $n = 1000$, $k = 1$.
Red stars: Chebyshev polynomial of the first kind

3. Arranged According to a Distribution

3.1. **Evenly Spaced Points Arranged According to a Distribution.** A variation on this idea is to consider evenly spaced points arranged according to different distributions. We take n evenly spaced points with a chosen value for ε. Then we take the points and input them into either the inverse cumulative distribution function of the Gaussian or the inverse cumulative distribution function of the exponential

distribution.

Inverse CDF of the Gaussian distribution is:

$$\mu + \sigma\sqrt{2}\,\mathrm{erf}^{-1}(2p-1)$$

Inverse CDF of exponential distribution is:

$$-\log\left(1-\frac{i}{n}\right), i = 0, 1, ..., n-1.$$

We remove points outside $[-2, 2]$ for the Gaussian distribution and outside $[0, 5]$ for the exponential distribution. In both cases, we then construct the Laplacian matrix and compute its eigenvalues and eigenvectors. We normalize the eigenvectors by dividing each eigenvector by maximum element. We then plot the normalized eigenvectors corresponding to the first two non-zero eigenvalues.

Below are some eigencoordinate pictures generated from n points selected according to either the Gaussian distribution or the exponential distribution.

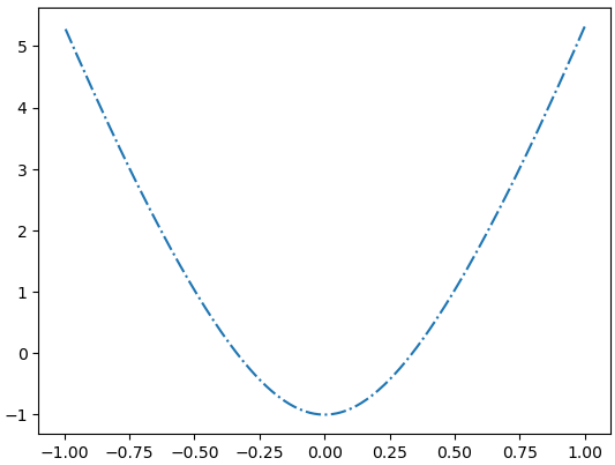


FIGURE 3. Evenly spaced points selected according to the Gaussian Distribution with $n = 5000, \varepsilon = .1$

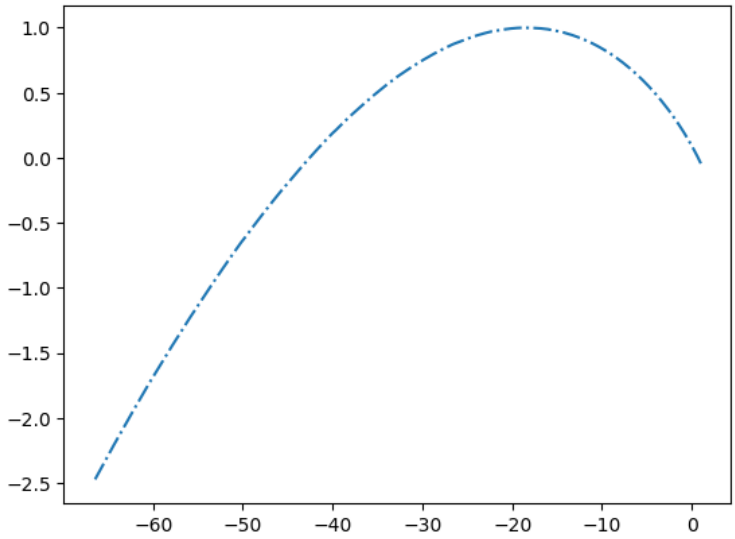


FIGURE 4. Evenly spaced points selected according to the Exponential Distribution with $n = 5000, \varepsilon = .1$

3.2. **Uniformly Random Points arranged according to a distribution.** Another variation on this idea is to consider n points randomly selected from the uniform distribution inputted into the inverse cumulative distribution functions of both the Gaussian distribution and the exponential distribution. Below are some eigencoordinate pictures generated from n points selected from a random uniform distribution of points according to either the Gaussian distribution or the exponential distribution.

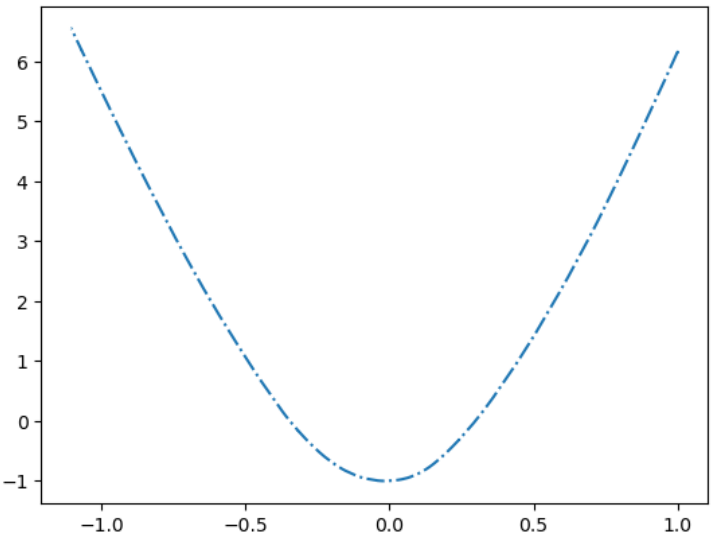


FIGURE 5. Uniformly Random Points arranged according to the Gaussian distribution with $n = 5000, \varepsilon = .1$

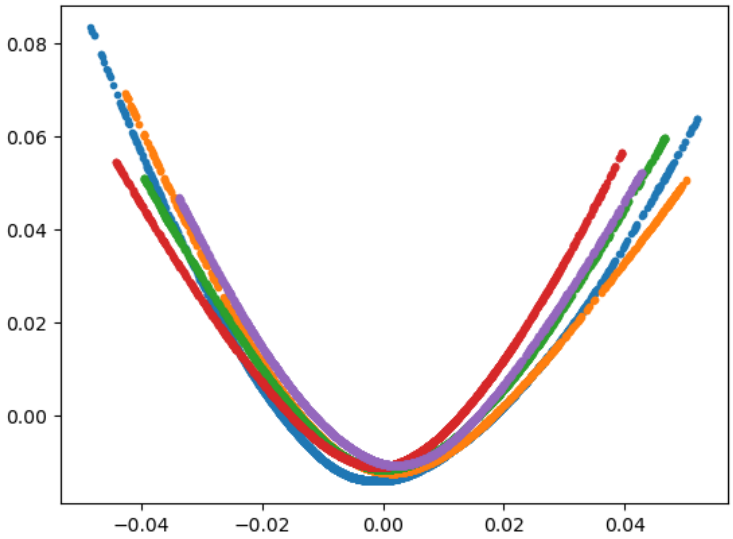


FIGURE 6. Uniformly Random Points arranged according to the Gaussian distribution with $n = 5000, \varepsilon = .1$, 5 runs

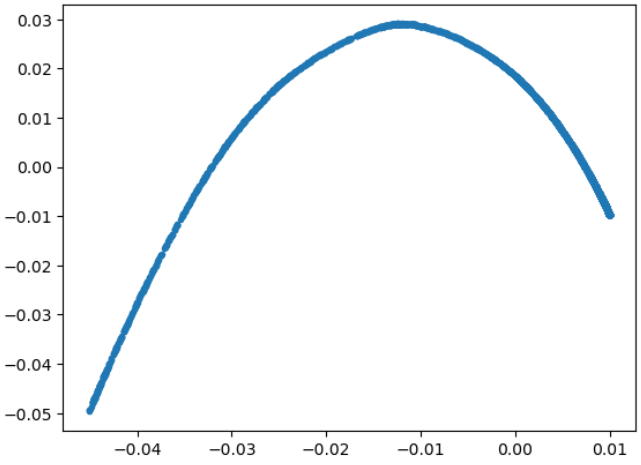


FIGURE 7. Uniformly Random Points arranged according to the exponential distribution with $n = 5000, \varepsilon = .1$

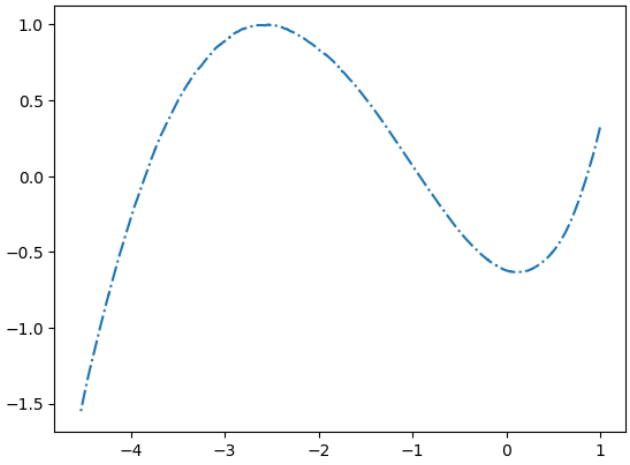


FIGURE 8. Uniformly Random Points arranged according to the exponential distribution with $n = 5000, \varepsilon = .1$, 1st and 3rd eigenvectors plotted

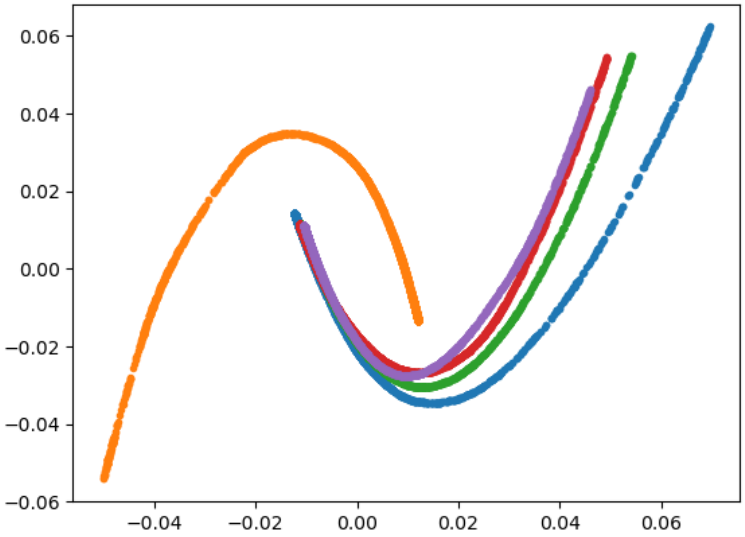


FIGURE 9. Uniformly Random Points arranged according to the exponential distribution with $n = 5000, \varepsilon = .1$, 5 runs

In the next section, we look at random points sampled directly from different distributions on intervals.

4. Random Points from an Interval

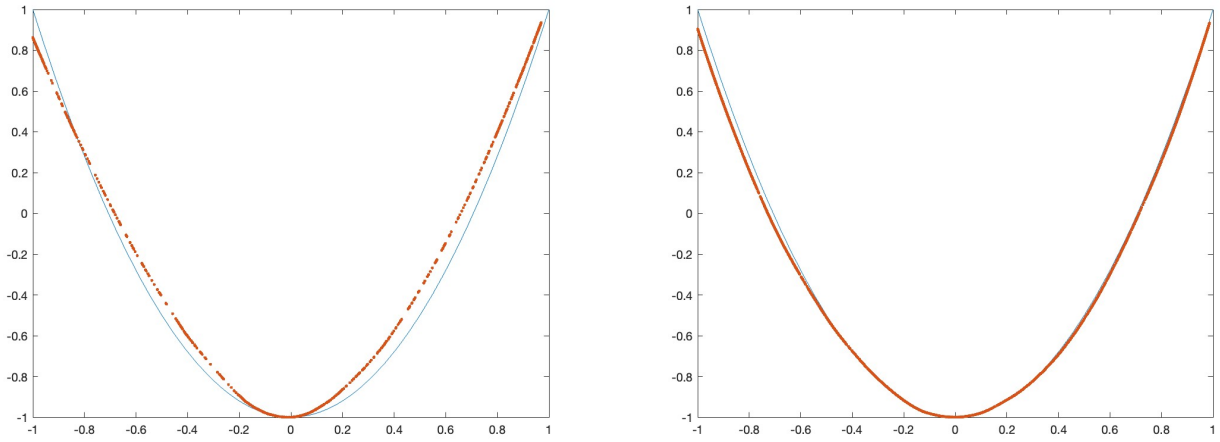
We begin by picking n randomly selected points from a distribution on a given interval. After sorting this vector of random points, and choosing a value of ε, we construct the Laplacian matrix and compute the eigenvalues and eigenvectors. We normalize the eigenvectors by dividing each eigenvector by the absolute value of its largest element, so that the greatest value in each eigenvector is now 1. We then plot the

normalized eigenvectors corresponding to the first two non-zero eigenvalues. As stated in Section 1, this is an eigencoordinate graph. It is important to note that when plotting the eigencoordinates, we multiply the second eigenvector by the sign of its first element so that the curve is positive. We apply this process to the uniform, gaussian, and exponential distributions, and explore the relationships between n, ε, and various polynomials below.

4.1. **Uniform.** When randomly selecting points from the uniform distribution, we chose points on the interval $[-1, 1]$.

Theorem 4.1. *As $n \to \infty$ and $\varepsilon \to 0$, the Laplacian eigencoordinates of uniformly distributed random points converge to the Chebyshev polynomial.*

We start with $n = 1000$ points and $\varepsilon = 0.05$, and plot the eigencoordinates against the quadratic Chebyshev polynomial (seen in Figure 10a). As we increase n and decrease ε, the eigencoordinates closely follow the Chebyshev polynomial (seen in Figure 10b).



(A) $n = 1000, \varepsilon = 0.05$ (B) $n = 5000, \varepsilon = 0.01$

FIGURE 10. Laplacian eigencoordinates plotted against the Chebyshev polynomial (blue) $T_2(x) = 2x^2 - 1$.

Since we are taking random points, the curves change slightly each time. To ensure the most accurate results, we plotted multiple runs of the eigencoordinates for a chosen n, ε on one graph.

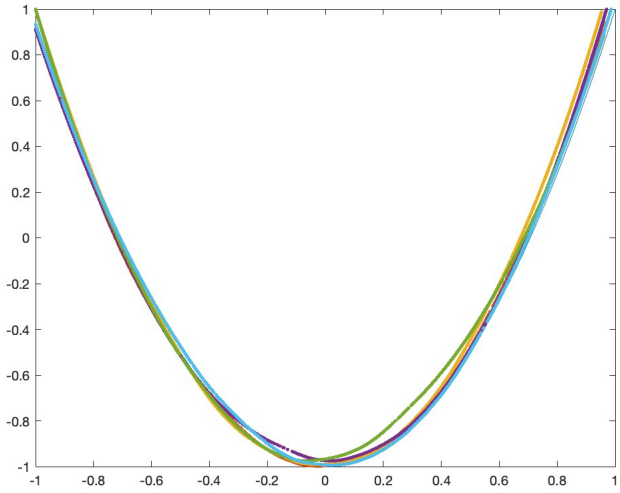
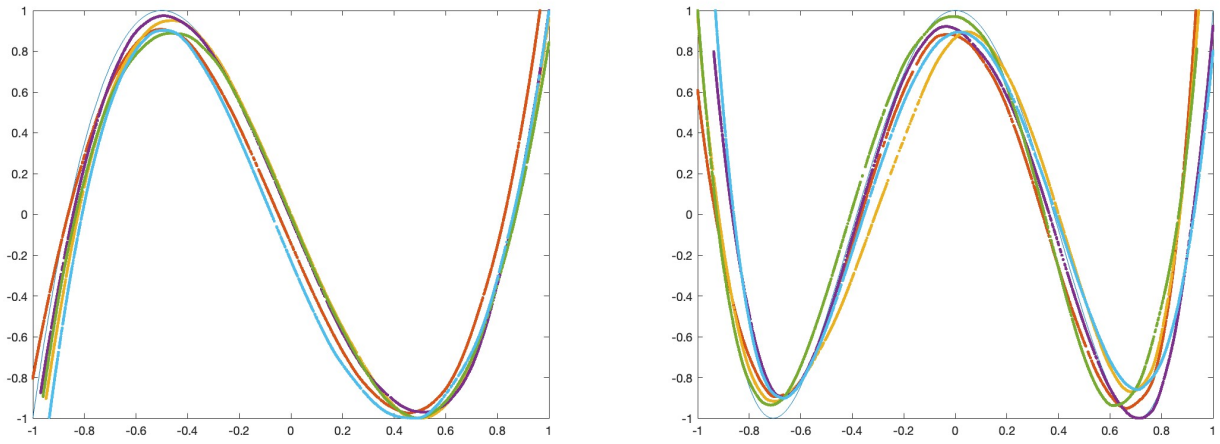


FIGURE 11. $n = 5000, \varepsilon = 0.01$, 5 runs.

Recall that the previous eigenmaps have been created with the first and second eigenvectors corresponding to the first two non-zero eigenvalues. We can do the same for the first and third, and first and fourth eigenvectors, and plot them against the cubic and quartic Chebyshev polynomials.



(A) 1st and 3rd eigenvectors, $T_3(x) = 4x^3 - 3x$

(B) 1st and 4th eigenvectors, $T_4(x) = 8x^4 - 8x^2 + 1$

FIGURE 12. Laplacian eigenmaps of $n = 5000, \varepsilon = 0.01$, 5 runs.

4.2. **Gaussian.** When randomly selecting points from the Gaussian distribution (also known as the Normal Distribution), in order to avoid the situation where outliers cause our graphs to be disconnected, we removed points outside of the interval $[-2, 2]$.

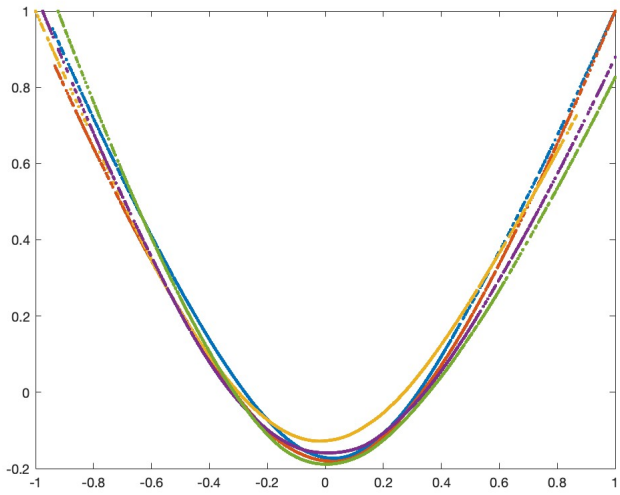


FIGURE 13. Eigenmap of normally distributed random points, $n = 5000, \varepsilon = 0.08$, 5 runs.

Note: We expected the eigencoordinates of normally distributed random points to converge to the Hermite polynomials. However, when plotting the first and third or fourth eigenvectors, we see that the eigencoordinates loosely follow the Legendre polynomials. Question: Is there theory behind this? Could this be happening perhaps because of our scaling/normalization?

Below are some eigencoordinate pictures generated from n points selected randomly according to a Gaussian distribution, plotted with the polynomial $\frac{7}{6}x^2 - \frac{1}{6}$ which we suspect to be the curve of best fit.

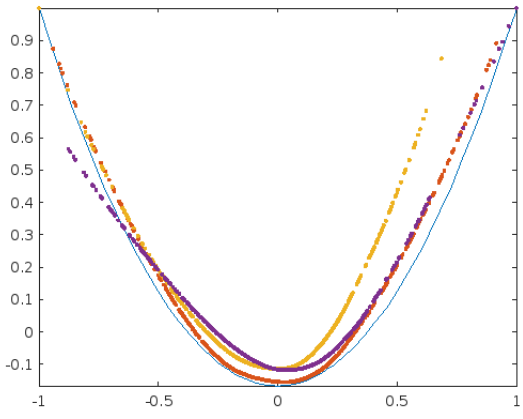


FIGURE 14. $n = 1000, \varepsilon = 0.3$, polynomial $= \frac{7}{6}x^2 - \frac{1}{6}$

We attempted to find a relationship between n and ϵ

4.3. **Exponential.** In order to avoid the situation where outliers cause our graphs to be disconnected, we removed points outside the interval $[0, 5]$. Below are some eigencoordinate pictures generated from n points selected randomly according to an Exponential distribution.

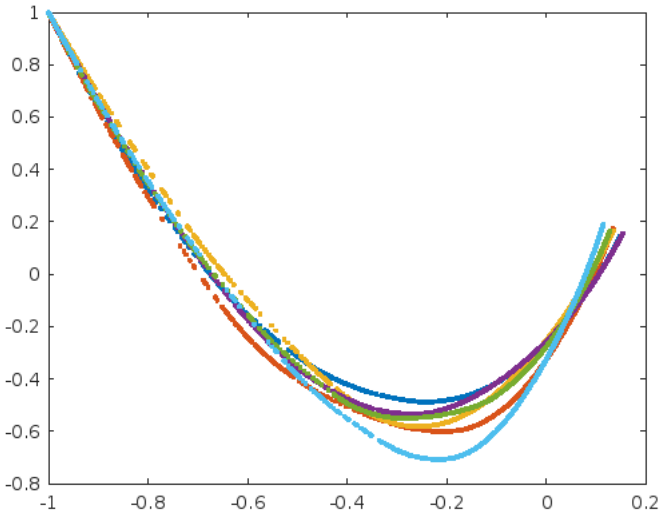


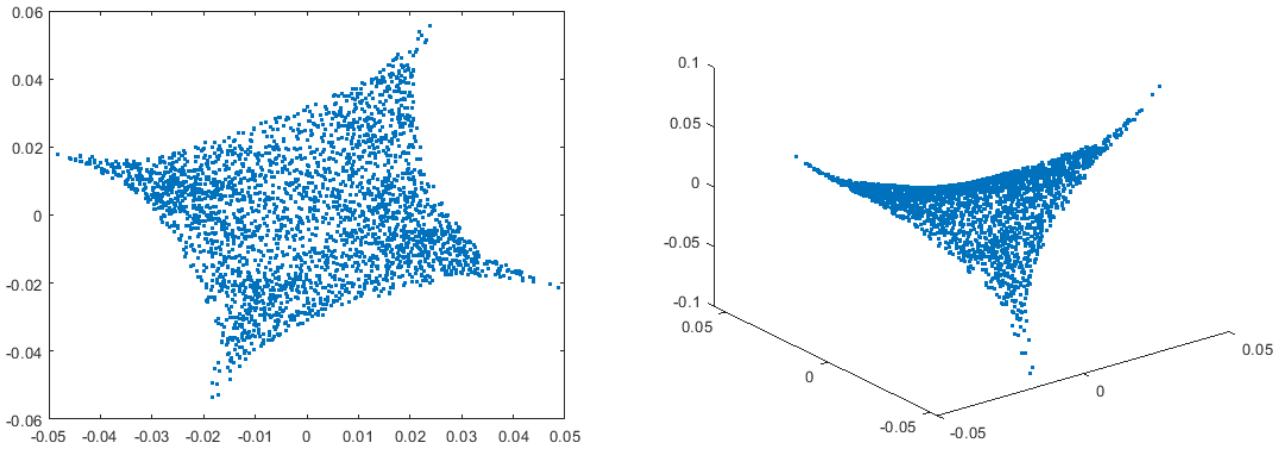
FIGURE 15. $n = 4000, \varepsilon = 0.1$

5. Random Points from a Shape

5.1. **Square.** When choosing points from the square we decided to choose from the unit square. That is, we choose points from $[0, 1] \times [0, 1]$. Thus, we randomly select an x value and a y to be the coordinates of the point p_j. Then, our weight is given by

$$w_{ij} = \begin{cases} 1 \text{ if } d_E(p_i, p_j) \leq \varepsilon, i \neq j \\ 0 \text{ else} \end{cases}$$

where $d_E(x_i, x_j)$ denotes the Euclidean metric. After completing the process of generating and plotting the eigenmap, the resulting plot appears as a hyperbolic square (seen in Figure 16a). If we plot a three-dimensional eigenmap where we plot the first three eigenvectors against each other (rather than the first two) we see a saddle-like surface (seen in Figure 16b).



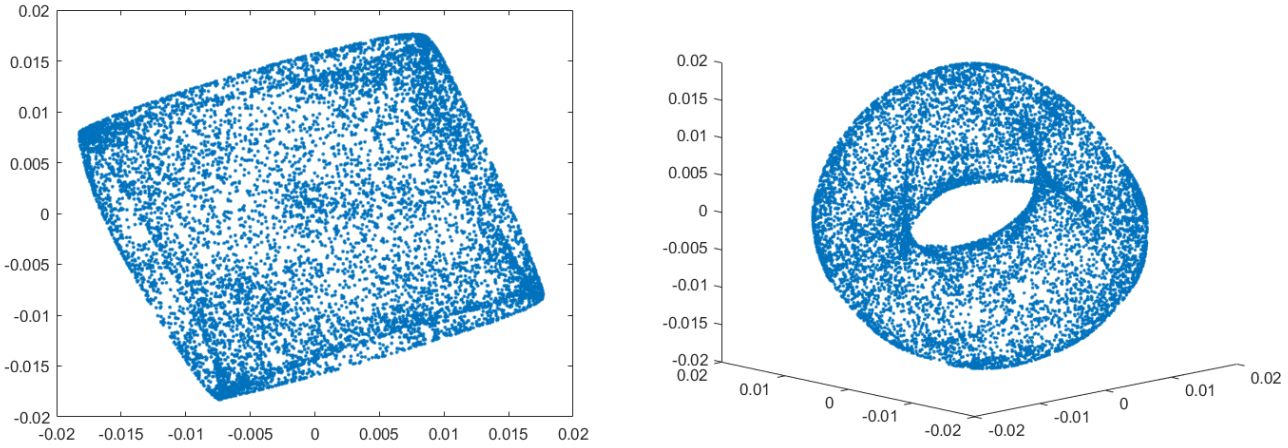
(A) Two-dimensional eigenmap. (B) Three-dimensional eigenmap.

FIGURE 16. The 2-dimensional and 3-dimensional eigenmaps for a points sample from a square where $n = 3000$ and $\varepsilon = 0.7$

5.2. **Torus.** The process for mapping points from the torus is almost identical to the process for a square. The primary difference is the metric that is used. Rather than the standard Euclidean metric, the distance between two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ is defined by

$$d(p_i, p_j) = \sqrt{(\min(|x_i - x_j|, 1 - |x_i - x_j|))^2 + (\min(|y_i - y_j|, 1 - |y_i - y_j|))^2}.$$

When plotting the eigenmap of the resulting Laplacian matrix, we see the



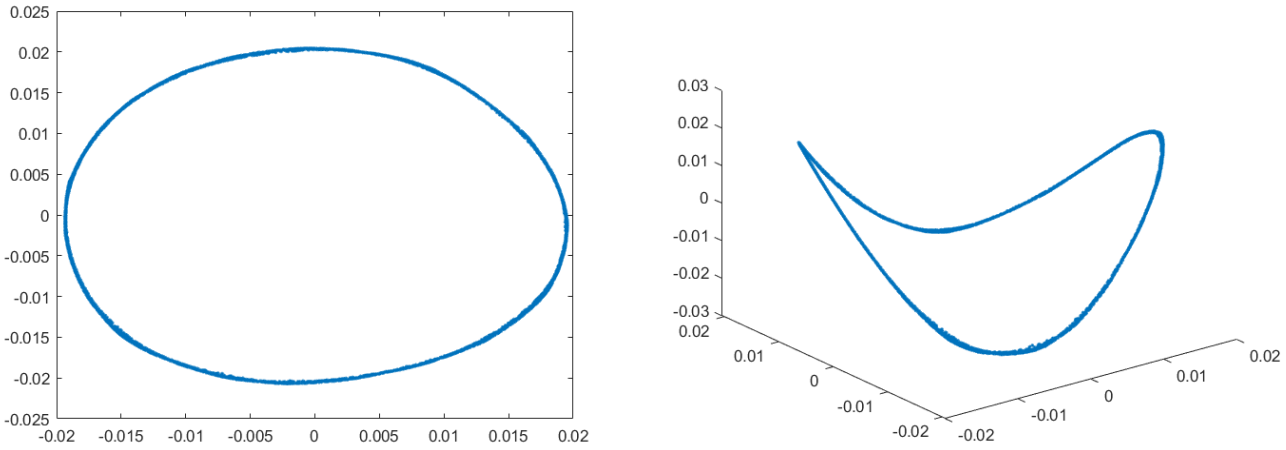
(A) Two-dimensional eigenmap. (B) Three-dimensional eigenmap.

FIGURE 17. The 2-dimensional and 3-dimensional eigenmaps for a points sample from a torus where $n = 10000$ and $\varepsilon = 0.5$

5.2.1. *Thin Torus.* For a thin torus, we further adjust the metric. The distance between two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ is defined by

$$d(p_i, p_j) = \sqrt{(\min(|x_i - x_j|, 1 - |x_i - x_j|))^2 + (\min(|y_i - y_j|, c - |y_i - y_j|))^2}$$

where $c \in Z$ is the dimension of the torus. In our code, $c = 5$ so the resulting torus is generated by way of a 1×5 rectangle.



(A) Two-dimensional eigenmap.

(B) Three-dimensional eigenmap.

FIGURE 18. The 2-dimensional and 3-dimensional eigenmaps for a points sample from a thin torus where $n = 5000$ and $\varepsilon = 0.1$

5.3. **Sierpinski Gasket.** Our next point of exploration was to take points from the Sierpinski Gasket. In order to do this, we start by defining a triangle on $\mathbb{R}^2$

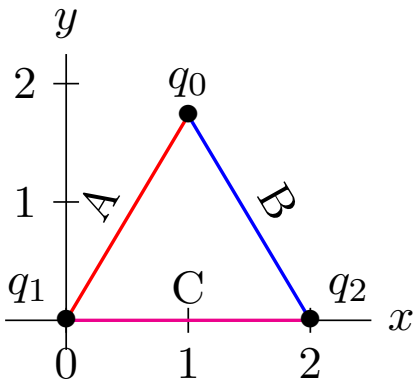


FIGURE 19. The initial equilateral triangle from which the gasket builds.

We define a variable X to be uniformly random within $[0, 3]$. If $X < 1$, then the corresponding point falls on the red segment, A, of the triangle. For $1 < X \leq 2$, the point is on segment B, and otherwise, the point is on segment C. The x-value of a point on A or B is defined as $x = X$. For a point on C, $2 \leq X \leq 3$, so $x = 2X - 4$. Using the x-value and the line the point is on, we calculate the y-value and end up with a seed point, $p_0 = (x, y)$. From here we define another variable a to be uniform random in $[0, 3]$ along with the set of similarities, $F = \{F_i(p) := \frac{1}{2}(p - q_i) + q_i \mid i = 0, 1, 2\}$. We then define a new point as follows

$$p_j = \begin{cases} F_0(p_{j-1}) \text{ if } 0 \leq a < 1 \\ F_1(p_{j-1}) \text{ if } 1 \leq a < 2 \\ F_2(p_{j-1}) \text{ otherwise} \end{cases}$$

A point undergoes this process a given m number of times where $m \in \mathbb{N}$. An example of the resulting set of points is depicted in Figure 20.

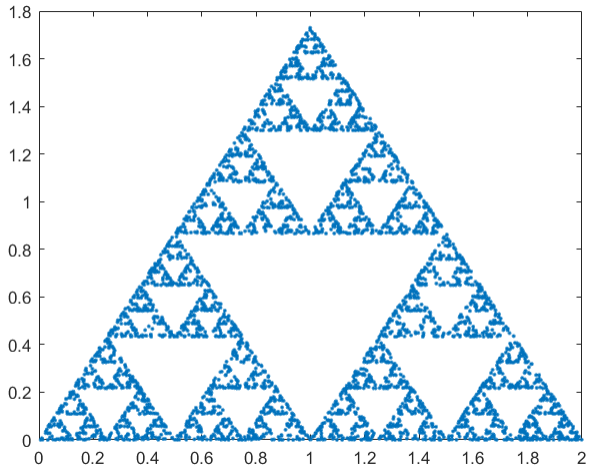
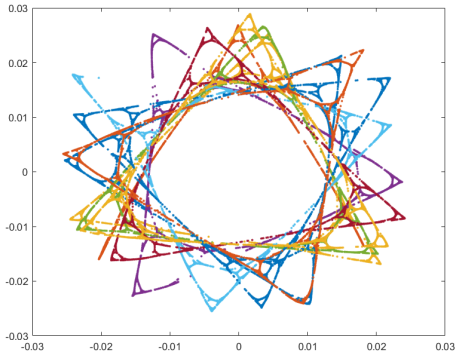
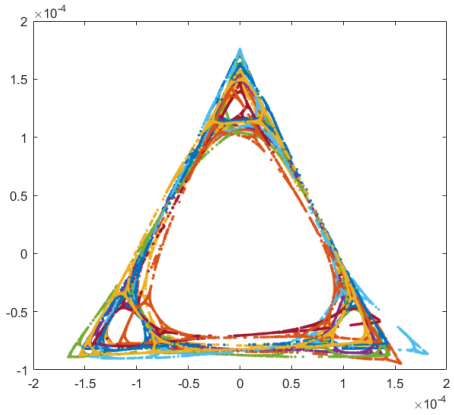


FIGURE 20. The resulting Sierpinski Gasket with 5000 points where $m = 15$.

To generate the Laplacian matrix, we use the standard Euclidean metric, d_E. Thus, the connections between points is etermined by whether $d_E(p_i, p_j) \leq \varepsilon$. We find that the resulting eigenmaps appear as the gasket as seen in Figure 21.



(A) Sierpinski Eigenmaps plotted over each other without any rotation. We affectionately called this the Sierpinski Flower.



(B) Sierpinski Eigenmaps plotted over each other each rotated through a linear transformation to align.

FIGURE 21. These plots depict 10 runs of the code, plotted over each other. Each time $n = 5000$, $m = 15$, and $\varepsilon = 0.04$.

6. Weighted graph

Another variation on this idea is to consider a Laplacian with different edge weights besides just 0 and 1. Let w_{ij} be defined as

$$w_{ij} = \begin{cases} e^{\frac{-|x_i - x_j|^2}{a}} & \text{if } |x_i - x_j| \leq \varepsilon, i = j, a > 0 \\ 0 \text{ else} \end{cases}$$

Like before, we write $W = (w_{ij})_{i,j=0}^{n-1}$ and call it the adjacency matrix. Using this new definition of the adjacency matrix, the degree matrix and subsequently the Laplacian matrix, eigenvalues, and eigenvectors, can be constructed. We normalize the eigenvectors by dividing each eigenvector by the minimum element of the vector, so that the elements within each eigenvector range from -1 to 1. We then plot eigencoordinates as before. We compare the eigencoordinates to Chebyshev polynomials using best fit curves. Polynomials for the quadratic, cubic, quartic, and quintic are shown in our pictures below.

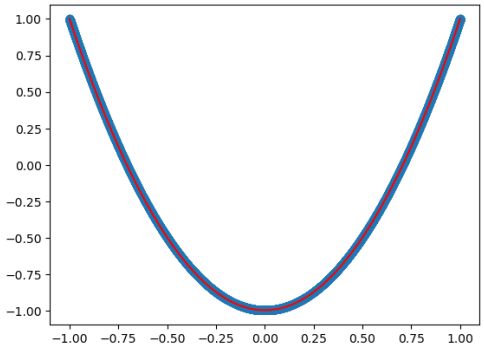


FIGURE 22. Best Fit Curve: $y = 2.0036467x^2 + 8.0965123 * 10^{-11}x - 1.000014520$
Compare with $T_2(x) = 2x^2 - 1$

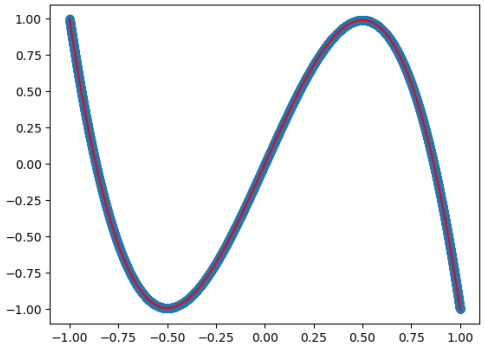


FIGURE 23. Best Fit Curve: $y = 3.977825x^3 + 1.13000164 * 10^{-10}x^2 - 2.978068x - 2.52833309 * 10^{-11}$
Compare with $T_3(x) = 4x^3 - 3x$

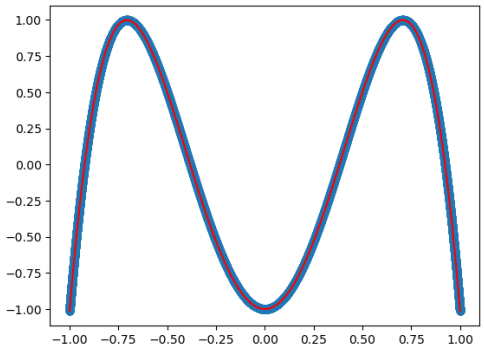


FIGURE 24. Best Fit Curve: $y = -8.0288566x^4 - 3.2939437 * 10^{-9}x^3 + 8.01490096x^2 + 1.36692 * 10^{-10}x + -1.00016946$
Compare with $T_4(x) = 8x^4 - 8x^2 + 1$

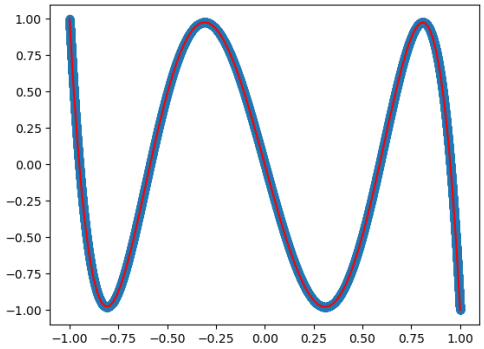


FIGURE 25. Best Fit Curve: $y = 15.7054676x^5 - 5.9303838 * 10^{-8}x^4 - 19.598658x^3 + 4.38873 * 10^{-8}x^2 + 4.89162x - 1.713723 * 10^{-9}$

Compare with $T_5(x) = 16x^5 - 20x^3 + 5x$