Report for exercise 5 from group I

Tasks addressed:	5
Authors:	Kejia Gao (03779844)
	Jingyi Zhang (03785924)
	Maximilian Mayr (03738789)
	Yizhi Liu (03779947)
	Felipe Antonio Diaz Laverde (03766289)
Last compiled:	2024–06–24

The work on tasks was divided in the following way:

Kejia Gao (03779844)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Jingyi Zhang (03785924)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Maximilian Mayr (03738789)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Yizhi Liu (03779947)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%
Felipe Antonio Diaz Laverde (03766289)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
	Task 5	20%

1 TASK 1

Report on task TASK 1, Approximating functions

1.1 Approximate the function in dataset (A) with a linear function

Figure 1 illustrates the visualization of original data points of dataset A and the approximated linear function. The parameter *cond* in the function $least\_squares(A, b, cond = 1e-5)$ is set to be $1e-5$ as default. The approximated function fits dataset A precisely, because the mean square error (MSE) is only 1.0605×10^{-10}.

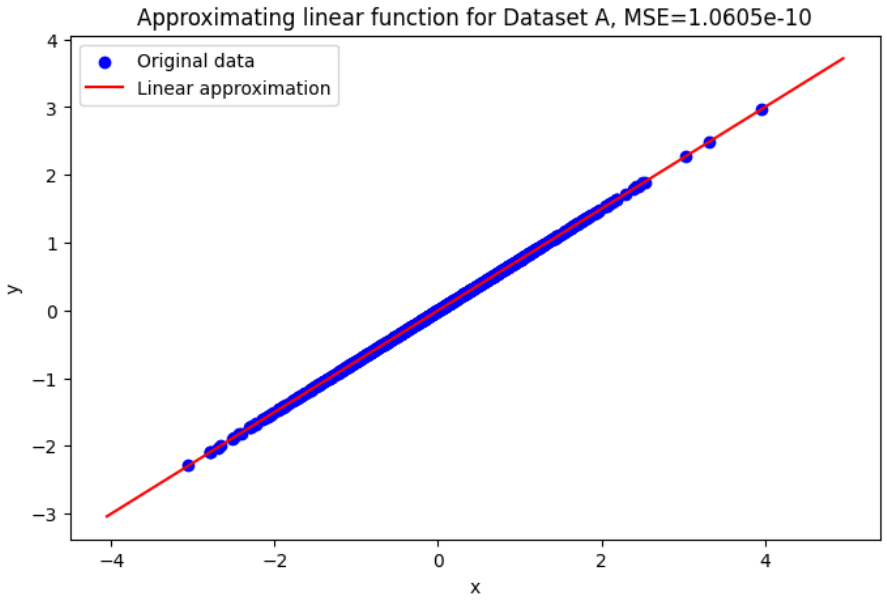


Figure 1: Approximating linear function for dataset A

1.2 Approximate the function in dataset (A) with radial basis functions

Figure 2 shows the visualization of original data points of dataset A and the approximated radial basis functions. The parameter *cond* in the function $least\_squares(A, b, cond = 1e-5)$ is set to be $1e-5$ as default. The approximated function is distorted at both of the ends, which is beyond the range of x of dataset A.

Radial basis functions are local approximators which mainly affect the regions close to their centers. For the regions beyond the original range of x of dataset A, the influence of these radial basis functions diminishes, which results in instability at the boundaries where there is less data to constrain the model.

The reason why radial basis functions are not appropriate for linear distributed data is that they tend to overfit by modeling simple linear relationships with complex nonlinear functions. The model may "memorize" the raw data and fail to generalize with unseen data. More specifically, the model is too complicated to interpret the linear distributed data that can be represented by 2 parameters: intercept and slope, which means that the huge complexity is unnecessary for this simple situation. Moreover, the complexity decreases the computational efficiency significantly with exponential operations, causing the waste of computational resources.

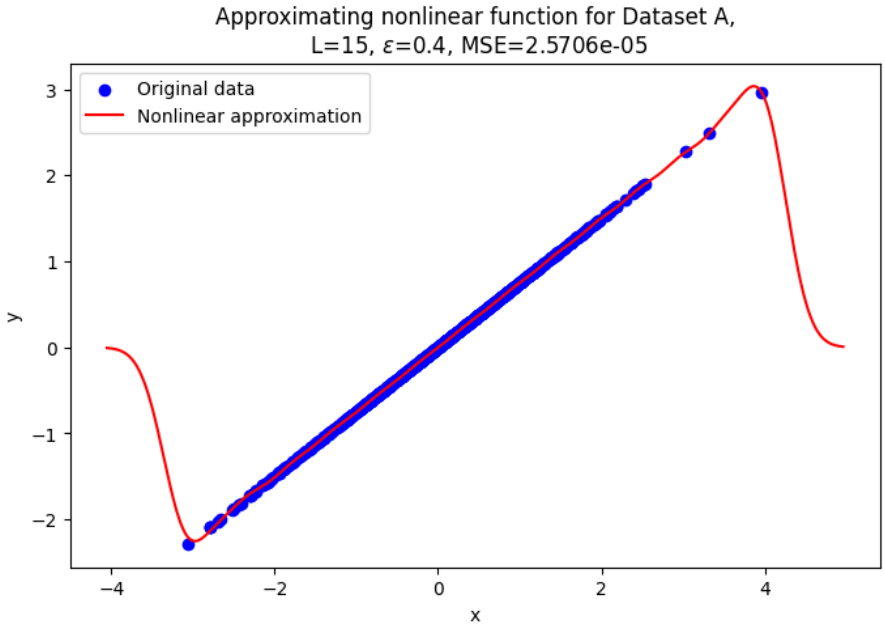


Figure 2: Approximating nonlinear function for dataset A

1.3 Approximate the function in dataset (B) with a linear function

Figure 3 displays the visualization of original data points of dataset B and the approximated linear function. The parameter *cond* in the function $least\_squares(A, b, cond = 1e-5)$ is set as $1e-5$ again. The approximated function fails to fit dataset B completely, because the linear function is a straight line but the original data points form a nonlinear curve. The mean square error (MSE) is 0.76256, a much larger value compared to that of fitting dataset A with a linear function. The main reason is that the linear function doesn't contain enough parameters to capture complex features of nonlinear data, which is called underfitting.

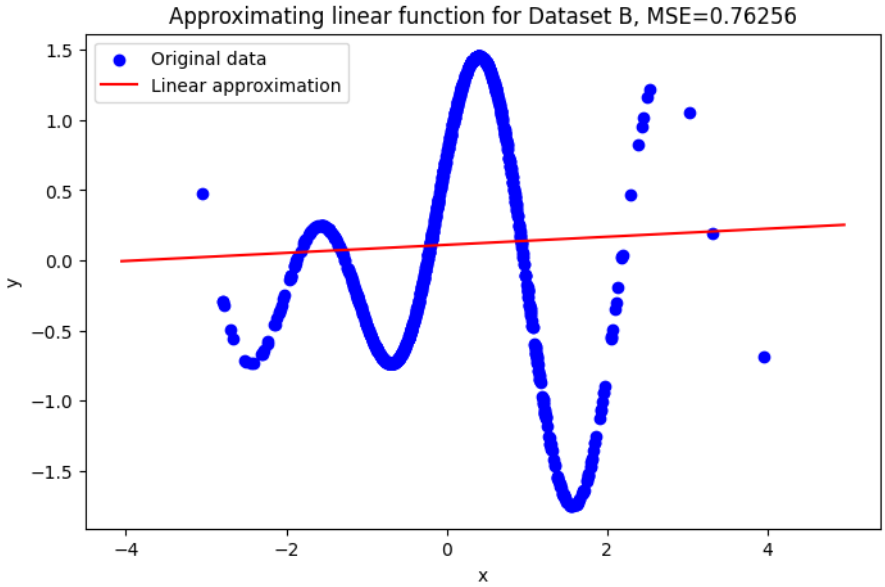


Figure 3: Approximating linear function for dataset B

Table 1: MSE of fitting dataset B w.r.t ϵ and L

$\epsilon \setminus$ L	5	10	15	20	25
0.1	0.58070	0.55407	0.40204	0.27340	0.15715
0.5	0.31696	0.07391	4.8923×10^{-5}	6.5365×10^{-6}	5.6469×10^{-7}
1.0	0.32005	2.1232×10^{-4}	4.9870×10^{-7}	1.5760×10^{-9}	2.4983×10^{-9}
1.5	0.31876	4.0129×10^{-5}	1.1718×10^{-5}	1.1070×10^{-5}	1.0796×10^{-5}
2.0	0.35031	0.0045959	0.0017526	0.0017473	0.0017265

1.4 Approximate the function in dataset (B) with radial basis functions

In this case, radial basis functions is defined by

$$\phi_l(x) = \exp\left(-\left\|x_l - x\right\|^2 / \epsilon^2\right) \tag{1}$$

The denominator for the exponential is ϵ^2 instead of ϵ, because it is more common and provides a smoother curve. And the approximated function is defined by the sum of $\phi_l(x)$:

$$f(x) = \sum_{l=1}^{L} c_l \phi_l(x), c_l \in \mathbb{R}^d \tag{2}$$

Grid search is implemented to select appropriate combination of L(number of radial basis functions) and ϵ. The candidates for L are 5, 10, 15, 20, 25 and the candidates for ϵ are 0.1, 0.5, 1.0, 1.5, 2.0. The center points are uniformly sampled over the domain of the dataset. For each of the 25 combinations, plot the comparison of original points from dataset B and approximated nonlinear function and calculate the MSE.

Table 1 displays the MSE of 25 combinations. With the increase of ϵ or L, MSE decreases at first and then increases. When ϵ is 0.1 and 2.0, the MSE is much larger than other situations. When $L = 5$, the model is underfitted due to insufficient radial basis functions. When $L = 25$, the model is overfitted because the number of radial basis functions is more than enough. Compared to the cases where $L = 20$, MSE increases from 1.5760×10^{-9} to 2.4983×10^{-9} when $\epsilon = 1.0$, which means the 5 more radial basis functions have negative effect on the model fitting.

According to the above mentioned analysis, 16 combinations are excluded and the rest will be visualized in the report. The remaining values are 0.5, 1.0, 1.5 for ϵ and 10, 15, 20 for L. The best combination is chosen by MSE and visualization.

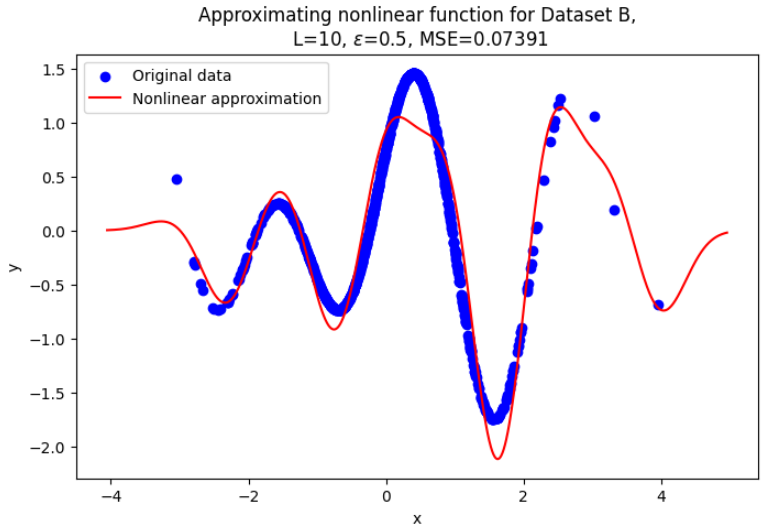
Figure 4 shows the plots when $L = 10$. The function with $\epsilon = 1.5$ fits the dataset best with the lowest MSE and the best fitting performance visualization.

Figure 5 illustrates the plots when $L = 15$. When $\epsilon = 0.5$, the curve is not very smooth at the end. The function with $\epsilon = 1.0$ or 1.5 fits the dataset better than that with $\epsilon = 0.5$.

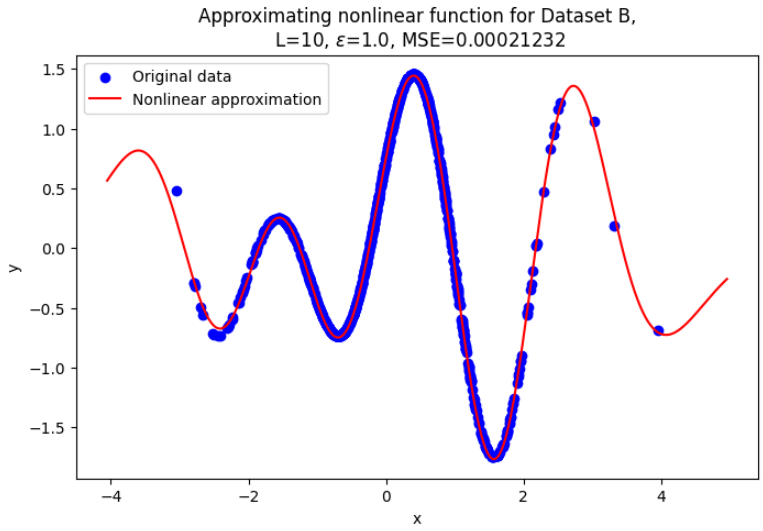
Figure 6 displays the plots when $L = 20$. All of them have a nice fit to dataset B.

Compared to $L = 15$, MSE is reduced when $L = 20$, but there is no obvious performance improvement with the 5 more radial basis functions. Therefore, it can be considered as overfitting for $L = 20$.

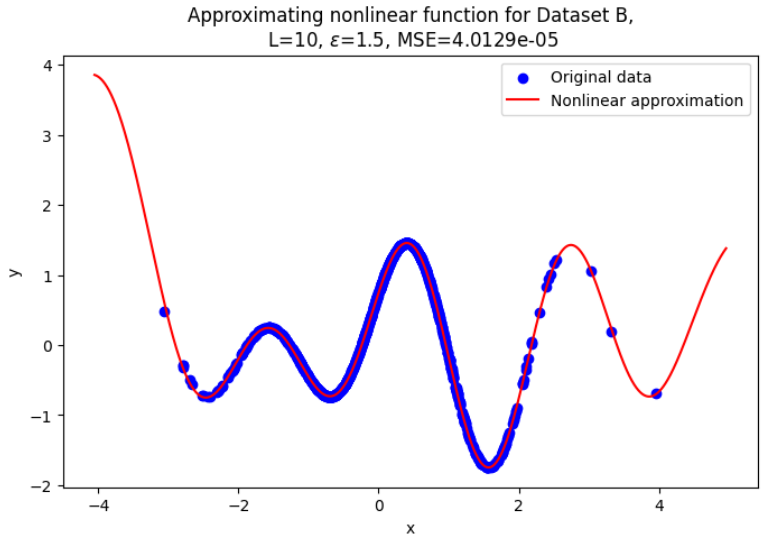
According to the exclusions, there exist only three combinations: $L = 10, \epsilon = 1.5$; $L = 15, \epsilon = 1.0$ and $L = 15, \epsilon = 1.5$. Finally, the combination $L = 15, \epsilon = 1.0$ is chosen because of the lowest MSE.



(a) $\epsilon = 0.5$

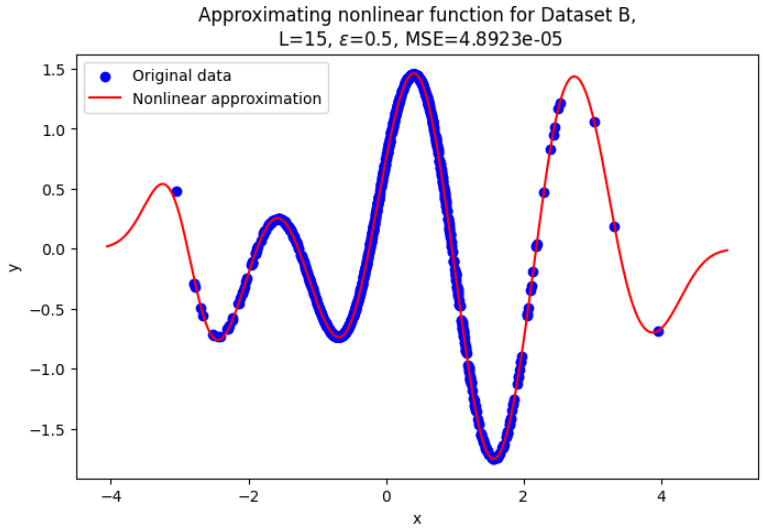


(b) $\epsilon = 1.0$

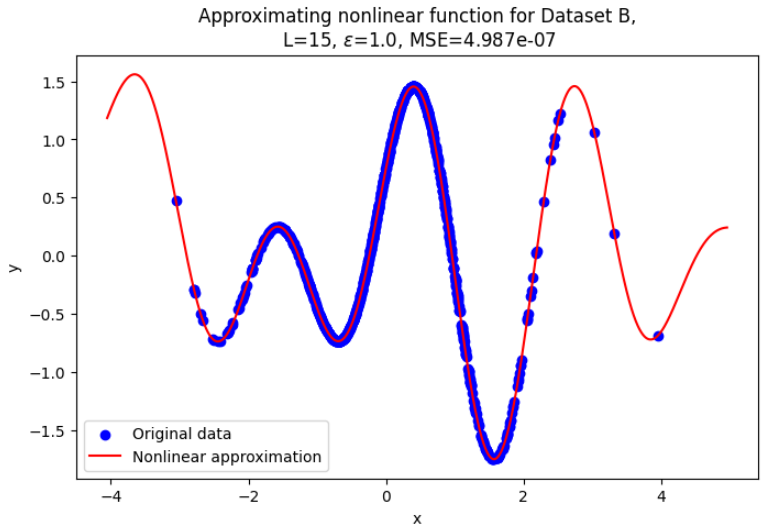


(c) $\epsilon = 1.5$

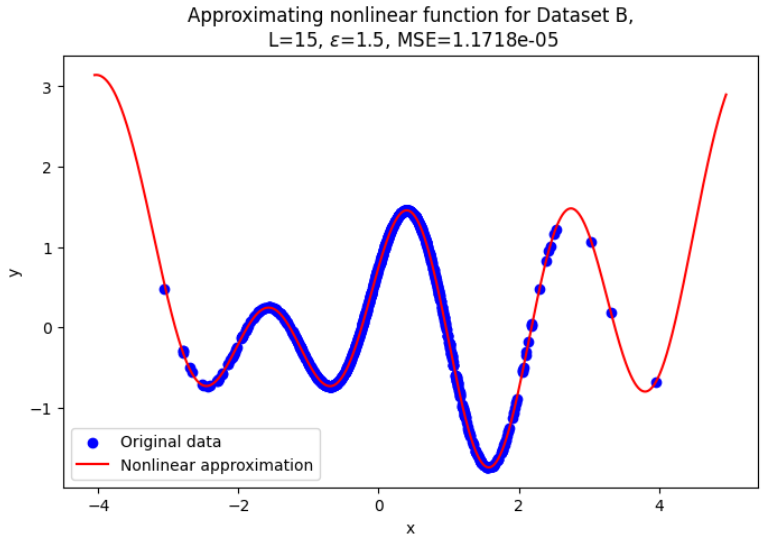
Figure 4: MSE of the nonlinear model prediction with $L = 10$



(a) $\epsilon = 0.5$

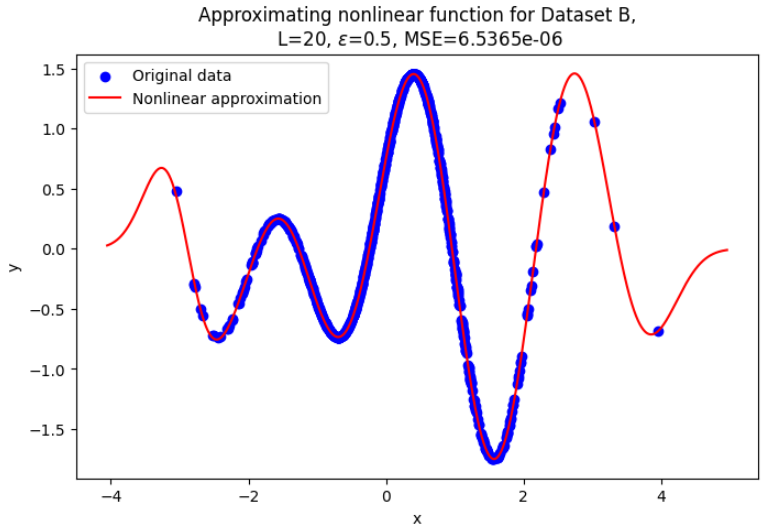


(b) $\epsilon = 1.0$

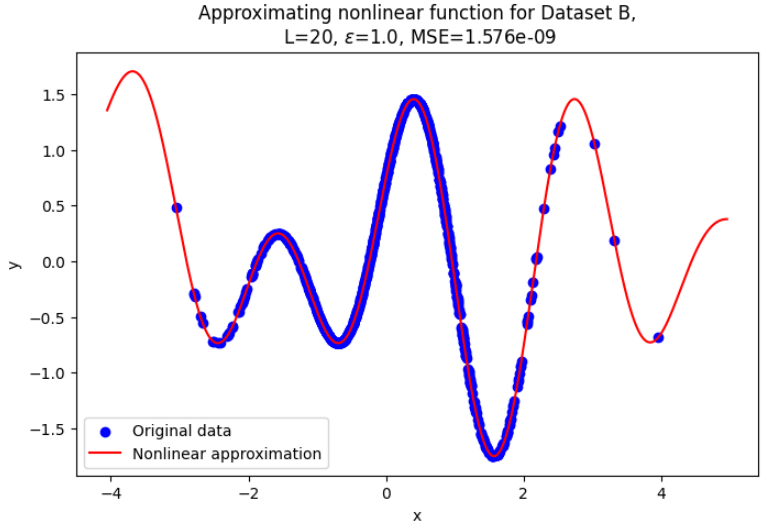


(c) $\epsilon = 1.5$

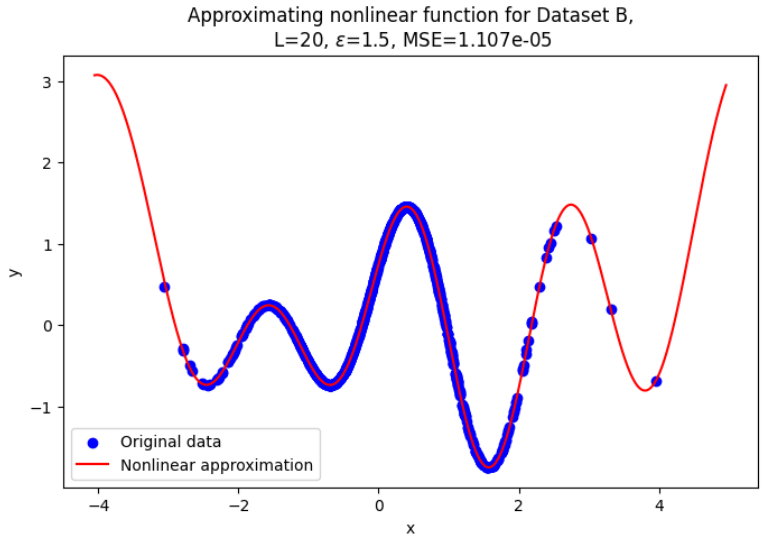
Figure 5: MSE of the nonlinear model prediction with $L = 15$



(a) $\epsilon = 0.5$



(b) $\epsilon = 1.0$



(c) $\epsilon = 1.5$

Figure 6: MSE of the nonlinear model prediction with $L = 20$

2 TASK 2

Report on task TASK 2, Approximating linear vector fields

2.1 Estimate the linear vector field

From the datasets `linear vectorfield data x0.txt` and `linear vectorfield data x1.txt`, shown in Figure 7

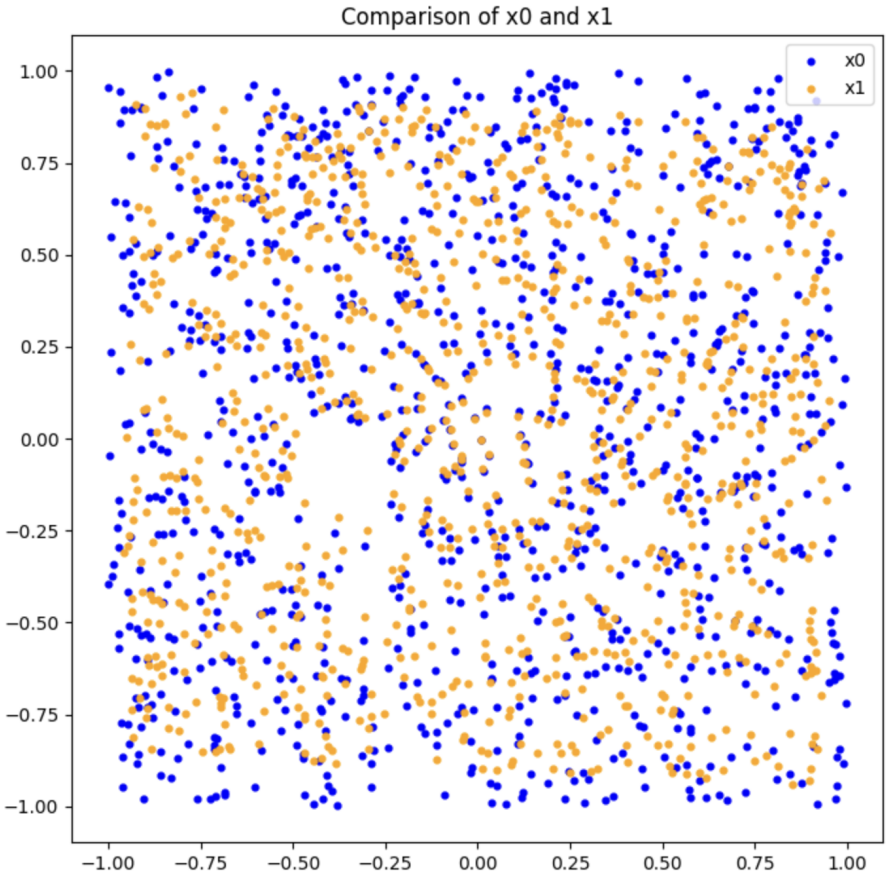


Figure 7: Comparison of x_0 and x_1

We can estimate the linear vector field by using:

$$\hat{v}^{(k)} = \frac{x_1^{(k)} - x_0^{(k)}}{\Delta t} \tag{3}$$

and

$$v(x_0^{(k)}) = v^{(k)} = Ax_0^{(k)}, \tag{4}$$

with a time step $\Delta t = 0.1$.
The estimated matrix A is:

$$\begin{bmatrix} -0.49355245 & -0.4638232 \\ 0.23191153 & -0.95737573 \end{bmatrix}$$

2.2 Compute the mean squared error

From the estimated matrix in 2.1, the mean squared error from this estimation is 0.0030599275959897355. This shows that the accuracy of the estimation is high.

2.3 Trajectory and phase portrait

By using the estimated linear vector field, we can solve the linear system and visualize the trajectory from point (10,10) in Figure 8. The phase portrait is also shown in Figure 8.

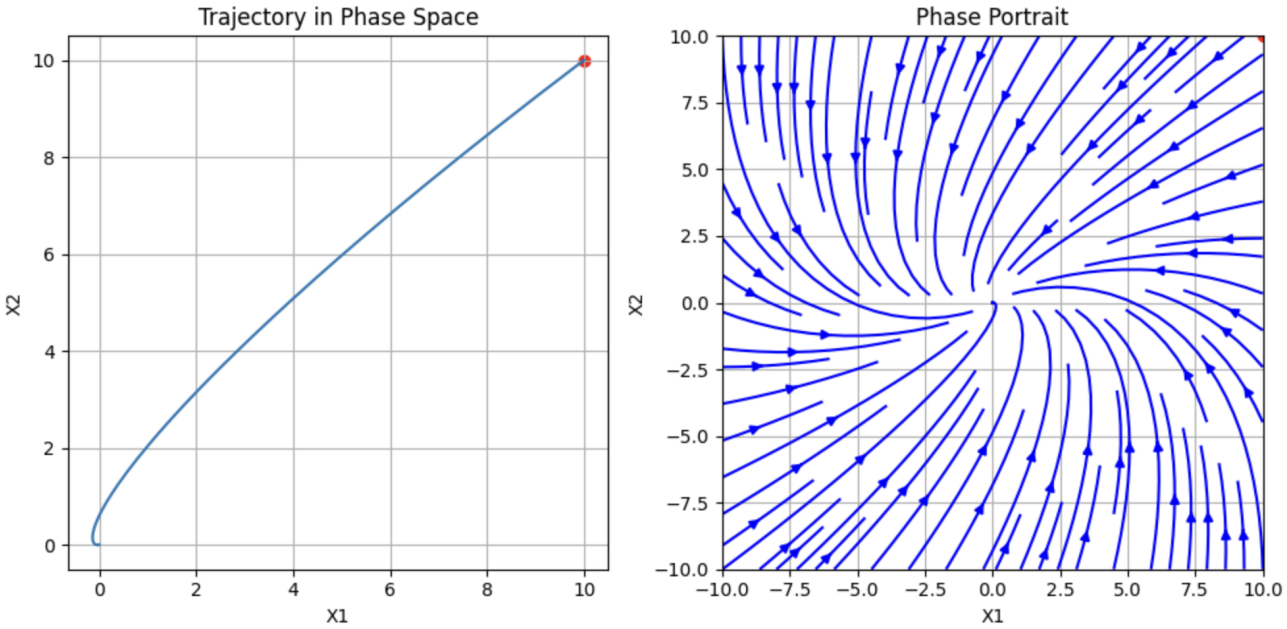


Figure 8: Trajectory from (10,10) and phase portrait of vector field

3 TASK 3

Report on task TASK 3, Approximating nonlinear vector fields

For this task, we are using a non-linear vector field datasets x0 and x1, being x0 the initial state, and x1 the time evolved state of x0 after some unknown time Δt, and we are asked to try to estimate the underlying time evolution operator ϕ describing the system, using both, a linear operator similar to previous tasks, with a dynamic equation of the form

$$\frac{d}{ds}\psi(s, x(t))|_{s=0} \approx Ax(t), \tag{5}$$

and a non-linear system of the form

$$\frac{d}{ds}\psi(s, x(t))|_{s=0} \approx C\phi(x(t)), \tag{6}$$

with A a matrix, C a coefficient matrix, and $\{\phi_l(x)\}_{l=1}^{L}$ a set of non-linear basis functions. In our case, we are using radial basis functions defined in Equation 1. For the centroids of the radial functions, we chose to sample the whole 2D space with uniformly spaced points, and vary the values of ϵ to find the one that better fits the data.

First, we import the data and visualize both states in time $x_0(t_0)$ and $x_1 = \psi(\Delta t, x_0)$

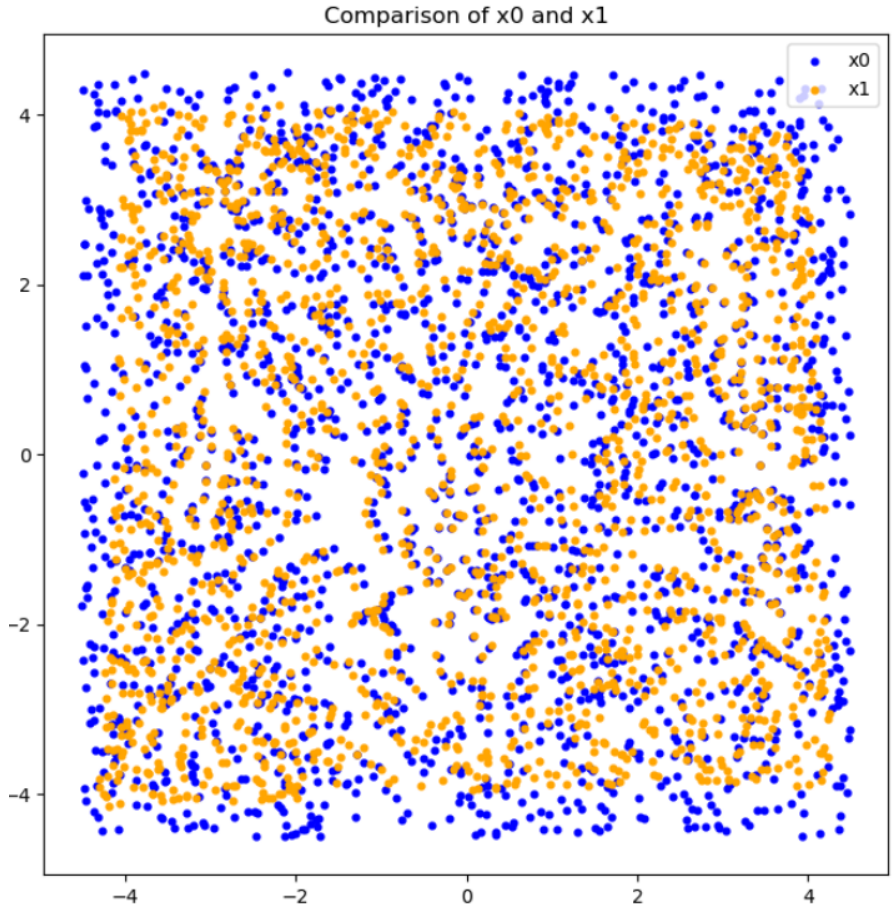


Figure 9: Comparison of x_0 and x_1 for a non-linear vector system.

In the first part, and similar to the previous tasks, we first have to use an approximate finite difference scheme to approximate the velocity vector field for small Δt using Equation 3. For this task, we use a small value of $\Delta t = 0.01$. Finally, with this vector field v_k, and again similar to the previous task, we can propagate or evolve the system in time, until a chosen final time t_f with step size of dt, such that at the end, the mean square error between the final approximated state $\hat{x}_1$ and the real final state x_1 is minimized.

To solve this task, we opted to chose a value of $t_f = 0.011$ and $dt = 0.001$, as to be able to have some relatively fine time discretization. We computed the total mean square error between all propagated points $\hat{x}_1^k$ and the real points x_1^k for all evenly spaced times between $[0,\ t_f]$ with spacing of dt. For this first part, we obtained the following result: the best fitting final time when propagating to the final state was for $t_f = 0.001$, with a MSE of approximately 0.037, and a MSE for the final time 0.011, MSE_{t_f}, of approximately 0.854. We simulated slightly above the actual final time Δt, which is technically unknown for the purposes of approximating the non-linear system, just to see if we actually can obtain possible better approximations after the actual final time, which is not desired, as stopping in the actual final time Δt might provide the best tried approximation by chance, while not actually necessarily it being a good approximation. The results support the hypothesis that the system is not lineal, as the best approximation MSE is actually relatively high (as will be seen in a moment when estimating for a non-linear system) and the best fitted final time 0.001 is quite far from Δt.

For the next part of the task, we estimate the system using a non-linear operator as described before. For this, we created uniformly spaced grids in 2D to sample the centroids of the radial basis functions with a total number of centroids sampled from $L = \{100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000\}$ and different values of epsilon, $\epsilon = \{0.1, 0.5, 1.0, 2.0, 5.0, 10.0\}$, increasing its value approximately by a factor of 2 in each iteration, and find the configuration which yields the lowest MSE with respect to the real final state x_1 similar to the previous task. Additionally, and for the sake of checking the correctness of the non-linear operator evolving the state as expected, and similar to the previous task, we also tested for different final times, to make sure that the best approximation $\hat{x}_1$ found was achieved in the expected final time. For this sub-task we

obtained that the best $\hat{x}_1$ was found for $t_f = 0.011, L = 250, \epsilon = 5.0$, with a MSE of approximately 2.569×10^{-4}. We see that, as mentioned before, not only this time the final time is very close to the actual final time Δt, but the MSE error obtained for the best approximation is much smaller (around 3 orders of magnitude) than that obtained for the linear approximation.

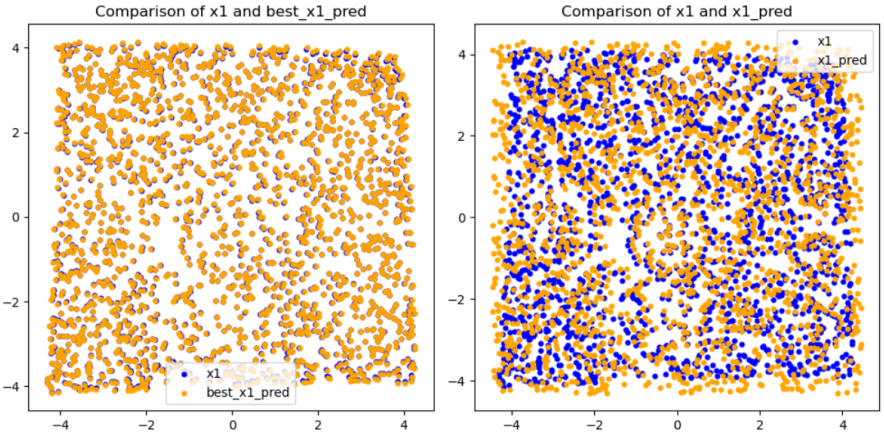


Figure 10: Comparison of $\hat{x}_1$ for a non-linear and linear systems respectively.

In Figure 10 we can see a close comparison of the final best approximation obtained using a non-linear (best_x1_pred, left) and linear (x1_pred, right) system respectively, where the difference between both solutions becomes very clear, hinting towards the fact that it seems impossible to model the time evolution of the system using a linear scheme.

For the final part of the task, we simulated the system using the optimal parameters found in the previous task, until a long final time $t_f = 30$, such that all different starting points x_0^k, which seem to be randomly sampled all over the initial 2D domain, converge towards the different possible steady states present in the underlying vector field governing the dynamics of the system. For this last task, although the function was implemented, at the end of the simulation an error does not allow to visualize the final steady states, although from the first two parts we can clearly conclude the system is non-linear, and therefore it should have a number of steady states bigger than 1, as otherwise it would be possible to describe it as a linear system, and we saw it does not seem to be the case from the first two parts.

4 TASK 4

Report on task TASK 4, Time-delay embedding

4.1 Part 1: Time-delay embedding of dataset

In the first part, we use the dataset in takens 1.txt with a shape of [1000,2]. This dataset represents the coordinates of 1000 points in space. We first load the dataset in the file as an array, and then plot the original dataset in two-dimensional space, using the first column to represent the x-axis coordinates and the second column to represent the y-axis coordinates, resulting in the following representation:

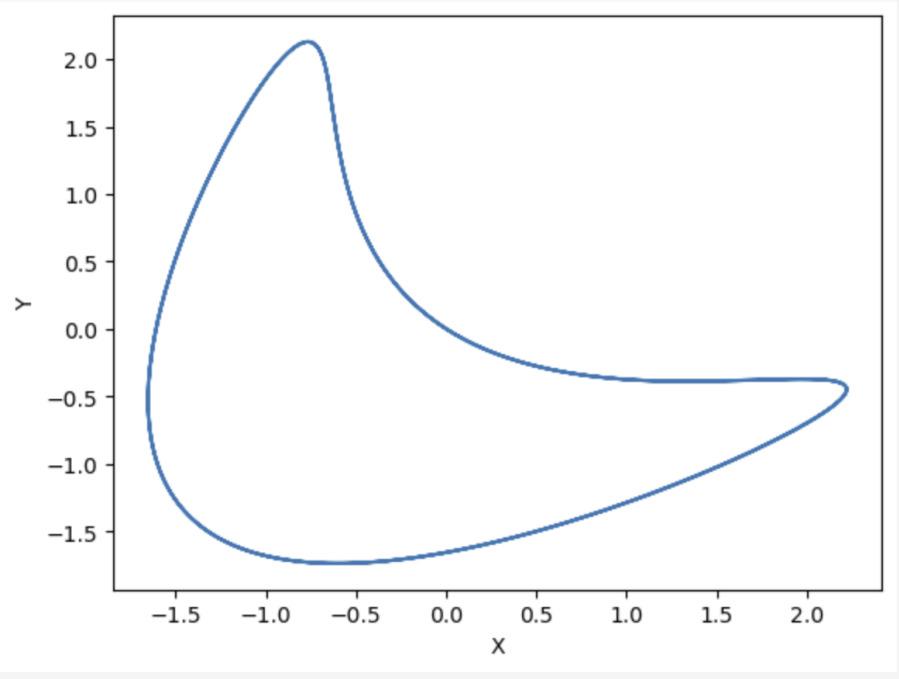


Figure 11: Visualization of raw data

After that, we only consider the first column of coordinates, assuming that the first coordinate is the value measured at each time unit. We can build a dynamic system based on the first coordinate. We use 1 to 1000 as the time represented as the x-axis coordinate, and the value of the first column is represented on the y-axis. The visualization result is as follows.

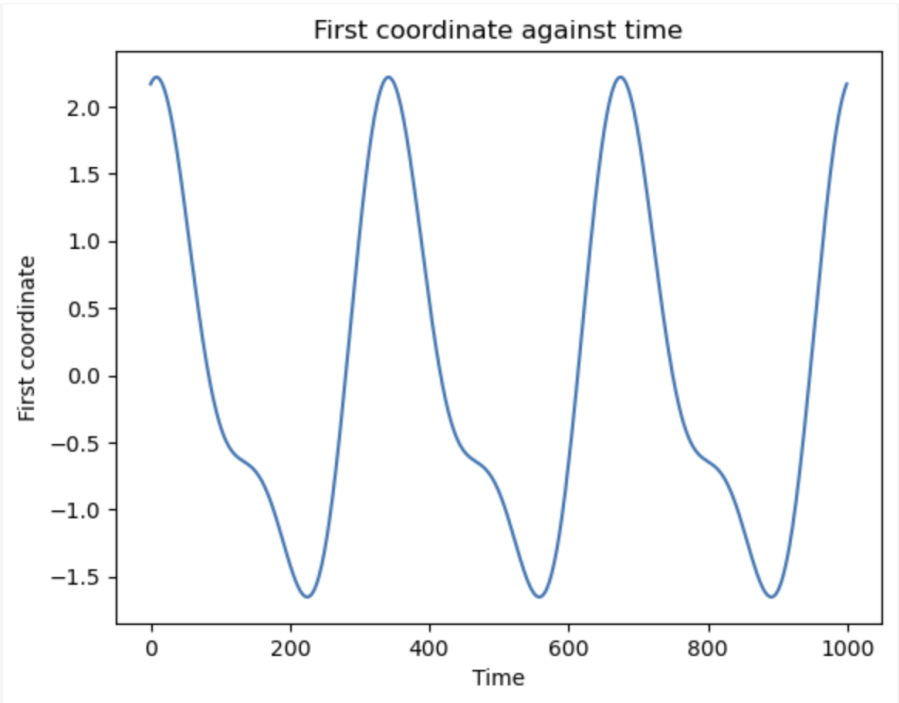
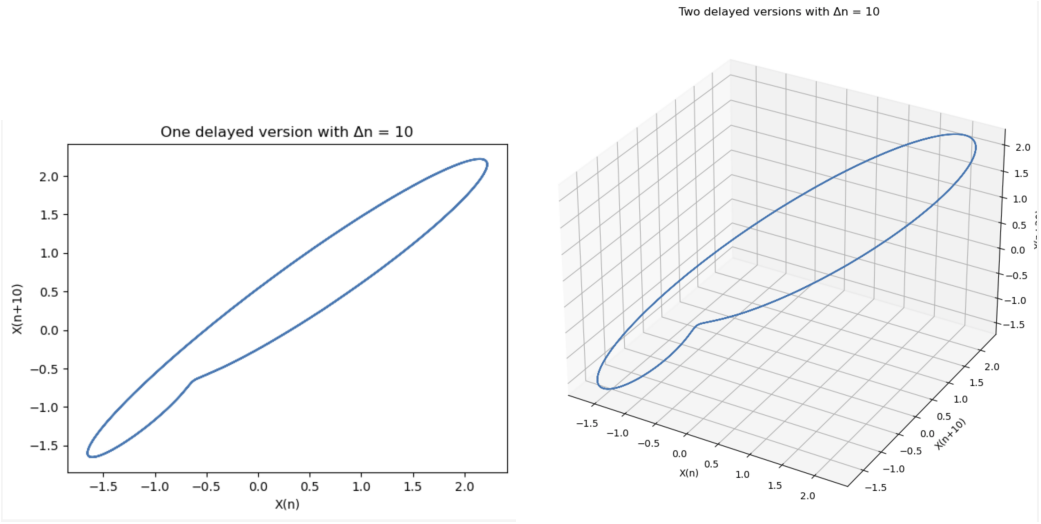


Figure 12: First coordinate against time

From the figure, we can observe that this is a periodic curve, showing three sets of cycles within a given time range. We want to explore whether the combination of these two coordinates is also periodic. We embed

it through time delay, which we can use to construct a phase space. According to Takens theorem, this phase space can capture the dynamic characteristics of the system.
For this purpose, we designed the 'time_delay' equation. The 'time_delay function is mainly used to create time delay coordinates from time series data to help analyze time-related patterns or dynamics in the data. The function works by receiving a dataset and some specified parameters, including the selected data column (col), the interval of the time step (delta_t), and the dimension of the output data (out_dim), which can be two-dimensional or three-dimensional. This approach allows users to derive multi-dimensional time delay representations from a single time series, which is very useful for applications such as dynamic system analysis, complex pattern recognition and predictive models. The above equations can be used to obtain coordinate against its delayed version.



(a) Δ n=10 (b) Δ n=10 and 2Δ n=20

Figure 13: First coordinate against its delayed version

From the above figure we can see that the first coordinate gives very similar results for the one delayed version and the two delayed versions, so 2 coordinates are enough to correctly embed this manifold.

4.2 Part 2: Time-delay embedding of trajectory of Lorenz attractor

In the second part of this task, we attempt to approximate chaotic dynamics using single time series data, specifically through the Lorenz attractor depicted in a previous exercise. The parameters used were $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$, ensuring that the system is in a chaotic regime. The starting point was (10, 10, 10). This exercise utilized Takens' Embedding Theorem to handle this fractal set. We use the functions lorenz_ODE and lorenz_traj to model and simulate the dynamics of the Lorenz attractor.

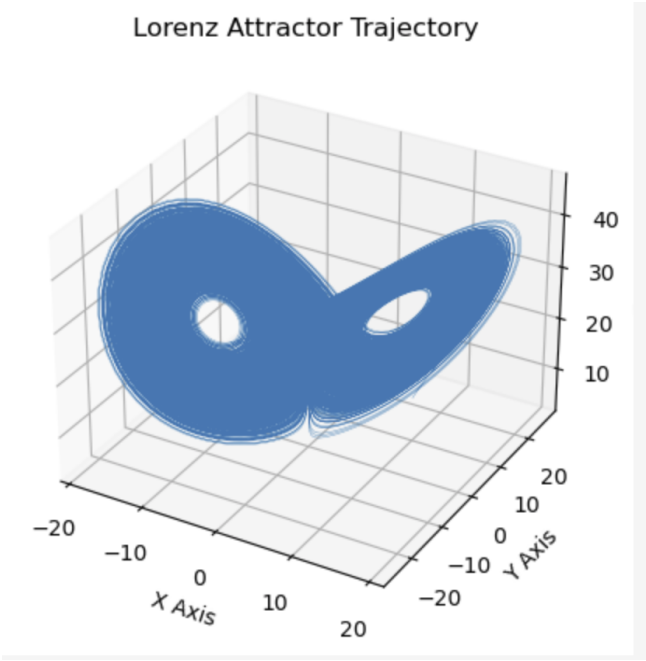


Figure 14: Lorenz attractor trajectory

We get an array of trajectories of shape [100000,3]. Similar to Part 1, we only consider the time delay in the x direction. By calling the method timedelay, we get a 3D array consisting of the coordinate x and its delayed version, with the delay time set to 10. This gives the relationship between the coordinate x(t) and x(t + Δt) and x(t + 2Δt).

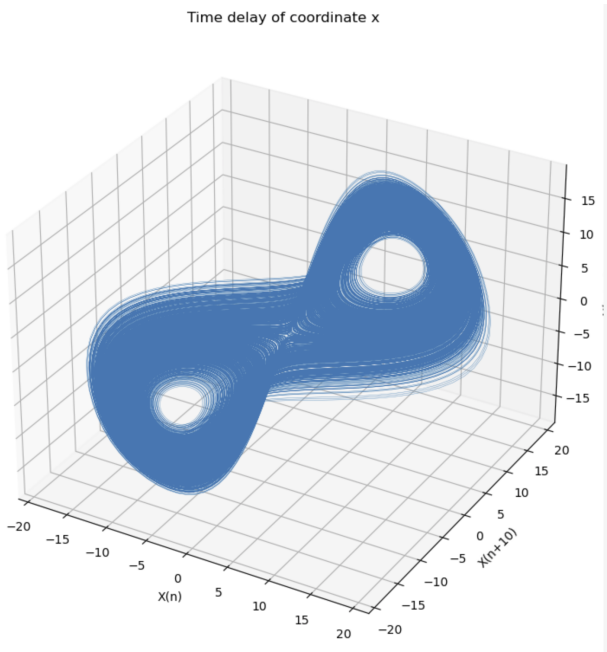


Figure 15: Time-delay embedding of Lorenz attractor in x axis

As shown in the figure, the embedding has similar overall characteristics to the original trajectory. Therefore, this time-delay embedding based on the coordinate x successfully captures the key features of the original trajectory.
The same operation is then performed on the coordinate z, which takes on a different shape than the x-axis.

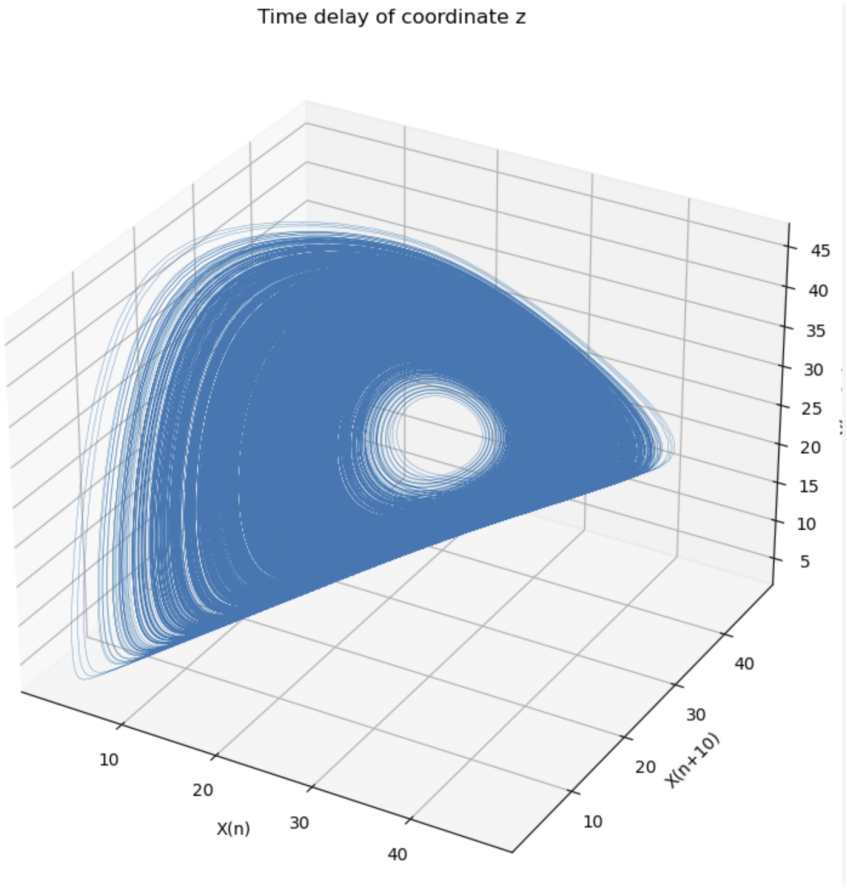


Figure 16: Time-delay embedding of Lorenz attractor in z axis

This is because the behavior of the variable $z$in the Lorenz system is distinctively different from that of x and y. The governing equations of the Lorenz system illustrate how the dynamics of z are influenced by both x and y. However, unlike the direct coupling observed between x and y, the feedback of z into the dynamics of x and y is not as straightforward. Specifically, the z-variable is driven by the interaction of x and y through the term xy minus a decay component governed by βz. It does not exert a direct influence on x, but impacts y through the term $\rho - z$. This indirect mode of influence may render z less representative of the system's overall dynamics when it is used alone for embedding purposes.

5 TASK 5

Report on task TASK 5, Learning crowd dynamics

5.1 Part 1: Delay embedding with 350 delays of the first three measurement areas

In this task, we worked with the MI data set while disregarding a burn-in period of 1000 time steps. As mentioned on the exercise sheet, this data set is periodic, which can also be empirically observed. As a result, its intrinsic dimension is 1. So, according to Takens theorem one would need at least 3 dimensions to embed the data. We were supposed to create an embedding consisting windows containing 3 measurement areas and 350 delays each. The next window starts one time step after the previous window ends, thus sharing the information about 350 time steps. We extracted the windows directly from the data matrix and used `numpy` to `flatten` them to vectors. Moreover, we used as many windows as possible and stacked them all together into an embedding matrix of shape $M \times 1053$, where M is the number of windows. Finally, we applied PCA to the matrix to reduce its dimension to $M \times 3$.

5.2 Part 2: Plotting the embedding

In this part, we plotted the embedding we received after applying PCA one time for each measurement area. These plots are displayed in Figure 17. Naturally, all plots show the same curve as the embedding is always the same. They differ in the coloring which represents the utilization of the respective measurement area at that point of the period.

5.3 Part 3: Learning the dynamics

To learn the dynamics on the curve embedded in principal components, the change of the arclength over time is studied. We computed the velocity on the arclength by iterating over the entire embedding and summing up the arclengths between every two embedding points x_{t+1} and x_t, i.e. its Euclidean norm. The velocity at a given time step was computed by dividing the cumulative arclength by the progressed time. Since the data set and also the embedding are periodic, the cumulative arclenths and also the progressed time are reset to 0 after every period.

Figure 18 shows the velocity on the arclength throughout all 7 days and nicely portrays the periodicity of the data set. Figure 19 shows the velocity in one period in more detail.

5.4 Part 4: Prediction of utilization of the MI building

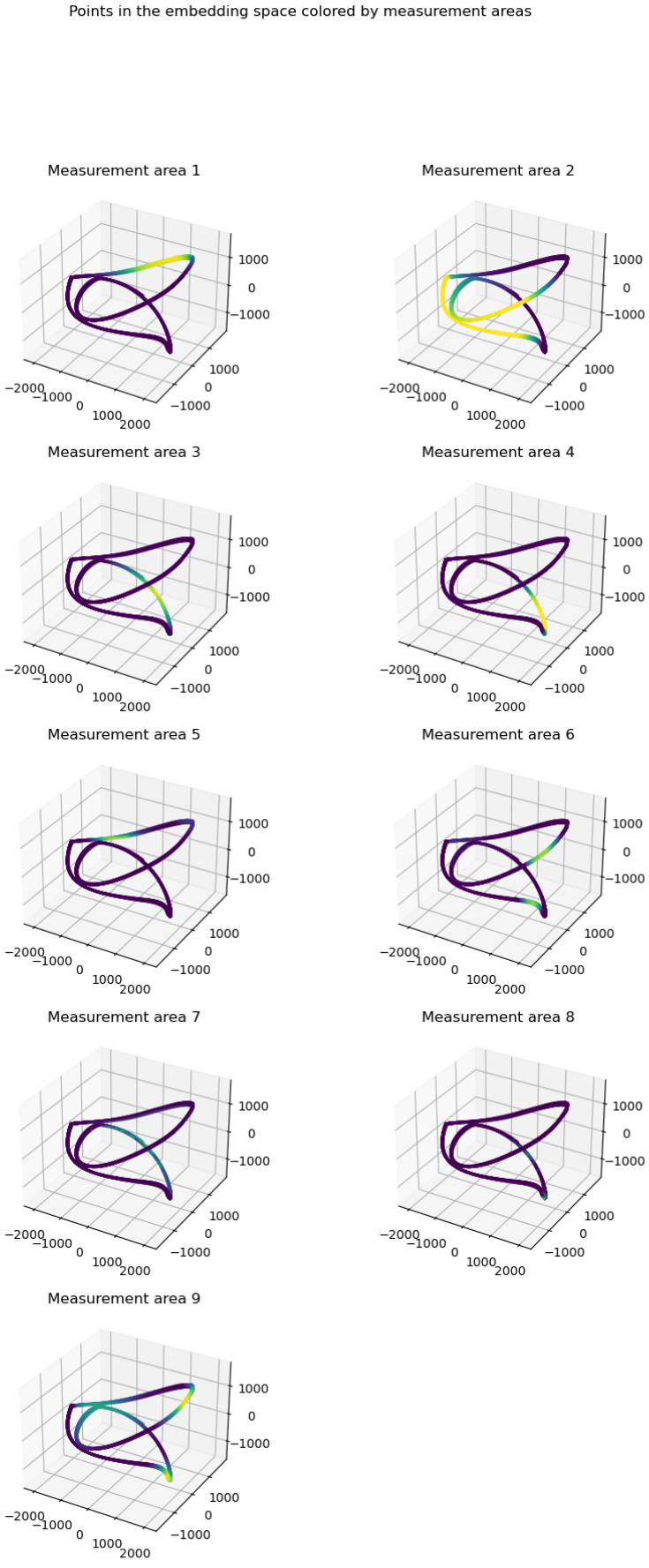
In the final part, we used the previous results to predict the utilization of the MI building (i.e. measurement area 1) over a time span of 14 days. We succeeded to do so by using the arclenghts over time of the first period as input values for the non-linear function approximation descrived in previous chapters. We were not interested in the radial basis functions but just in their coefficients because we could reuse them to solve the IVP which yields the arclengths for the entire 14 days.

We then retrieved the prediction by conducting another non-linear function approximation which maps the arclength to the measurement data. Since the measurement data only contains 7 days of data, we just repeated the mapped arclengths for a second time which is not problematic because the data is periodic.

For all computations of radial basis functions, the values $L = 2000$ and $\epsilon = 5$ were used. Similar to task 1, those values were determined by empirically studying and comparing different values.

Figure 20 shows the predicted utilization. During the first half, the actually measured data is plotted as well, to provide a comparison between the two. By visual observation, we conclude that our prediction is in general good while still not matching the precise values.

References


Points in the embedding space colored by measurement areas
Measurement area 1
Measurement area 2
Measurement area 3
Measurement area 4
Measurement area 5
Measurement area 6
Measurement area 7
Measurement area 8
Measurement area 9

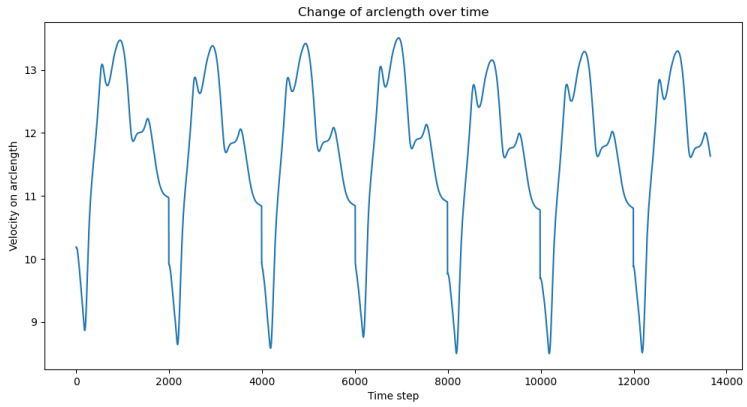


Figure 18: Change of the arclength over time in the entire measured time

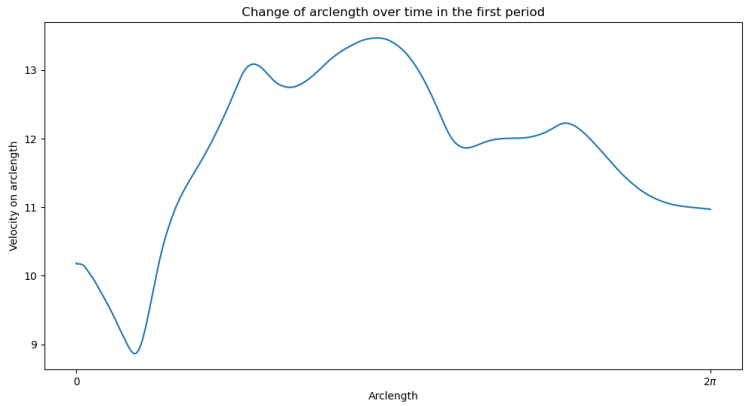


Figure 19: Change of the arclength over time in one period

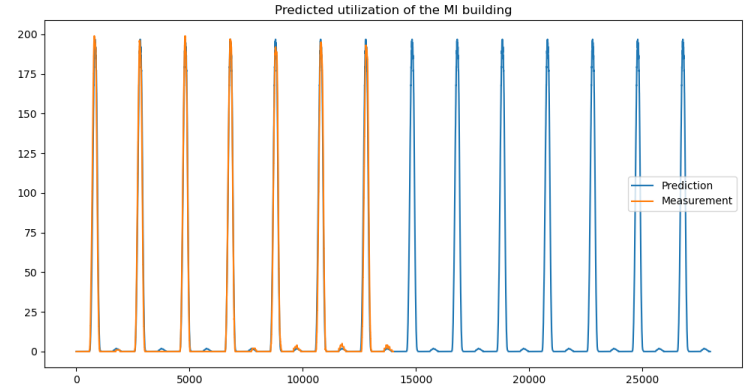


Figure 20: Utilization of the MI building