**Report for exercise 5 from group J**

| | |
|---|---|
| Tasks addressed: | 5 |
| Authors: | Jia Long Ji Qiu<br>Jiabo Wang<br>Yilun Liu |
| Last compiled: | 2023–02–25 |
| Source code: | https://github.com/jialongjq/mlcms |

The work on tasks was divided in the following way:

| | | |
|---|---|---|
| Jia Long Ji Qiu | Task 1 | 1/3 |
| | Task 2 | 1/3 |
| | Task 3 | 1/3 |
| | Task 4 | 1/3 |
| | Task 5 | 1/3 |
| Jiabo Wang | Task 1 | 1/3 |
| | Task 2 | 1/3 |
| | Task 3 | 1/3 |
| | Task 4 | 1/3 |
| | Task 5 | 1/3 |
| Yilun Liu | Task 1 | 1/3 |
| | Task 2 | 1/3 |
| | Task 3 | 1/3 |
| | Task 4 | 1/3 |
| | Task 5 | 1/3 |

**Report on task 1, Approximating functions**

In this first exercise, we have to demonstrate our understanding of function approximation with two examples. These two datasets (A) `linear_function_data.txt` and (B) `nonlinear_function_data.txt` are downloaded from Moodle and each of them contains 1000 one-dimensional point with two columns: x (first column) and f(x) (second column). This task consists of three parts and in all parts we have used a least-squares minimization to obtain the matrices A (linear case) and C (nonlinear case) as described in the following equations:

$$\min_{\hat{f}} e(\hat{f}) = \min_{\hat{f}} \|F - \hat{f}(X)\|^2 = \min_{A} \|F - XA^T\|^2 \tag{1}$$

$$\min_{\hat{f}} e(\hat{f}) = \min_{\hat{f}} \|F - \hat{f}(X)\|^2 = \min_{C} \|F - \phi(X)C^T\|^2 \tag{2}$$

The notebook `task1.ipynb` has been created to make and plot the approximations of the datasets. The algorithms used to make the approximations such as the LSE (Least Squares Error) and the RBF (Radial Basis Functions) are implemented in the file `utils.py`.

**Part 1:**

In the first part, we have to approximate the function in dataset (A) with a linear function, so we have to minimize the equation 1. To solve it we have used the least-squares minimization algorithm of the scipy library. In the figure of above we can see how our approximated function fits with the dataset, in fact, it fits very well.

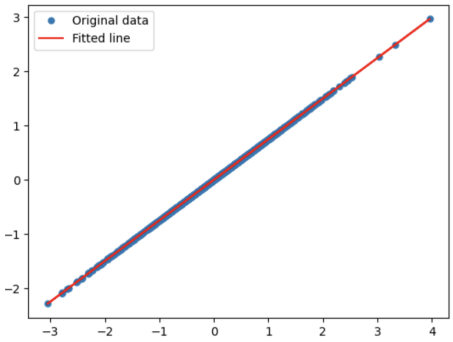


Figure 1: Dataset (A) with a linear function

Finally regarding to the question why it is not a good idea to use radial basis functions for this dataset is that as we can see in the figure 1 the data is linear, so it's obvious that using a linear function to approximate a linear dataset is the best option and the radial basis functions is to approximate nonlinear functions.

**Part 2:**

In this second part, we have to approximate the function in dataset (B) with a linear function, so as in the previous part, we have minimized the equation 1 and we have solved it with the least-squares minimization algorithm. In the figure of above we can see our approximated function with the dataset, clearly, it does not fit well. This is due to that our dataset is not linear, so we cannot approximate a nonlinear dataset with a linear function.

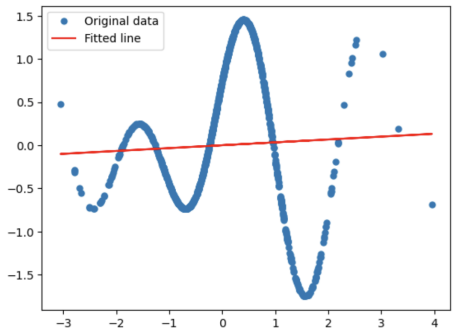


Figure 2: Dataset (B) with a linear function

**Part 3:**

Finally, in this last part we have to approximate the function in dataset (B) with a combination of radial functions, that turns out to be a nonlinear function. In this case, we have to minimize the equation 2. First, we have applied the L radial functions to our dataset (B) and then we have applied the least-squares minimization algorithm to the L transformed dataset. As we can see in the figure above, this function fits much better than the linear function in the previous task.

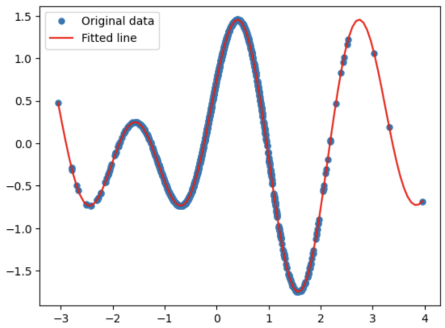


Figure 3: Dataset (B) with a nonlinear function

The radial basis functions we have created is defined by:

$$\phi_l = \exp(-\|x_l - x\|^2/\epsilon) \tag{3}$$

As said in the statement, we have just used a uniformly spaced sampling over the space $[x_{min}, x_{max}]$, which is the domain where we want to approximate the function to define our center points $x_l$. We have use the instruction np.linspace(-$x_{min}$, $x_{max}$, $L$) to define the center points, where L is the number of center points and also the number of radial basis functions we have created. If L is exactly the same as the number of data points, then it will fit perfectly, but it is not what we want because it is causing overfitting on our dataset and generalization, interpolation and extrapolation may not be very good. In our case, we have chosen 50 for the parameter L for 1000 datapoints of our dataset. It is quite small comparing with the number of datapoints, but as we can see in the figure 3, it is performing quite well. And after trying several $\epsilon$'s, we have chosen $\epsilon = 1$, which is the one that performs the best. Also comparing the results of choosing $\epsilon$ or $\epsilon^2$ in the denominator for the radial functions, we have gone with $\epsilon$ because it gives better results.

**Report on task 2, Approximating linear vector fields**

In this second task, we have to approximate linear vector fields. For this purpose, we have downloaded the datasets `linear_vectorfield_data_x0.txt` and `linear_vectorfield_data_x1.txt` from Moodle. Each of them contains 1000 rows and two columns, for 1000 data points $x_0$ and $x_1$ in two dimensions. This task also contains three parts.

The notebook `task2.ipynb` has been created to estimate the linear vector fields.

**Part 1:**

In the first part of this task, we have estimated the linear vector field that was used to generate the points $x_1$ from the points $x_0$ (Figure 4).

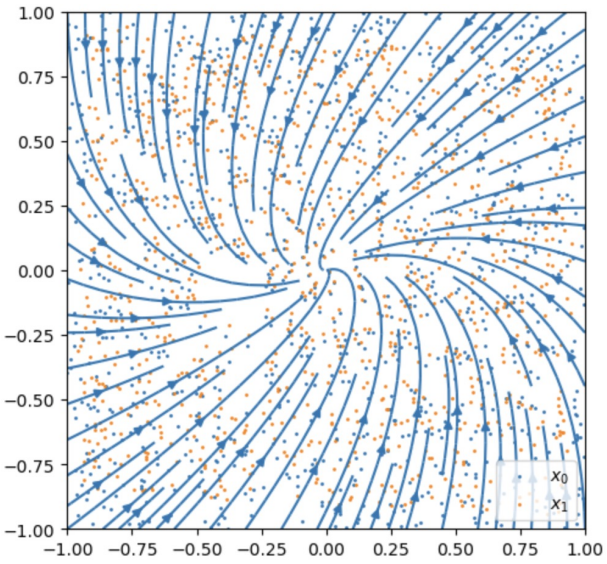


Figure 4: Estimated linear vector field that was used to generate the points $x_1$ from the points $x_0$

Firstly, we have used the finite-difference formula from the equation 4 to estimate the vectors $v^{(k)}$ at all points $x_0^{(k)}$, with a time step $\Delta t = 0.1$.

$$\hat{v}^{(k)} = \frac{\hat{x}_1^{(k)} - \hat{x}_0^{(k)}}{\Delta t} \tag{4}$$

Then, we have approximated the matrix $A \in \mathbb{R}^{2x2}$ with a supervised learning problem, by using the least-squares minimization algorithm. Since the vector field is linear, for all k:

$$v(\hat{x}_0^{(k)}) = \hat{v}^{(k)} = A\hat{x}_0^{(k)} \tag{5}$$

**Part 2:**

In the second part, we have solved the linear system $\dot{x} = \hat{A}x$, where $\hat{A}$ is the approximated matrix we have found in the previous part, such that $\hat{A} \approx A$, with all $x_0^{(k)}$ as initial points, up to a time $T_{end} = \Delta t = 0.1$. Then we got the estimates for the points $x_1^{(k)}$. The linear system has been solved in two different ways: with the Euler's method's approach and using the method `integrate.solve_ivp` of the Python library and, since this second method is a more accurate integration method, it will be used for the next tasks as well. Once we got $\hat{x}_1^{(k)}$ (the estimates of $x_1^{(k)}$), we have computed the mean squared error to all the known points $x_1^{(k)}$ with the following formula:

$$\frac{1}{N}\sum_{k=1}^{N} \|\hat{x}_1^{(k)} - x_1^{(k)}\|^2 \tag{6}$$

with N = 1000. The error of our estimation is $3.06 \times 10^{-3}$, which is small enough to conclude that approximating the vector field using a linear approach was a good choice (as expected), and that it was correctly approximated.

**Part 3:**

In the last part of this task, we solved again the linear system $\dot{x} = \hat{A}x$ with our matrix approximation, but now with a initial point (10, 10), far outside the initial data and for $T_{end} = 100$. In this case, for the trajectory, we have used the Euler's method as it was the method provided in the previous exercise. Then we have plotted the trajectory as well as the phase portrait in a domain $[-10, 10]^2$ as we can see in the Figure 5.

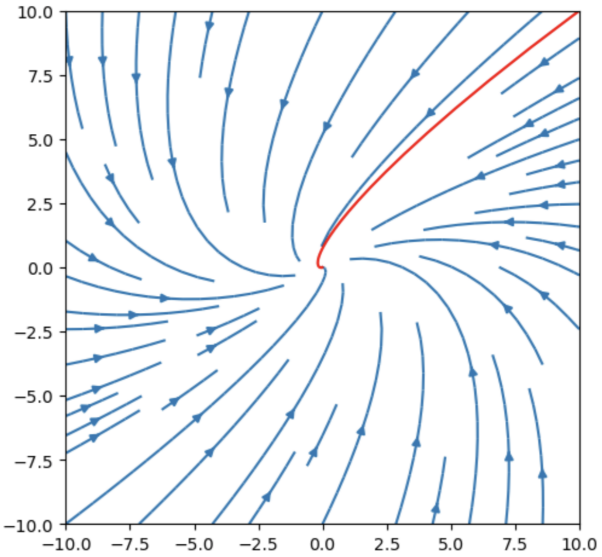


Figure 5: Phase portrait with a trajectory in the domain $[-10, 10]^2$

We can observe that there is a attractive steady state in the origin (0, 0) in the phase portrait and the red line is the trajectory that the initial point at (10, 10) follows to reach the steady state.

**Report on task 3, Approximating nonlinear vector fields**

After approximating linear vector fields, in this task we have to approximate now nonlinear vector fields using the datasets `nonlinear_vectorfield_data_x0.txt` and `nonlinear_vectorfield_data_x1.txt`. These datasets contains each of them 2000 rows and two columns, for 2000 data points $x_0$ and $x_1$ in two dimensions.

The first dataset contains the initial points $x_0$ over the domain $[-4.5, 4.5]^2$, while the second one contains the same points advanced with an unknown evolution operator $\psi : T \times \mathbb{R}^2 \to \mathbb{R}^2$, such that:

$$x_1^{(k)} = \psi(\Delta t, x_0^{(k)}), k = 1, ..., N \tag{7}$$

with a small $\Delta t = 0.01$, i.e., smaller than in the task on linear vector fields.

The notebook `task3.ipynb` has been created to cover this task, which has three parts as well.

**Part 1:**

As in the previous task, in this first part we have to estimate the vector field describing $\psi$ with a linear operator $A \in \mathbb{R}^{2x2}$, such that

$$\frac{d}{ds}\psi(s, x(t))\Big|_{s=0} \approx \hat{f}_{linear}(x(t)) = Ax(t). \tag{8}$$

For estimating the vector field $A$ we have used once again the least-squares minimization algorithm. The phase portrait of this vector field alongside the points $x_0$ and $x_1$ can be observed in Figure 6.

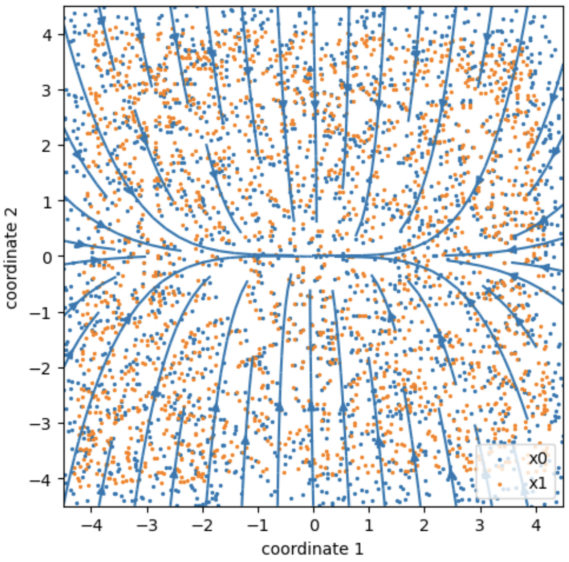


Figure 6: Estimated vector field with a linear operator

Once we have obtained $A$, we have to check now how good was the approximation by calculating the mean squared error between all the approximated and known end points. Since we had applied the finite-difference formula with $\Delta t = 0.01$ for estimating $A$, it is logical to choose it again for integrating the linear system in order to obtain the approximated end points $\hat{x}_1^{(k)}$ as close as possible to the known end points $x_1^{(k)}$.

The mean squared error between all the approximated end points and the known end points for $\Delta t = 0.01$ is equal to $3.729 \times 10^{-2}$, which is small but pretty big compared to the error obtained in the previous task.

**Part 2:**

Now, we have to estimate the vector field using radial basis functions, such that

$$\frac{d}{ds}\psi(s, x(t))\bigg|_{s=0} \approx \hat{f}_{rbf}(x(t)) = C\phi(x(t)). \tag{9}$$

The central points have been equally distributed all over the domain $[-4.5, 4.5]^2$. For choosing the number of central points $L$, we have tried different values between 100 and 1000 and we have stuck with the one that minimized the mean squared error later on, which was 100. Moreover, choosing a relatively small number compared to the number of points is a good option for achieving a good generalization and reducing the execution time. As for $\epsilon$, the value that minimized the MSE was $\epsilon = 0.75$ (not squared).

Having the radial basis functions $\phi(X)$ defined and applied to all points $x_0$, we obtained the coefficient matrix $C$ using the least-squares minimization algorithm. For visualizing the phase portrait of this new vector field, the function $\hat{f}_{rbf}(x(t)) = C\phi(x(t))$ has been applied to several points equally distributed over the domain in order to obtain the approximated vector in each point. The phase portrait alongside the points $x_0$ and $x_1$ is displayed in figure 7. Note that this new nonlinear vector field fits much better the dataset points than the linear one.

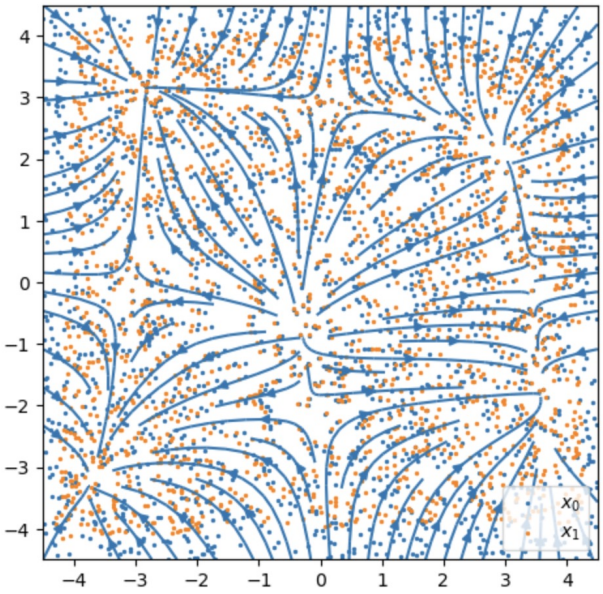


Figure 7: Estimated vector field using radial basis functions

Let's check again how good was the approximation by using the same methodology as in the previous part, but now integrating the linear system given by $\dot{x} = C\phi(x(t))$. The mean squared error is now equal to $6.731 \times 10^{-4}$, which is significantly smaller compared to the error obtained in the last part, where we used a linear approximation. With this we can conclude that the vector field is nonlinear, as a more precise vector field has been obtained with a nonlinear approximation.

**Part 3:**

In this last part we were asked to figure out where are the steady states of the system we have chosen, which is the nonlinear one.

For this, we have used the approximated vector field to solve the system for a larger time, with all initial points $x_0$. All points $x_0$ converged into four steady states for a time $\Delta t = 5$, in the positions shown in Figure 8. Since this nonlinear system has multiple steady states, it cannot be topologically equivalent to a linear system.

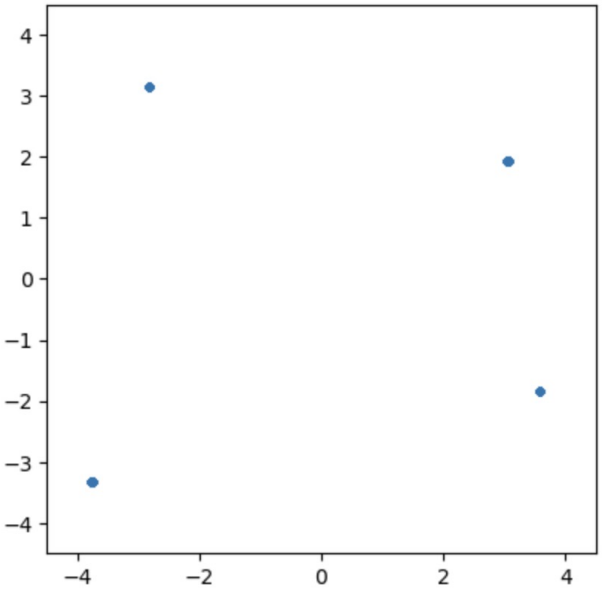


Figure 8: Steady states of the nonlinear vector field

## Report on task 4, Time-delay embedding

### Part 1: Embedding periodic signal into state space

In this task we embed a periodic signal into a state space where each point carries enough information to advance in time. After loading the `takens_1.txt` dataset we plot the datapoints in 2D-space and the first coordinate $x_0(t)$ against time (row number).

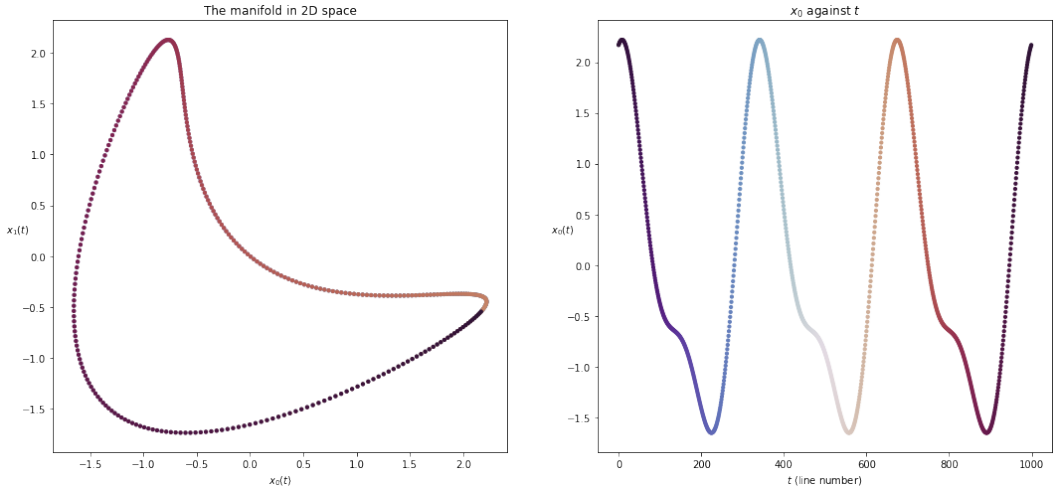


Figure 9: Signal datapoints in 2D-space and the first coordinate $x_0(t)$ against time.

Now, we take the coordinate $x_0$, and plot it against its delayed version, $x_0(n+\Delta n)$, where $\Delta n$ is the number of lines by which the coordinate is shifted. In the first graph we plot the first coordinate $x_0(t)$ against its delayed version $x_0(t-\Delta n)$, where $\Delta n$ is the number of rows (time) by which $x_0(t)$ is delayed.

At the same time, according to Takens theorem we only need 2 more coordinates to plot that the 1d series $x_0(t)$ is sure to be embedded correctly. We can use a smooth diffeomorphism $\psi : \mathbb{R} \to \mathbb{R}$ defined as a shift operation

$$\psi(t) = t - \Delta n$$

and the map $E_{(\psi,x_0)} : \mathbb{R} \to \mathbb{R}^3$ defined as

$$E_{(\psi,x_0)}(t) = (x_0(t), x_0(\psi(t)), x_0(\psi(\psi(t))))$$

to embed $x_0(t)$ from $\mathbb{R}$ into $\mathbb{R}^3$.

In the second graph the three line correspond with the three embedded coordinates $x_0(t), x_0(\psi(t)), x_0(\psi(\psi(t)))$ against time $t$, and in the last is the embedded vector result plotted in 3D space.

We also plot the results for the second coordinate $x_1(t)$, along with which there are 6 coordinates have we used in total to sufficiently display the charateristics of the entire periodic manifold.

Results for this part are showing in figure 10 on the next page.

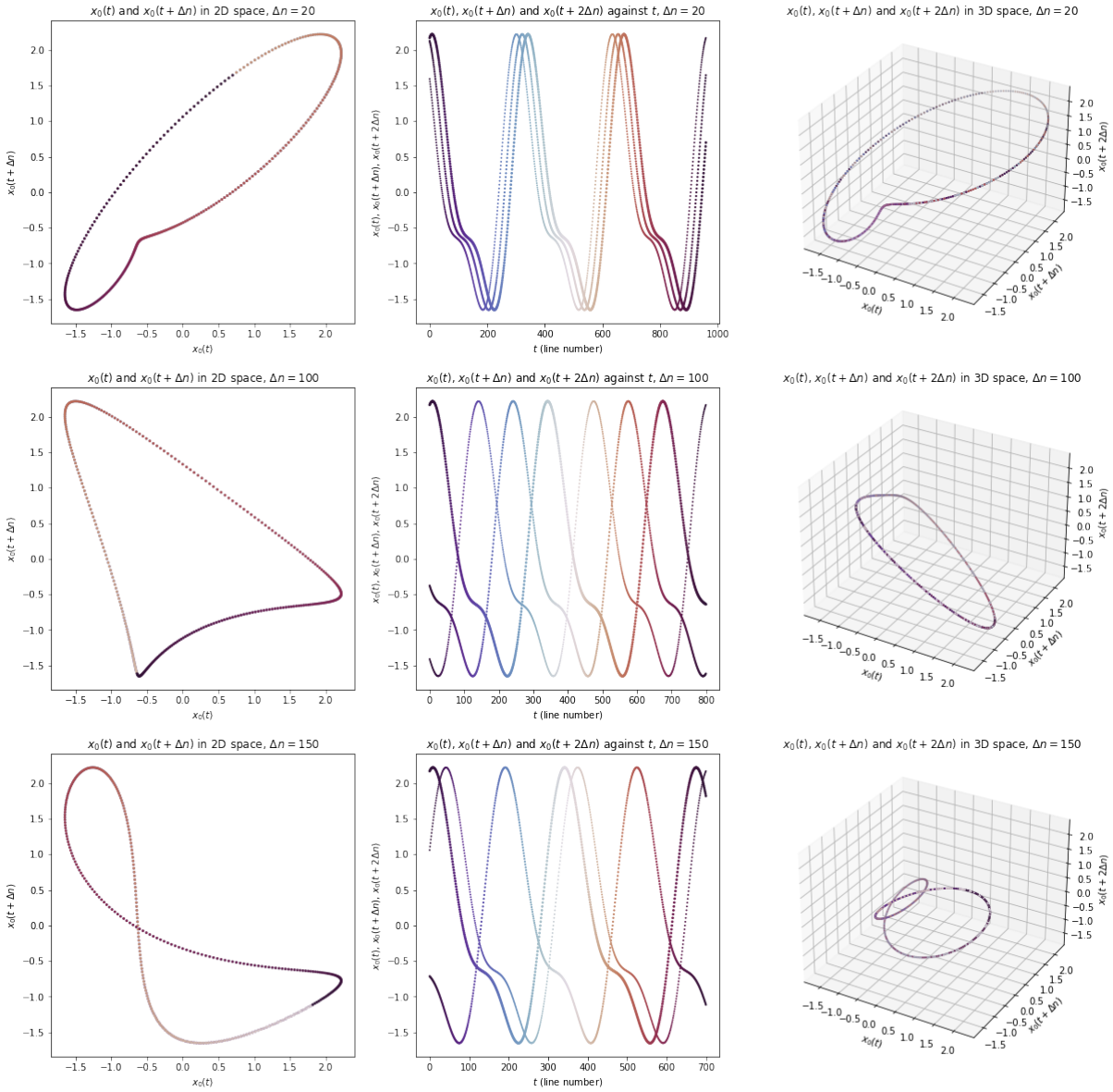


Figure 10: Shifted results for using $\Delta n = 20, 50, 100$

### Part 2: Approximating chaotic dynamics from time series

In this task we test the Takens theorem and our results on the Lorenz system. For a Lorenz system with $\sigma = 10, \rho = 28, \beta = \frac{8}{3}$, we split the calculated trajectories into its three coordinates and observe the behavior of each. By using Takens theorem and the shift operations similarly, we can embed each coordinates' projections of trajectory into 3 variables as aforementioned. The result is simulated with $T_{end} = 20$ seconds and simulation step length $\mathrm{d}t = 0.001$ second.

Results on variables $x$ and $z$ are showing in figure 11 and 12 on the following pages.

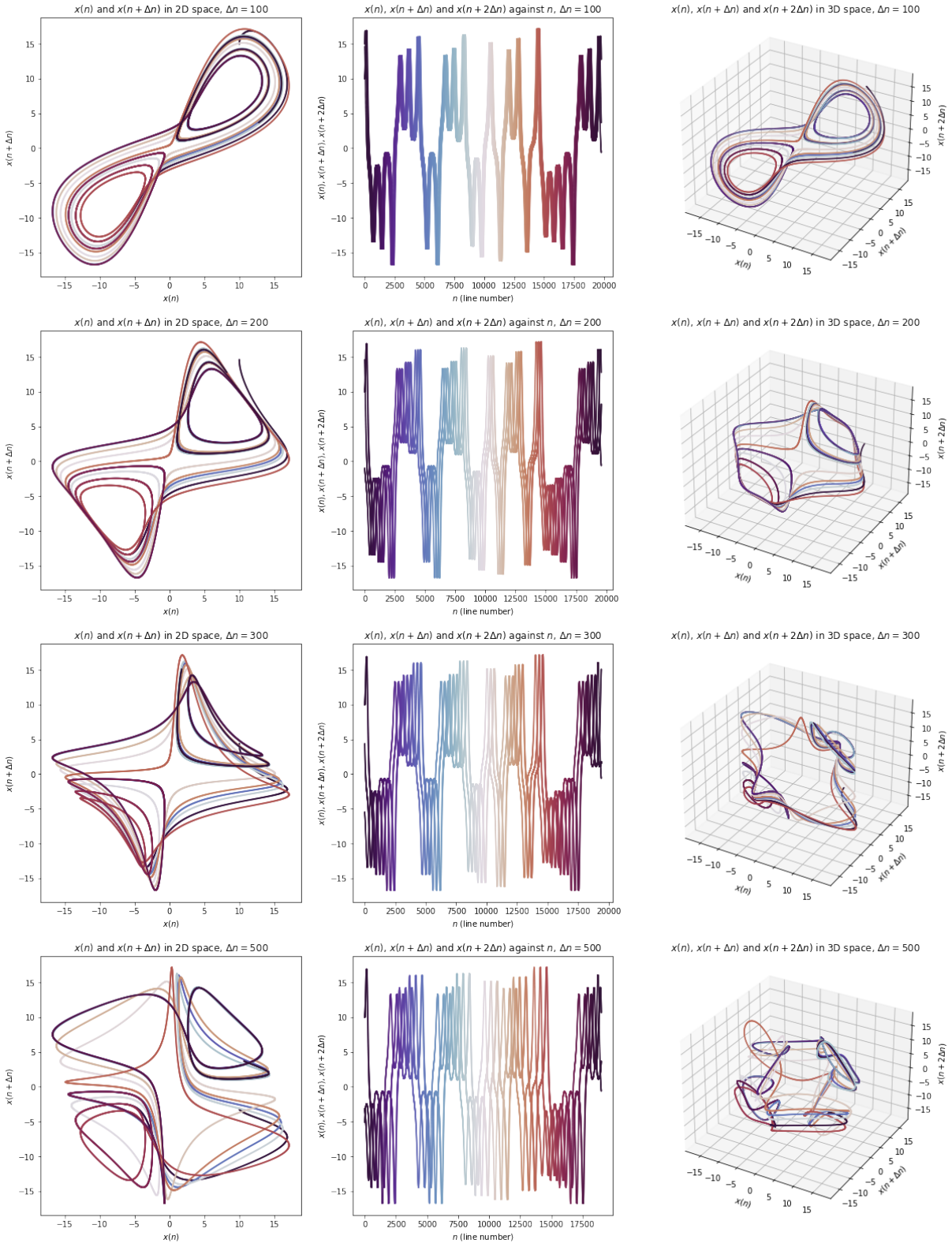


Figure 11: Shifted results for the trajectory's $x$ coordinate in the Lorenz system, using $\Delta t = 0.1, 0.2, 0.3, 0.5$

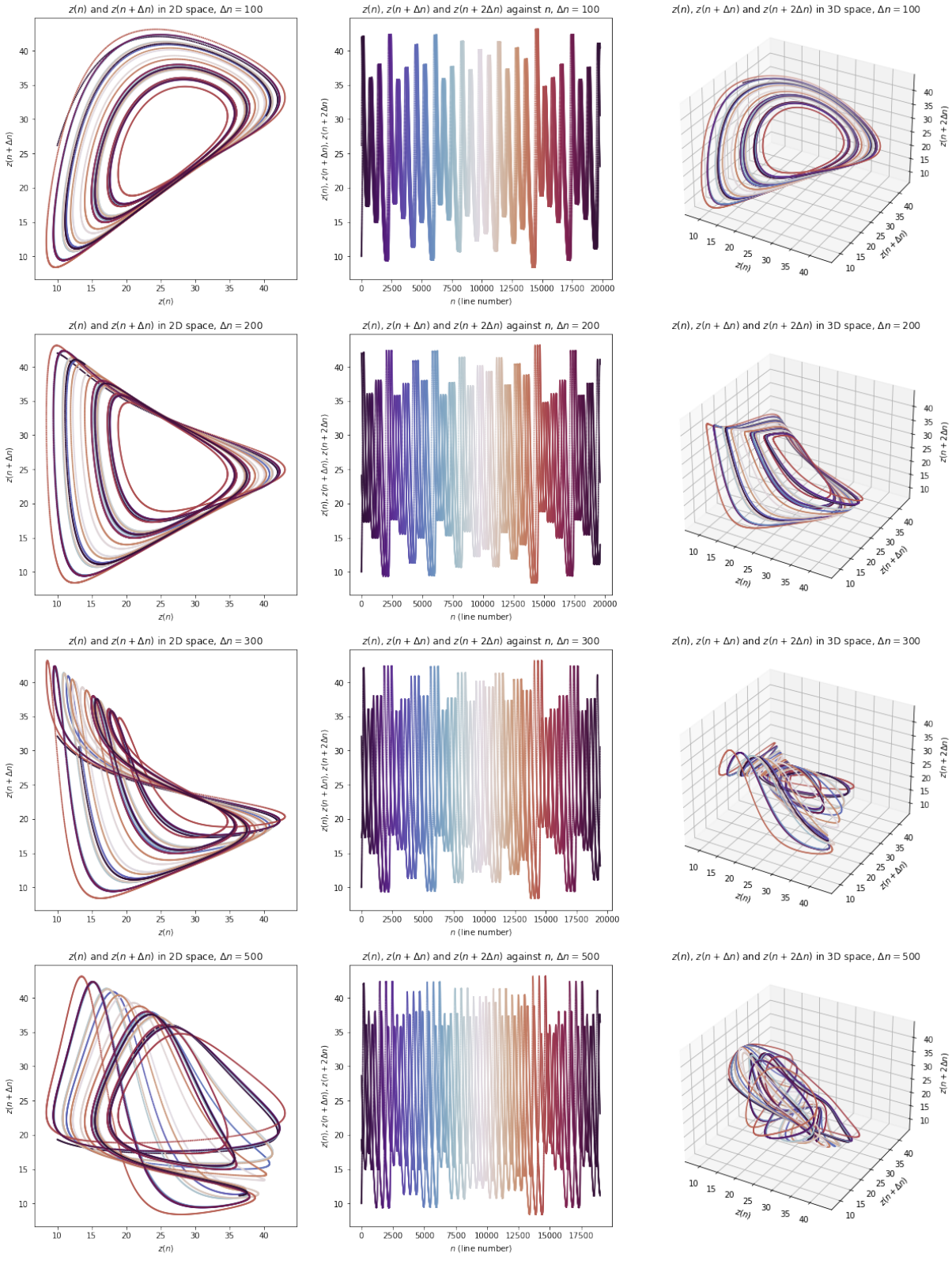


Figure 12: Shifted results for the trajectory's $z$ coordinate in the Lorenz system, using $\Delta t = 0.1, 0.2, 0.3, 0.5$

From the results we can tell that embedding works for $x$ and fails for $z$. The shape we got for $x$ with using suitable $\Delta_t$ is similar to the original butterfly shape and captured the characteristics of the two attractors, whereas for $z$ we only have a circling shape. We further certify this by testing for $y$ in figure 13 where we could

see that it also works.

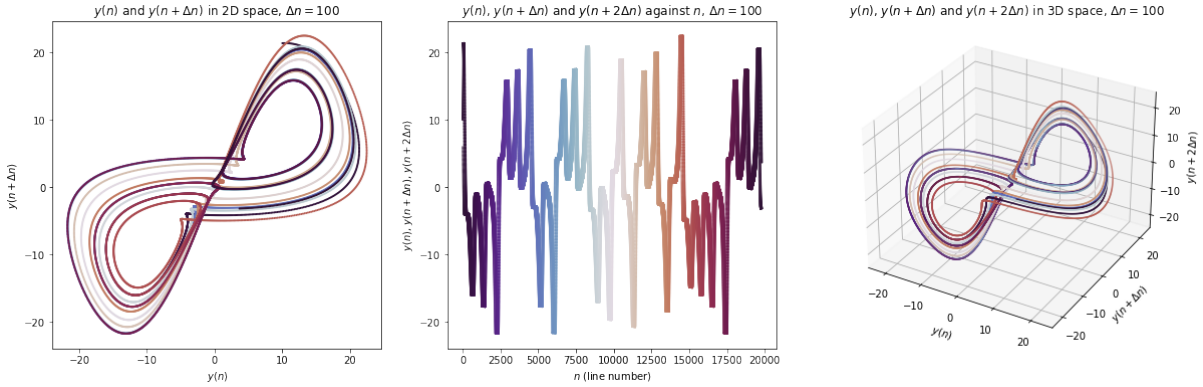


Figure 13: Shifted results for the trajectory's $y$ coordinate in the Lorenz system, using $\Delta t = 0.1$

By observing the middle subplot in the figures, we could tell the reason why time-delay embedding for the $z$ coordinate fail to capture the behaviorism with attractor is because that the $z$-axis component of the original butterfly trajectory does not contain information about the bifurcation behavior at the beginning. The system's position on $z$-axis keeps orbiting within a same range of values, whereas on $x$ and $y$ axis could it exhibit the behavioral pattern of alternately oscillating around two different attractors.

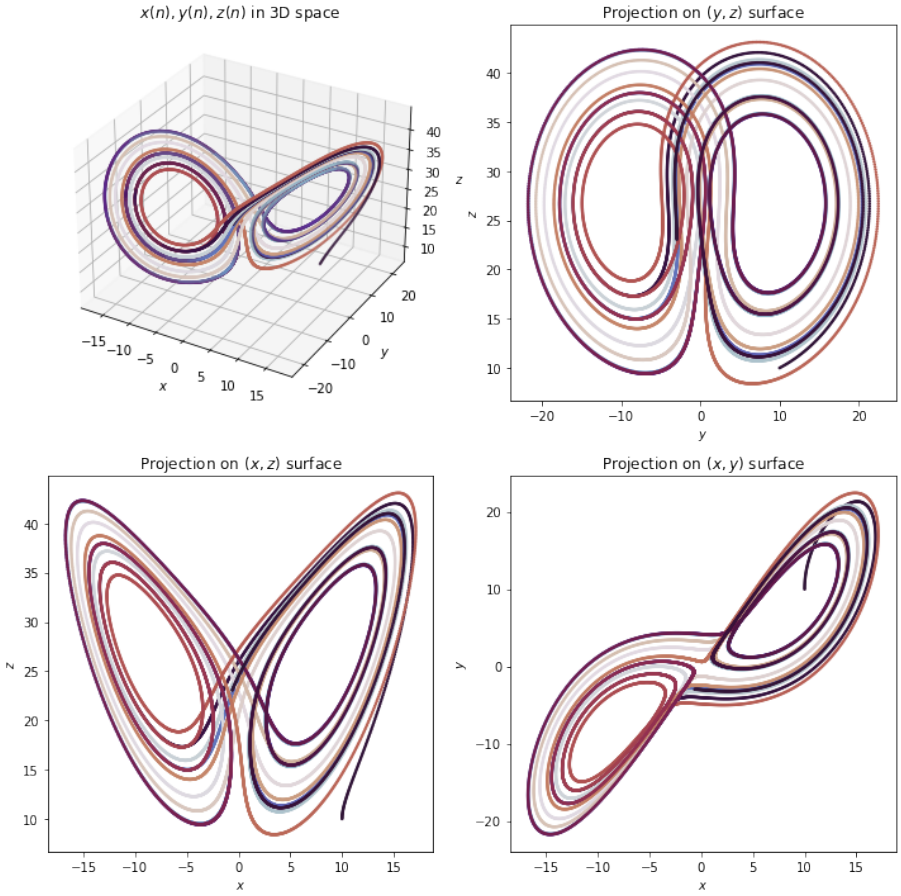


Figure 14: The trajectory and its projections in the Lorenz system

This reasoning can also be observed in the original 3D plot of the Lorenz attractor, where we can see that the underlying reason behind this phenomenon is that the centroids' position of the two attractors shares almost

the same $z$ value, but distinctively not $x$ and $y$. In this way the system cannot tell which attractor the system is currently engaged with by only using information on the $z$ axis.

**Bonus: Approximation of vector field for the Lorenz attractor**

In the bonus task we take the shifted $x$-coordinate to approximate the vector field $\hat{\nu}$ of the Lorenz-attractor, and plot the shifted vector field from its starting positions in a quiver plot. After concatenating the values of coordinated $x(t), x(t+\Delta t)$ and $x(t+2\Delta t)$ together we calculate the vectors pointing from data points to their corresponding targets, which is shifted using another $\Delta n = 1$. Hereby the process behaves as the computation of the approximation through the radial basis functions.

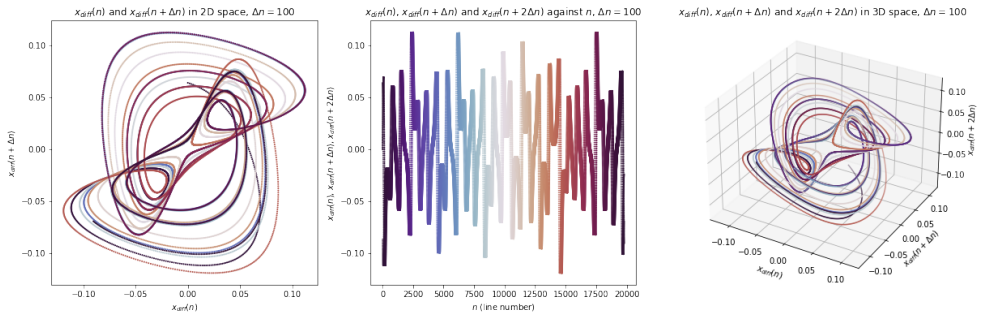

Figure 15: The medium results of the vectors pointing from data points to corresponding targets

We proceed creating a quiver plot showing reconstructed vector field in 3D space from their starting locations, obtaining similar graph to the embedding of the attractor with the $x$ coordinate.

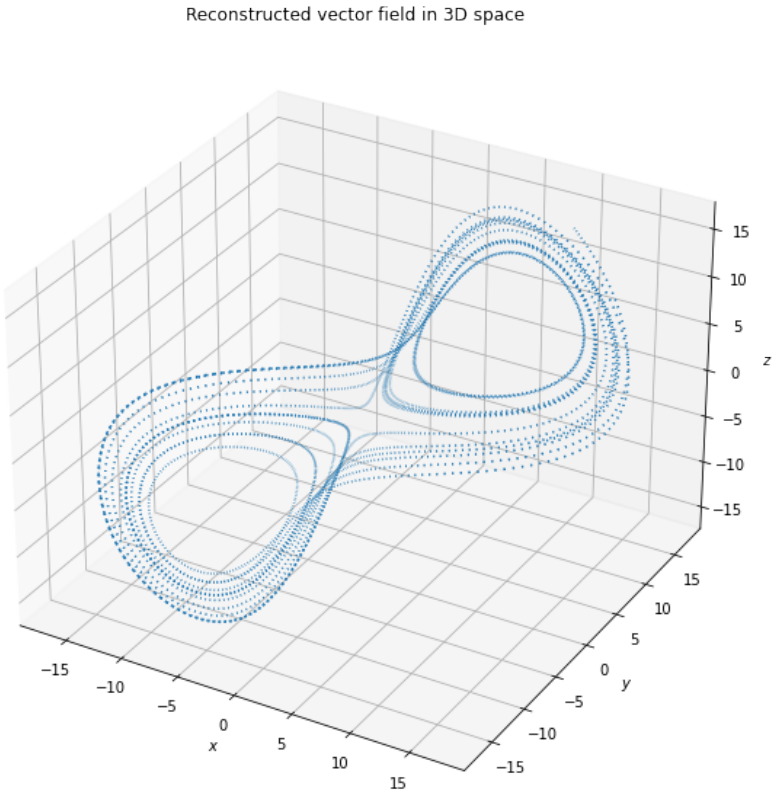


Figure 16: The reconstruction of the $x$ coordinate time delay embedding of the Lorenz attractor

**Report on task 5, Learning crowd dynamics**

The last task is centered on learning a dynamical system describing the people flow in nine different areas of TUM Garching, containing the MI Building and the 2 mensas. The given database describes the system over the course of 7 days and is divided as follows: it has a first column containing the time index (which is simply a natural number going up one by one and starting from 1) and 9 other columns containing the number of people per area at that certain time.

**Part 1:**

After loading and exploring data we already know the dataset being periodic with the same pattern and no parametric dependence, therefore can be reduced to become 1-dimensional. Following Takens theorem we can assert to construct an embedding with 3 dimension to get a reasonable state space.

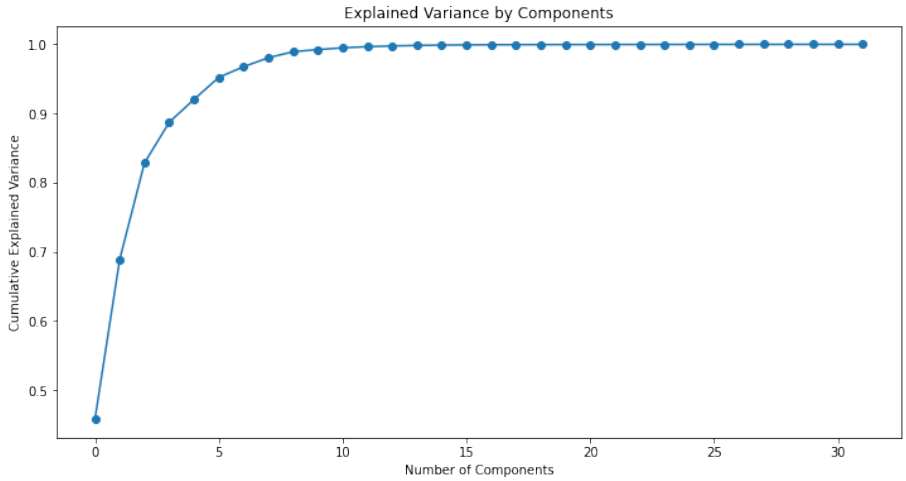


Figure 17: The cumulative explanined variance using different number of principle components

For PCA, we test the cumulative explained variance by using different number of components, so as to select a most suitable parameter. By which we can see that only using 3 principal components can we represent the state of the system with 83% of the energy now.

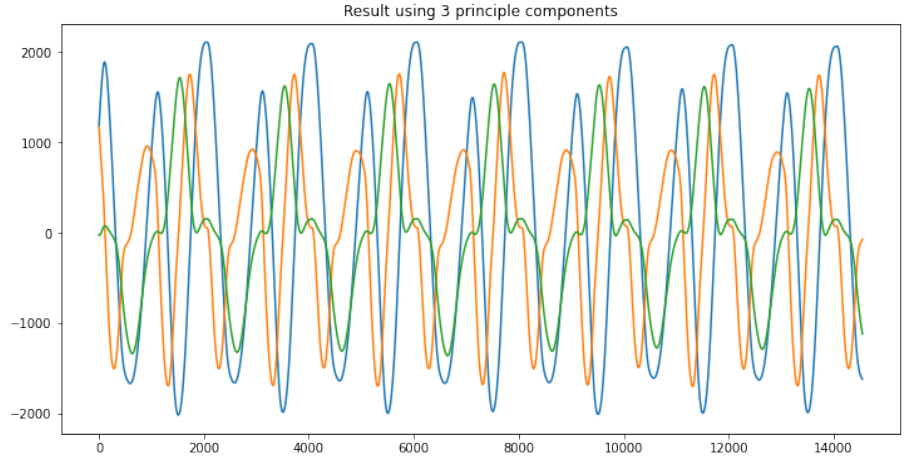


Figure 18: The PCA embedded result with 3 principle components

We can also plot the result in 3D space with each dimensions representing one principle component.

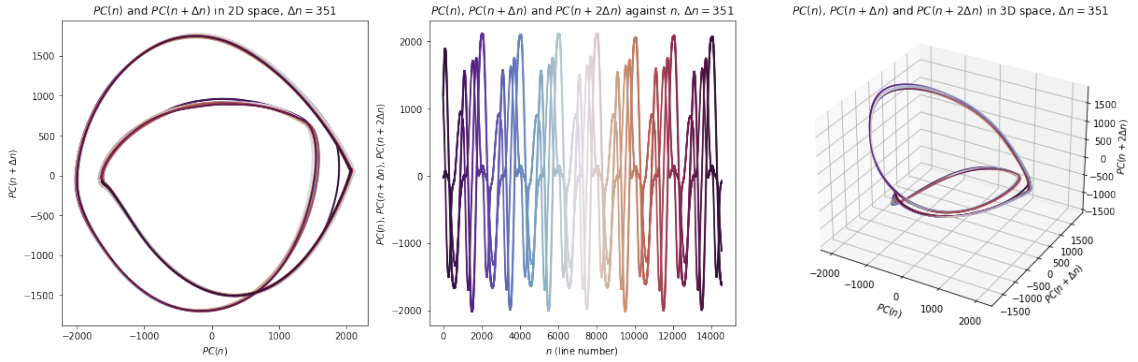


Figure 19: The PCA embedded result with 3 principle components in 3D space

**Part 2:**

For the second part we color the data points by all measurements taken at the first time point of the delays into 9 plots where all the points will be in the same position, only the color changes. This can be achieved by passing different columns as coloring parameter for plotting scatter points.

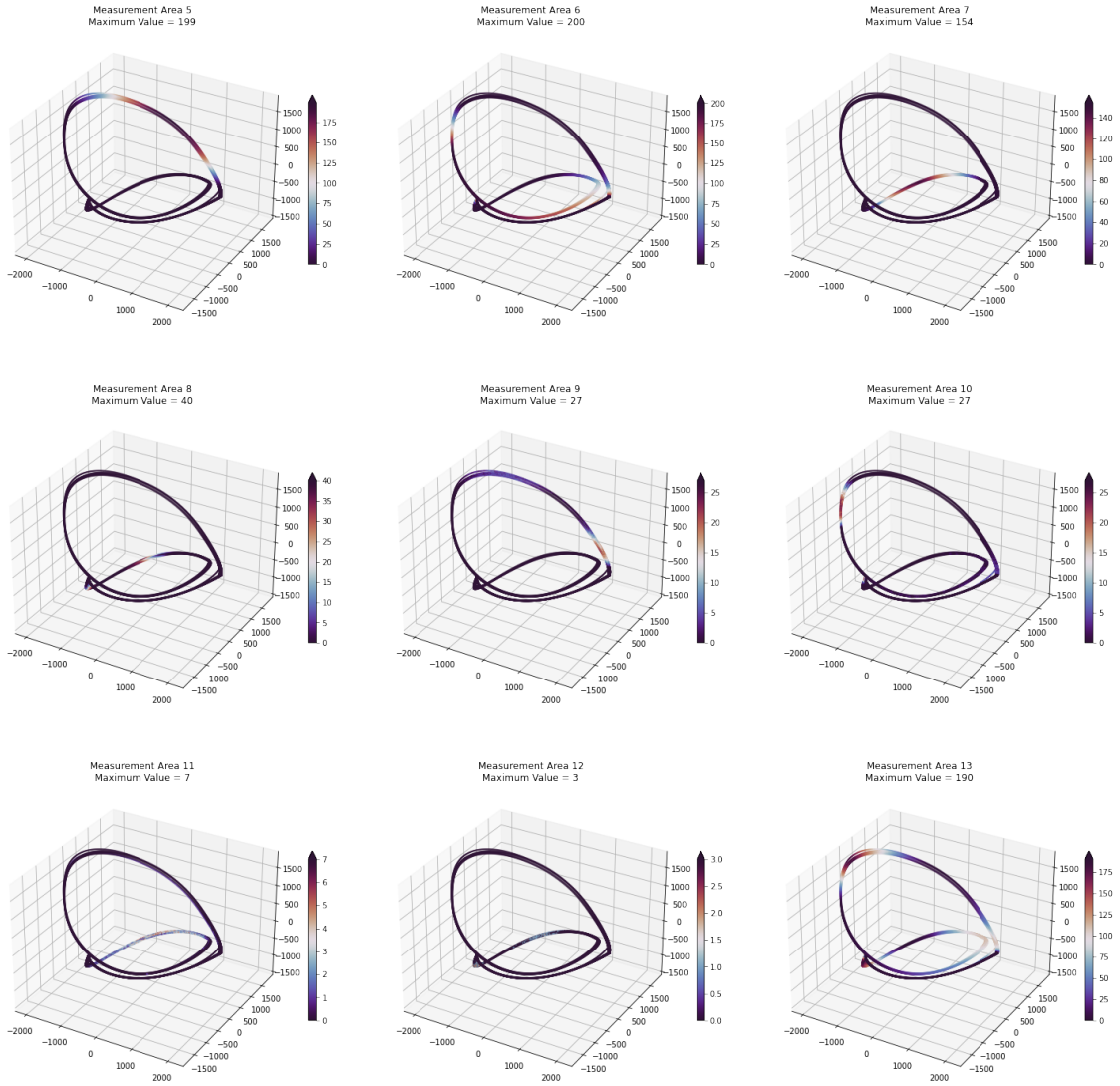


Figure 20: The PCA embedded result with 3 principle components in 3D space

**Part 3:**

In this part we want to learn the dynamics on the periodic curve we embedded in the principal components above by computing the arclength of the curve in the PCA space and then approximate the change of arclength over time.

By observing figure 17 we can tell that a periodicity around every 2000 time step (row numbers) exists for all three components. Hence we calculate the velocity of the arc length by determining how fast the system advances over the PCA space at every point in the space, and then approximate the change in arclength over time. We do this by comparing each time step with the previous one, and use our previously implemented RBF to approximate the velocity. As the data is perodic we only need to do this for the first 2000 data points.

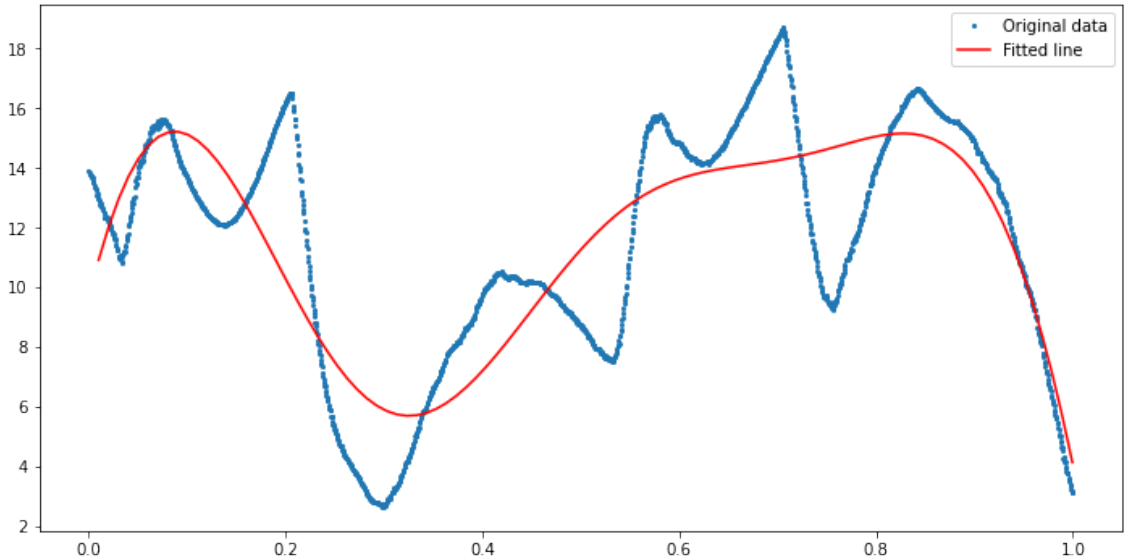


Figure 21: The velocity (vector field) result to arclength, with RBF approximation

We tested the combinations of different $L$ and $\epsilon$ and selected the current $L = 100, \epsilon = 0.2$ as it could generate sufficient result efficiently.