Report for exercise 5 from group H

Tasks addressed:	5
Authors:	Ahmad Bin Qasim (03693345) Kaan Atukalp (03709123) Martin Meinel (03710370)
Last compiled:	2020–01–08
Source code:	https://gitlab.lrz.de/ga53rog/praktikum-ml-crowd

The work on tasks was divided in the following way:

Ahmad Bin Qasim (03693345)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
	Task 5	33%
Kaan Atukalp (03709123)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
	Task 5	33%
Martin Meinel (03710370)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
	Task 5	33%

Report on task 1, Approximating functions

Part 1:

For part 1 we loaded the linear data from the "linear_function.txt" file and tried to approximate the function linearly. We used a linear regression model to minimize the mean square error. For this problem there exists a closed form solution to compute the matrix A which contains the parameters to map the input to the output with a minimal mean square error to the original function, we want to approximate. The mean squares error problem is defined by following equation:

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

Figure 1 shows the original linear data in blue and its linear approximation in orange. It can be easily seen that all data points are laying on the linear straight which was computed with the closed form solution. The data is linear so the function can be approximated perfectly.

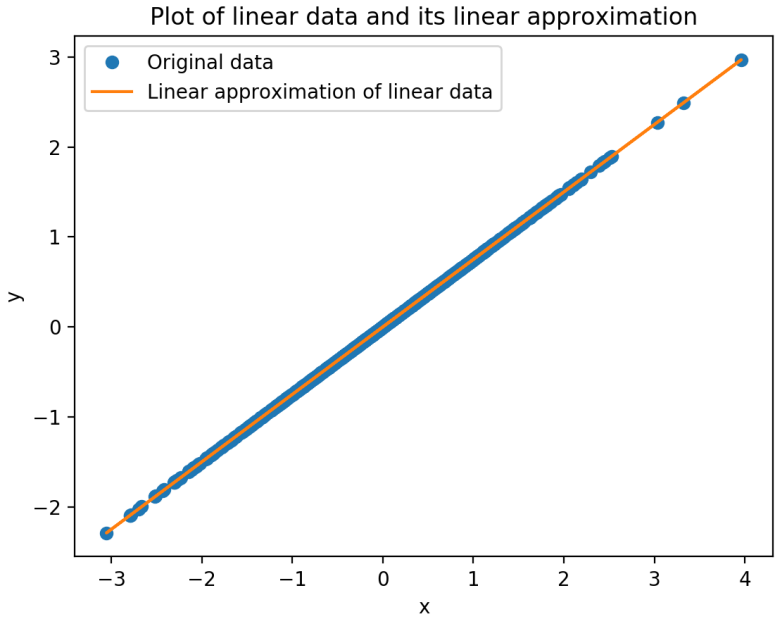

Figure 1: Plot of the linear data and its linear approximation

Part 2:

In the second part we use a nonlinear dataset from the "nonlinear_function.txt" file and tried to approximate the function in a linear way again. Figure 2 shows then original data in blue again. The linear approximation of the function is shown in orange. It can be easily seen that the linear approximation fits very bad. The reason for that is that the function is nonlinear and we try to approximate it linearly.

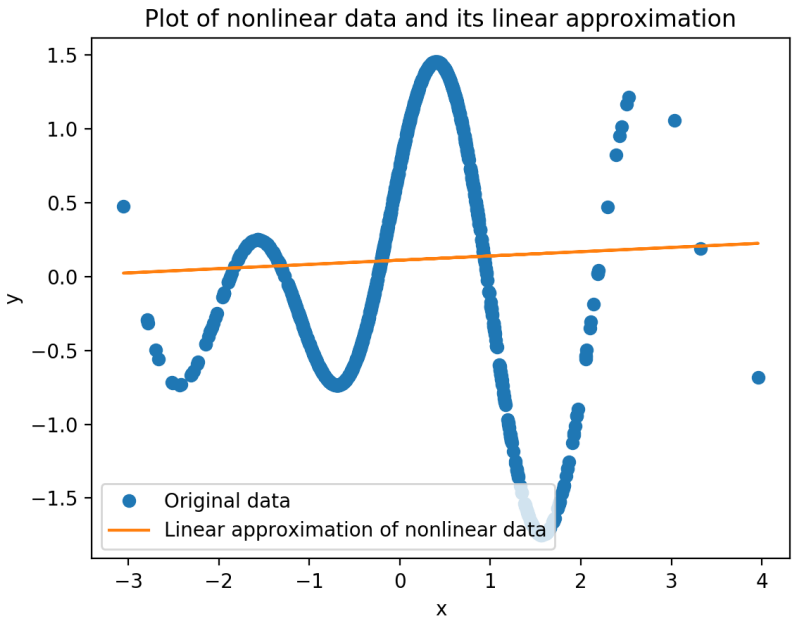

Figure 2: Nonlinear data and its linear approximation

Part 3:
After we tried to approximate the nonlinear dataset with a linear function in part 2, we now try to approximate the unknown nonlinear function with a combination of radial basis function:

$$\phi_l(x) = \exp(-||x_l - x||^2/\epsilon^2)$$

x_l is the center of the basis function and usually just a randon point of the data set. There is one x_l for each radial basis function. We have to choose how many basis functions L we use to approximate the nonlinear function. Besides, we have to choose ϵ appropriately. Figure 3 shows the original data in blue and the approximated function which makes use of radial basis functions in orange. It can be easily seen that the approximation is by far better than the linear approximation of part 2.
In general the approximation fits the original function quite well. We chose ϵ according to the ϵ in the diffusion maps task of the previous exercise sheet. This means we computed the distance matrix among all the x values and took the maximum value for ϵ. This meaximum value is multiplied by 0.05. Besides, we use $L = 15$ radial basis functions for the nonlinear approximation.

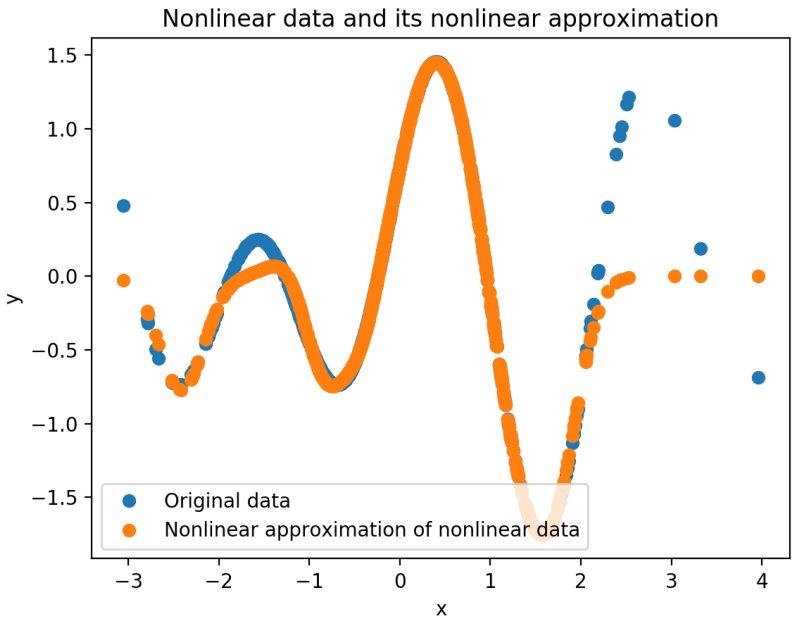

Figure 3: Nonlinear data and its nonlinear approximation with radial basis functions

It is not a good idea to use radial basis functions to approximate a linear function such as the linear data set of A. You can use the radial basis functions to approximate a linear data set, but you should use a linear approximator, because the computational cost of the radial basis function is higher due to the fact that the distance matrix for all x has to be computed to determinate ϵ. Besides the parameters of L and ϵ have to be chosen appropiately to obtain a good approximation.

Report on task 2, First step of a single pedestrian

Part 1:
We use the pandas read_csv function to read the text files containing two dimensional x0 and x1 data. We estimate the vector fields $v^{(k)}$ for each x0, using the finite-difference formula, given hereby:

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

Then we approximate the matrix A such that:

$$v(x_0^{(k)}) = v^{(k)} = Ax_0^{(k)}$$

An interesting thing to note is that, bias b is not added in this equation. This will have implications to the prediction later on. It will be mentioned in part 3.

For this purpose, we use the linear approximator implementation from task 1. As mentioned before the goal is to estimate the matrix A, such that the mean square error is minimized. This can be represented through the following equation:

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

Part 2:
After fitting the linear approximator, we predict the vector fields, $\hat{v}^{(k)}$ for each x0. Then we use $\hat{v}^{(k)}$ with $\Delta t = 0.1$ to calculate $\hat{x1}^{(k)}$ using the formula.

$$\hat{x1}^{(k)} = \hat{v}^{(k)}\Delta t + x0^{(k)}$$

We compare $x1^{(k)}$ and $\hat{x1}^{(k)}$ using the mean squared error formula and obtain the value of: 1.0532185339804091e-13

Part 3:
We set $x0 = (10, 10)^{(0)}$ and then estimate $\hat{v}^{(0)}$. We calculate $\hat{x1}^{(0)}$ using the equation given in part 2. Then using the corresponding $\hat{x1}^{(k-1)}$, we calculate the vector field $\hat{v}^{(k)}$ for it. We repeat this process for $T = 100$. We use steps of $\Delta t = 0.1$, so the total number of iterations are $\frac{T}{\Delta t} = 1000$. As seen in the figure 4, the motion stops at (0,0). We think that reason for this is that, we do not add bias term b to the linear approximator equation. Hence when $x = 0$, then the matrix A has no effect on $\hat{x}$.

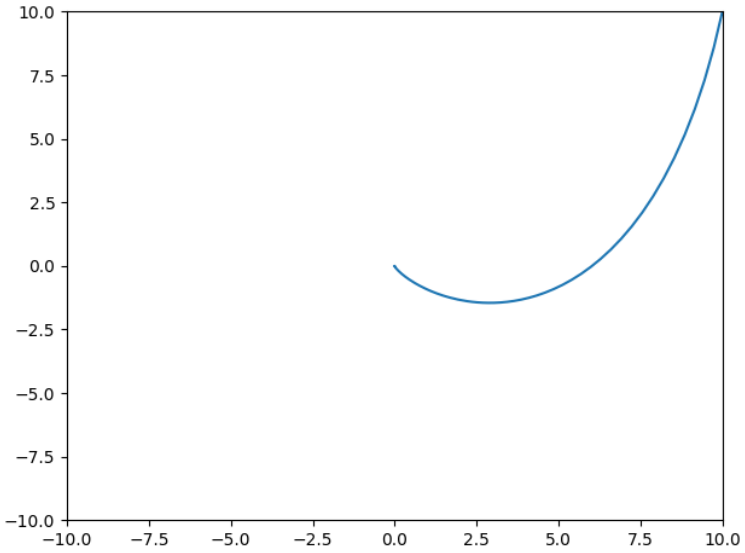

Figure 4: Trajectory of the motion

The phase portrait is given hereby:

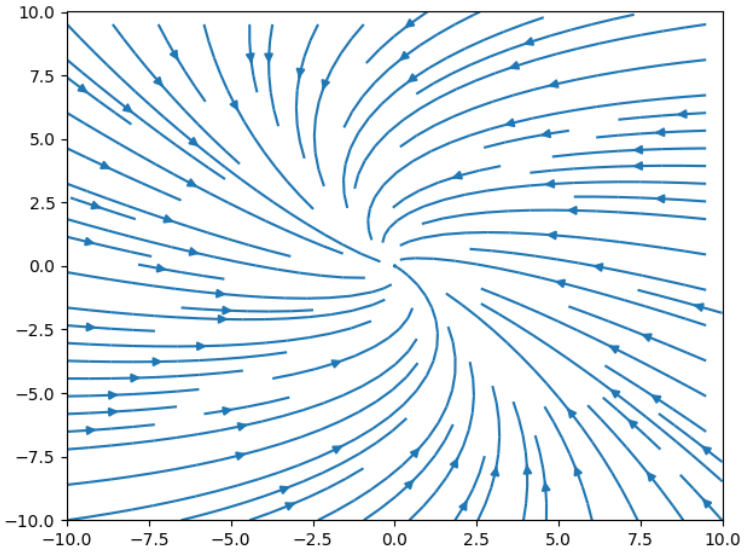

Figure 5: The phase portrait

Report on task 3, Approximating nonlinear vector fields

Part 1:
At first we compute a vector $\hat{v}^{(k)}$ for every intial point $x_0^{(k)}$ with the equation:

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

We use the linear approximator to approximate the vector field and obtain A. After getting A we approximate $\hat{x_1^{(k)}}$ and compute the mean squarred error between the aproximated and the known end points for a chosen $\Delta t = 0.5$. The mean squared error is 75.54.

Part 2:
Part 2 is similar to part 1, but this time we approximate the vector field with radial basis functions. We choose $L = 10$ and take for ϵ the biggest value of a distance matrix between all initial points x_0. The mean squared error is 0.24.
The mean squarred error is extremely low, so the performance of the nonlinear approximator is very good. Consequently you can see that the vector field is nonlinear. Otherwise the mean squared error of the linear approximation would be lower. You can see that the difference between both mean squared errors is high, because the mean squared error of the nonlinear approximated vector field is very low.

Part 3:
We use for this part the nonlinear approximator, because in our opinion the vector field is nonlinear. The nonlinear approximator shows better performance in comparison to the lineaer approximator. We use the approximated vector field to solve the system for a larger time with all intitial points x_0.
Figure 6 shows the end state of the system. You can see several steady states. The steady states are at the following coordinates:

x-coordinate of steady state	-2.92	3.22	3.65	-4.26
y- coordinate of steady state	3.13	1.85	-1.27	-3.65

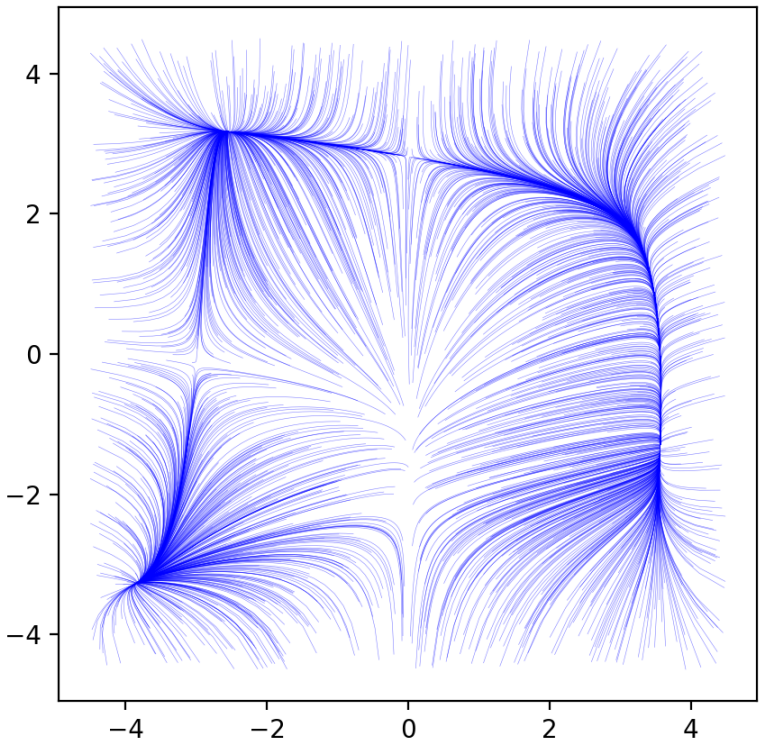

Figure 6: Predictions of x_1 for a larger time given initial points x_0

Report on task 4, Obstacle avoidance

Part 1:
We read the takens.txt file using the read_csv function from pandas library. We plot the first coordinate (column) of the data against time t. 7

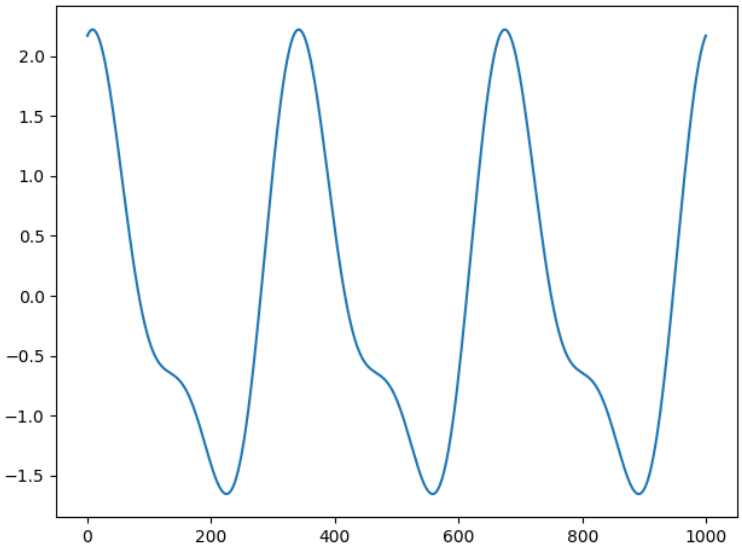

Figure 7: The plot of first coordinate against time t

Then we plot a time delayed version of the first coordinate against itself. The first coordinate is delayed by $\Delta n = 70$, because we observe the first plot and measure the time it takes for the first coordinate to complete a fourth of the periodic cycle. 8

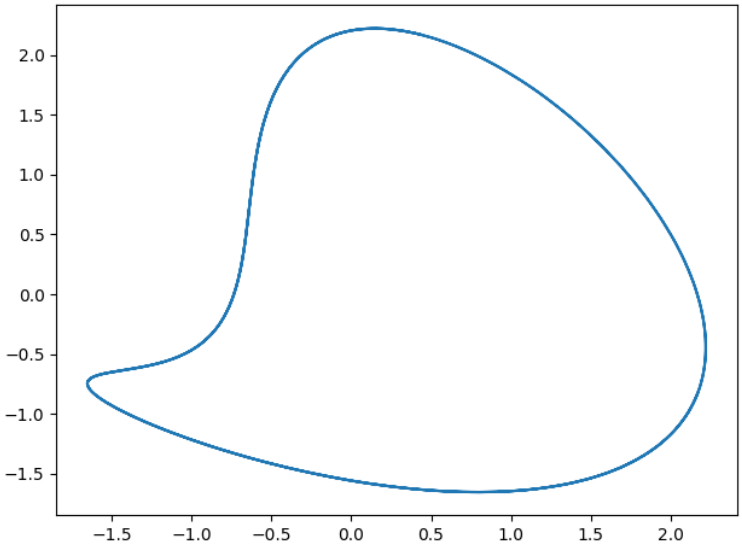

Figure 8: The plot of the first coordinate against the time delayed version of the first coordinate

We know that Takens has shown that embeddings with $d^e > 2n$ will be faithful generically so that there is a smooth map:

$$f : \mathbb{R}^{d^e} \to \mathbb{R}$$

where d^e is the embedding coordinates and n is the number of coordinates of the data. Hence, as the data has two coordinates, so according to Takens, atleast 5 time delayed embedding coordinates will be needed to make sure that the periodic manifold is embedded correctly.

Part 2:
We use the lorenz attractor implementation from Exercise 3 in this part. Using the solve_ivp function from the Sklearn library we calculate the x, y and z coordinates of the trajectory, after setting the initial position as (10, 10, 10). 9

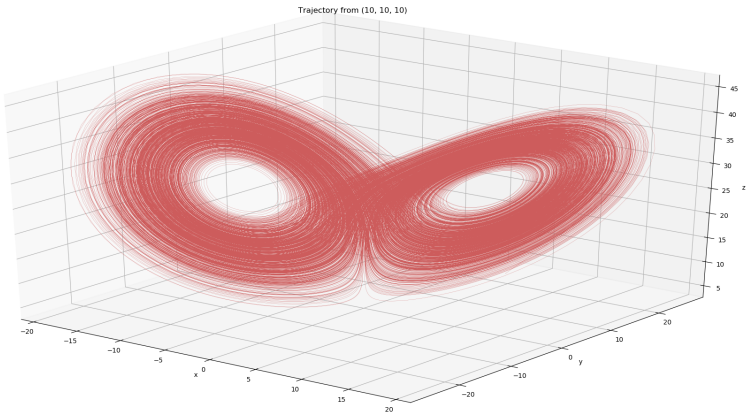

Figure 9: Plot of the lorenz attractor trajectory

Then we obtain the x-coordinate as $x0 = x(t)$ then we plot x0 against $x1 = x(t + \Delta t)$ and $x2 = x(t + 2\Delta t)$. We set $\Delta t = 18$. 8

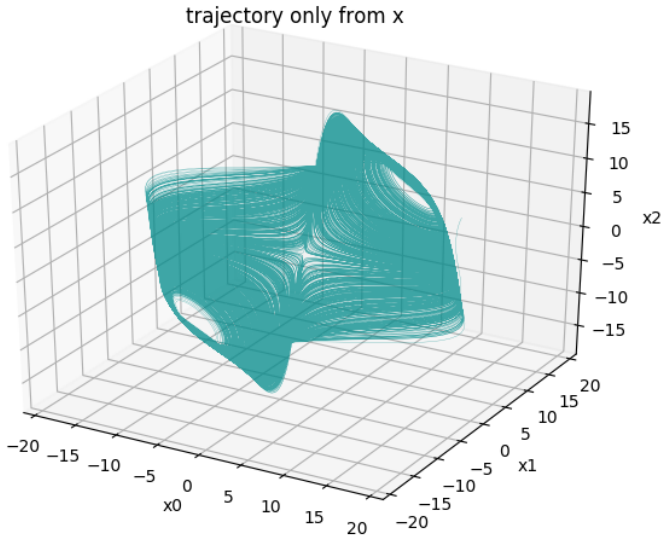

Figure 10: Plot of x0, x1 and x2 in a 3 dimensional space

Lastly, we obtain the z-coordinate from the solve_ivp result that we obtained earlier, as $z0 = z(t)$ then we plot z0 against $z1 = z(t + \Delta t)$ and $z2 = z(t + 2\Delta t)$. We set $\Delta t = 20$. 11

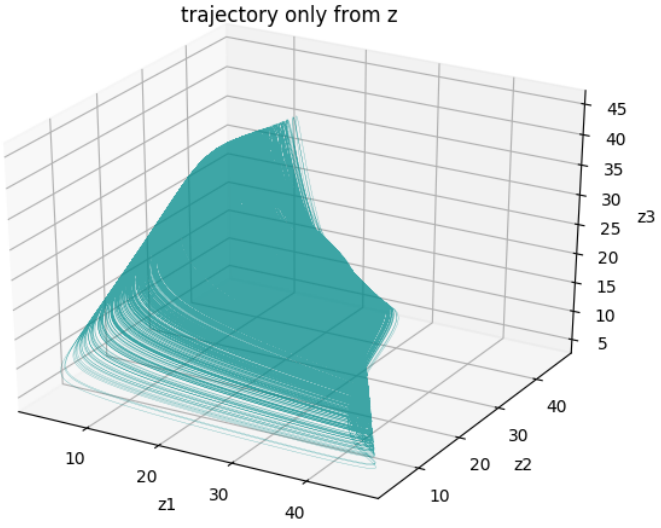

Figure 11: Plot of z0, z1 and z2 in a 3 dimensional space

Plotting z0, z1 and z2 does not result in a representative visualization of the original data, compared to the plot of x0, x1 and x2. The reason for this is that, lorenz attractor system consists of three ordinary differential equations. The differential equation responsible for the z-coordinate has the z-coordinate term in it but when using time delayed embedding, the subsequent time delayed coordinates z1 and z2 do not account for this. It results in the error propagating through the embedding.

Part 2, Bonus task:
Using the x0, x1 and x2 coordinates obtained in the previous task, we approximate the vector fields $v^{(k)}$ at each timestep using the finite-difference formula. Then we fit the non-linear approximator to predict the $\hat{v}^{(k)}$ given the embedding coordinates.

The next step is to use the fitted linear approximator as the vector field predictor and using the solve_ivp function from Sklearn library to generate the trajectories. The starting point is set as the first point of the embedding coordinates. The results obtained are not good. Even though the resulting trajectory loops, it does not resemble the trajectory of the training data.

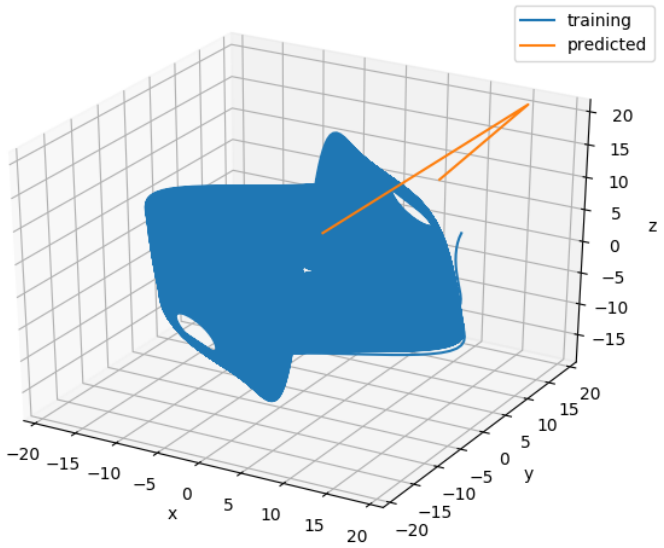

Figure 12: Comparison of the training and predicted data

Part 3:
We used the old implementation of pedestrian trajectory parser from exercise 3 to get the coordinates of pedestrians in the case when the path is not blocked by the object. As mentioned in the exercise we concatenate 200 time delay embedding into one data point and then using $\Delta t = 1$, we generate several data points. We obtain, 3551 data points from this process as the total number of data points in the original data was 3751. We use the PCA implementation from exercise 4 to obtain the first two principal components of the resulting data. 13

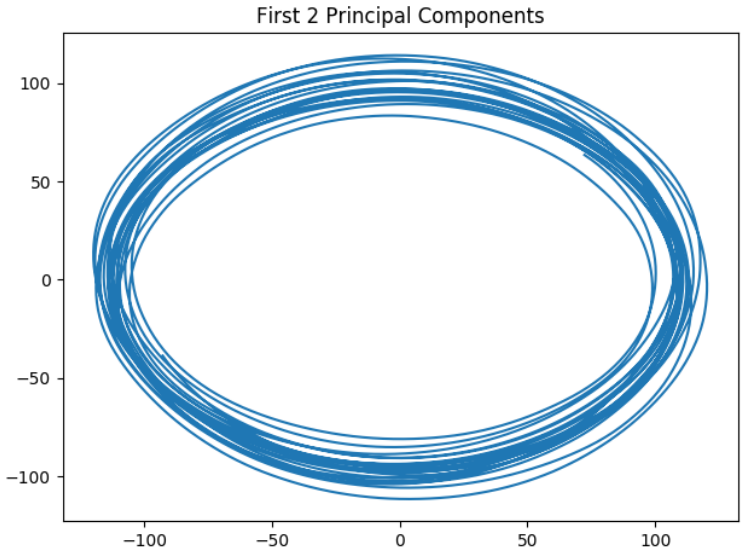

Figure 13: Plotting the first two principal components

-
- as mentioned in the exercise

Report on task 5, Tests

TEST1: RiMEA scenario 1 (straight line, ignore premovement time)
- not done, but citing RiMEA guidelines -

TEST2: RiMEA scenario 4 (fundamental diagram, be careful with periodic boundary conditions).
- test successful -

TEST3: RiMEA scenario 6 (movement around a corner).
- test successful -

TEST4: RiMEA scenario
- test successful -