Silvan Loew 11/15/2021

Assignment 2

1) The leaning aim of this assignment is to learn the motion model of the robot. In assignment 0 I saw that the nonlinear model was useful for local motion, but had drift for global motion, since the errors summed up over time. In this assignment, I try to learn and improve the local model, which was already good and try to find a global model. The algorithm used for it is locally weighted linear regression. Since local analysis of models is mostly a good enough (or at least a valid start point) approximation to come up with controllers for robots, I think that the locally weighted linear regression algorithm should lead to good results in the local model, and hopefully also for the global model. Thus, different data is used for the different goals. To get a global model, the input to the model is the state (x_t, y_t, θ_t) and input commands (v_t, w_t) at time t and the output is the state at time t+1 $(x_{t+1}, y_{t+1}, \theta_{t+1})$.

$$v_t, w_t, x_t, y_t, \theta_t \rightarrow x_{t+1}, y_{t+1}, \theta_{t+1}$$

For the local model, we only need the input commands (v_t, w_t) and heading angle θ_t to get the difference in the state $(\Delta x_t, \Delta y_t, \Delta\theta_t)$.

$$v_t, w_t, \theta_t \rightarrow \Delta x_t, \Delta y_t, \Delta\theta_t$$

We will compare the results of our algorithm to the nonlinear motion model from assignment 0:

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + dt \begin{pmatrix} v_t \cos(\theta_{t-1} + dt\ \omega_t) \\ v_t \sin(\theta_{t-1} + dt\ \omega_t) \\ \omega_t \end{pmatrix}$$

Figure 1 and 2 and table 1 show the performance of the performance of the motion model. One can clearly see that the global performance of the nonlinear motion model is not as desired and leads to a large mean of the position error as well as a large variance of the position error. The local error is indeed smaller but can still be improved.

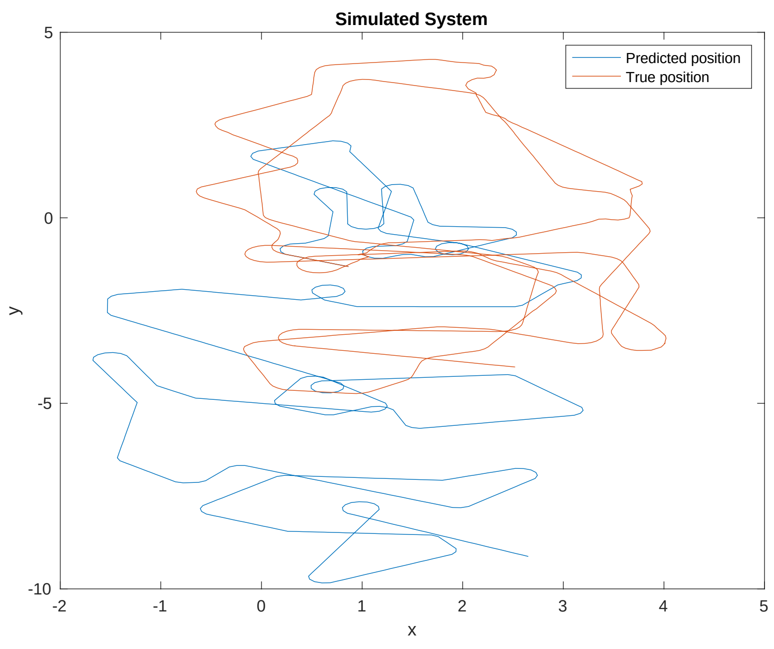

Figure 1: Nonlinear motion model position prediction vs true position.

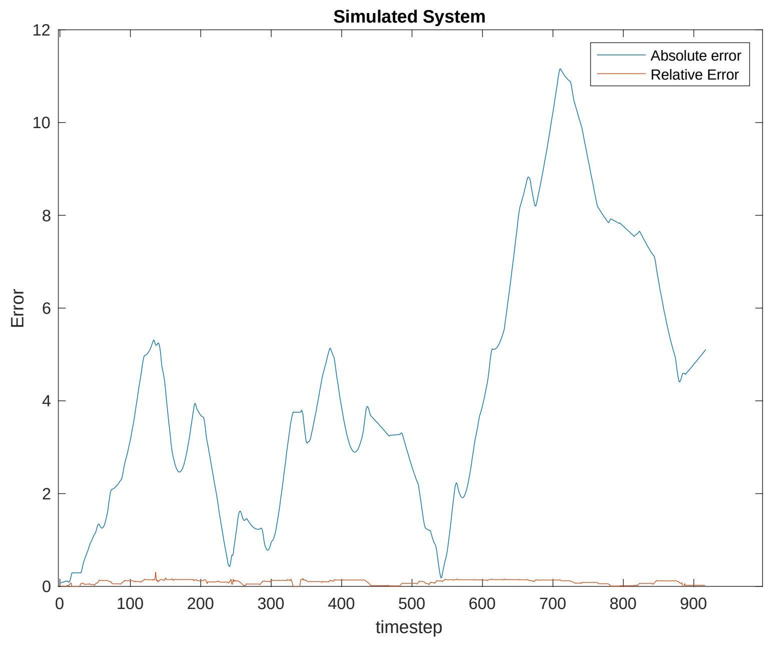

Figure 2: Nonlinear motion model global and relative position error.

Table 1: Nonlinear motion model global and local error

	GLOBAL ERROR MEAN	GLOBAL ERROR VARIANCE	LOCAL ERROR MEAN	LOCAL ERROR VARIANCE
NONLINEAR MOTION MODEL	4.2378	8.0151	0.0944	0.0023

To get the data uniformly sampled and not use the time difference dt as an input variable, I sampled the data (state and input commands) from the robot at a fixed sampling frequency. To make the dataset not too big and the compilation time of the code not too long, I decided to sample at 2 Hz.

2) As stated in the first part, I implemented the locally weighted linear regression algorithm. The adapts the base linear regression algorithm, where we have the input matrix X, the weight matrix β and the output matrix Y with the relationship:

$$X\,\beta = Y$$

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}$$

Where x_i are the i-th input datapoint and y_i the i-th output datapoints and β holding all linear parameters that map x_i to y_i.

If we weight the data according to a distance function $\Phi(d(x_i, x_j))$, we get the transformation

$$Z = W\,X$$

$$V = W\,Y$$

Where W is a diagonal matrix with the i-th element on its diagonal w_i

$$w_i = \sqrt{\Phi(d(x_i, x_q))}$$

$\Phi(d(x_i, x_j))$ was chosen as the Gaussian / RBF Kernel and therefore

$$\Phi(d(x_i, x_q) = \exp\left(-(x_q - x_i)^T \Sigma^{-1} (x_q - x_i)\right)$$

With sigma as a tuning parameter. Sigma was chosen to be a diagonal matrix for the algorithm. The bigger the element on the i-th row was chosen, only nearby points of the i-th input dimension were considered. The parameters for our experiments were tuned by trial and error until a desired performance was achieved compared to larger / smaller values.

To calculate the output y_q for a desired query point x_q, we can solve this system and get

$$y_q = x_q^T\,(Z^T\,Z)^{-1} Z^T\,V$$

The algorithms functionality is proved by estimating a noisy sine curve as shown in figure 3.

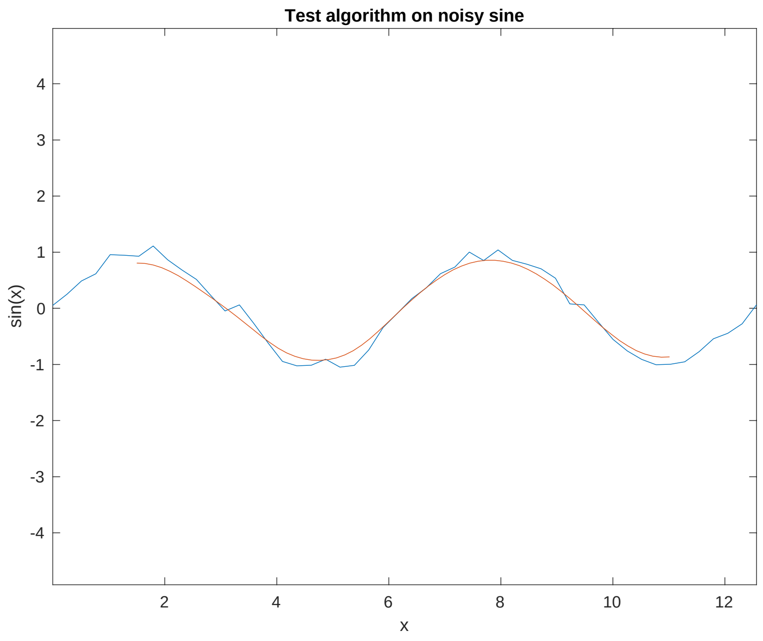

Figure 3: Noisy sine curve (blue) and estimated sine curve (red) from the noisy data.

3) This section analyzes the performance of the algorithm on the different datasets. The global method it shown first and afterwards the local method and the local method adapted with a nonlinear addition to it. Both methods were evaluated with either perfect state knowledge at each time step (meaning the input to the prediction model was the ground truth) or with only the estimated state (meaning the estimated state was used for the next time step as an input).

   Global model:

$$v_t, w_t, x_t, y_t, \theta_t \rightarrow x_{t+1}, y_{t+1}, \theta_{t+1}$$

   The estimated position with the global model with perfect state knowledge is shown in figure 4 and the global model without perfect state knowledge in figure 5. We can see from both figures that the perfect state knowledge leads to a small error globally as well as locally and not having this perfect state knowledge leads to an accumulation of the error of the time, and therefore only the relative position error is small. This can also be seen in the error plots in figure 6 and 7 as well as in table 2.

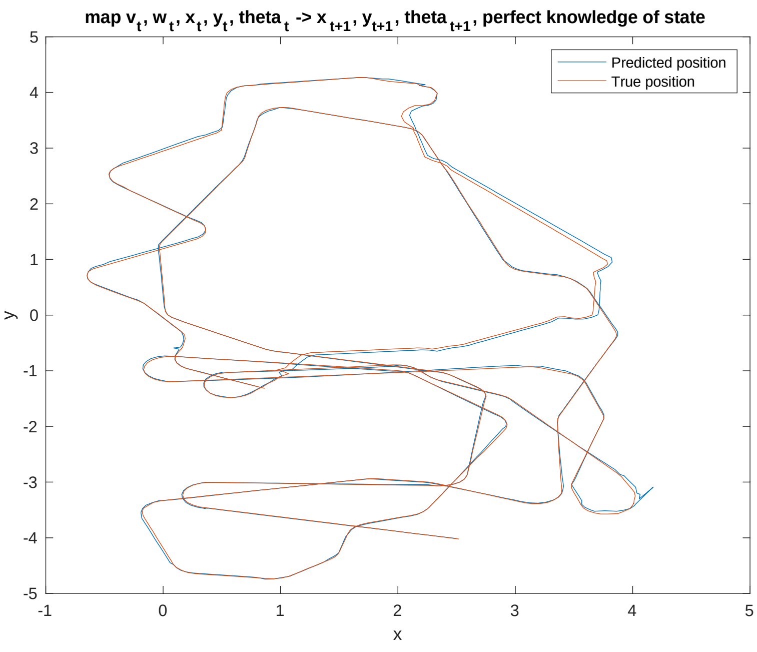

Figure 4: Global model with perfect state knowledge.

   If we compare the error plots in figure 6 and 7 and well as the mean and variance of the global and local error to the error of the nonlinear model, we can see that both models outperform the baseline. A reason for that could be that our global models are able to learn unmodeled dynamics of either the robot or the environment. Nevertheless, we can still see that the global model without perfect state knowledge struggles at some points locally.

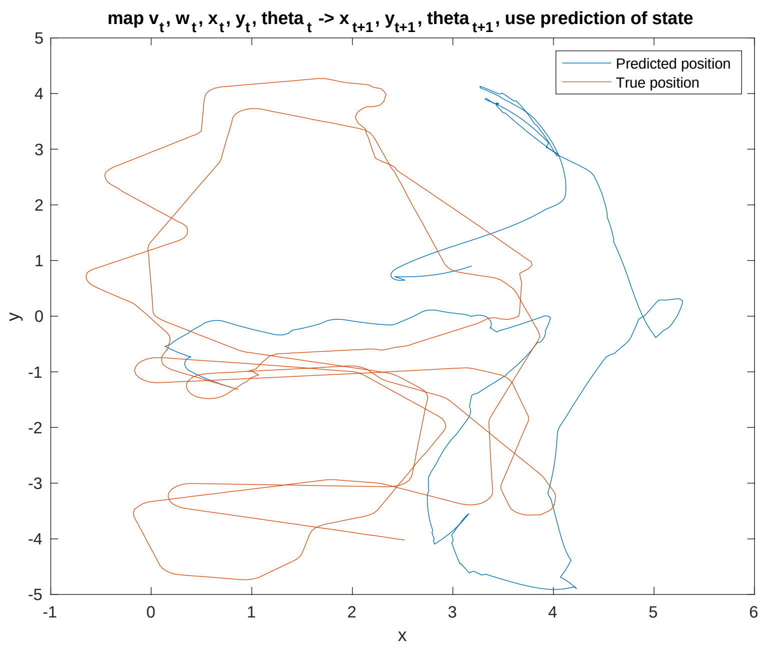

Figure 5: Global state model without perfect state knowledge.

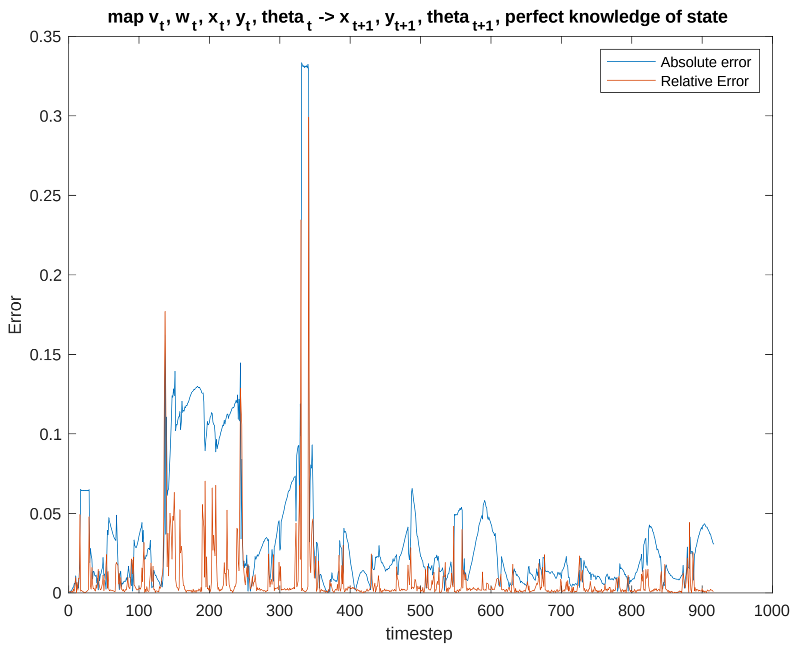

Figure 6: Global model with perfect state knowledge error plot.

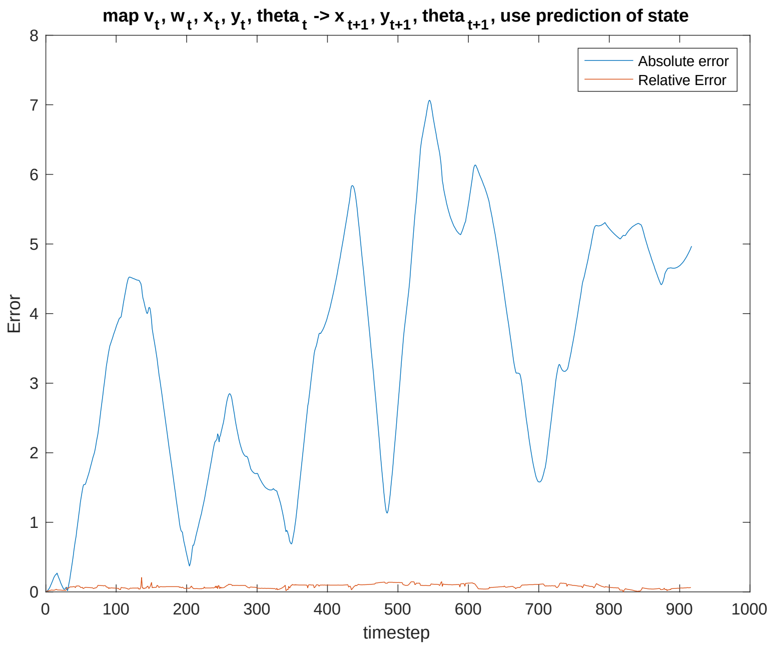

Figure 7: Global model without perfect state knowledge error plot.

Table 2: Global models local and global errors

	GLOBAL ERROR MEAN	GLOBAL ERROR VARIANCE	LOCAL ERROR MEAN	LOCAL ERROR VARIANCE
GLOBAL MODEL (PERFECT STATE)	0.0362	0.0022	0.0076	0.0004
GLOBAL MODEL (NOT PERFECT STATE)	3.4213	3.1035	0.0766	0.0009

Local model:

$$v_t, w_t, \theta_t \ \rightarrow \Delta x_t, \Delta y_t, \Delta\theta_t$$

The position plots of the local models are shown in figure 8 and 9. The plots of the errors are shown in figure 10 and 11 and the error measures are listed in table 3. In the plot in figure 8 as well as in the error plot in figure 10 one can see that the local model with perfect state knowledge struggled at one point globally and was able to keep the error constant everywhere else. This is the reason why the two global error measures are worse than the ones of the nonlinear model. The local error measures outperformed the one of the nonlinear model again, this time probably due to a better understanding of unmodeled dynamics of the robot.
The local model without perfect state knowledge has a poor global performance. This can be seen in all the plots as well as in the error measures. Its local performance is way better and even able to outperform the nonlinear model.

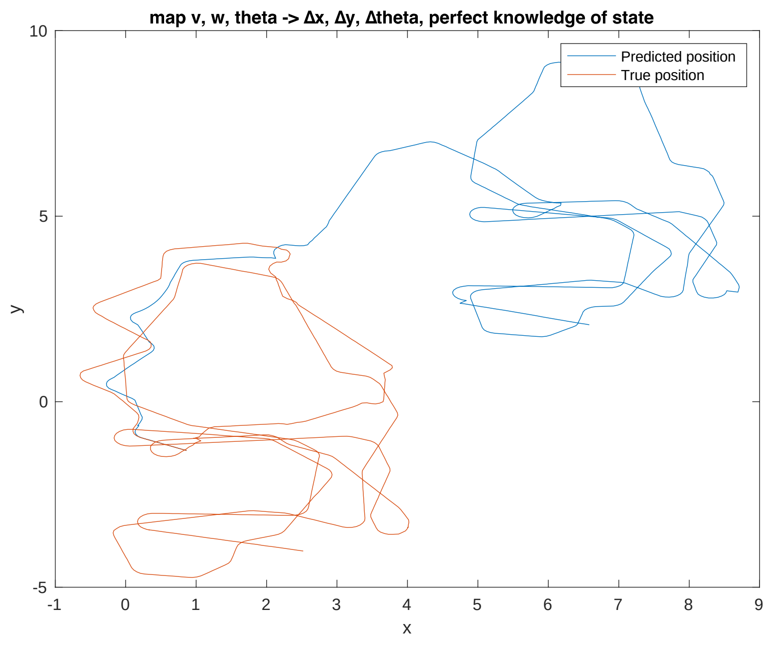

Figure 8: Local model with perfect state knowledge.

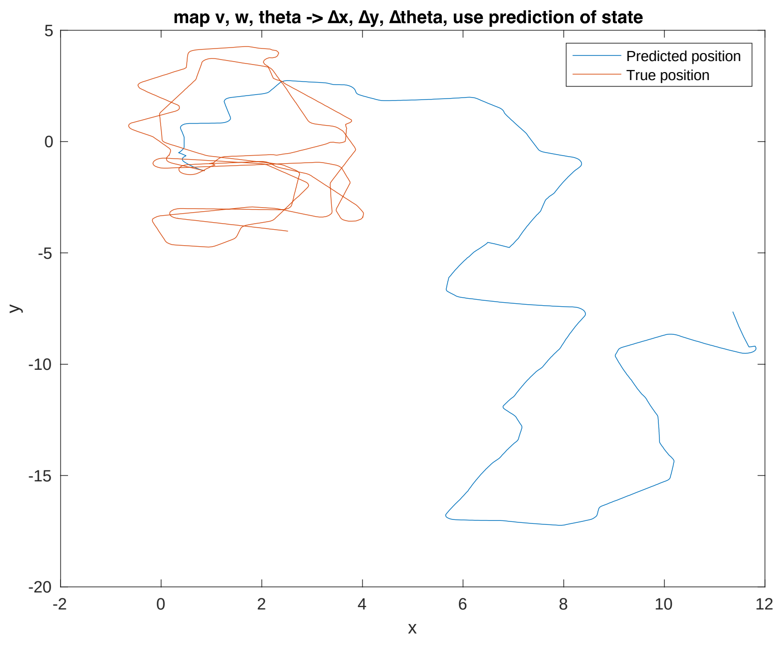

Figure 9: Local model without perfect state knowledge.

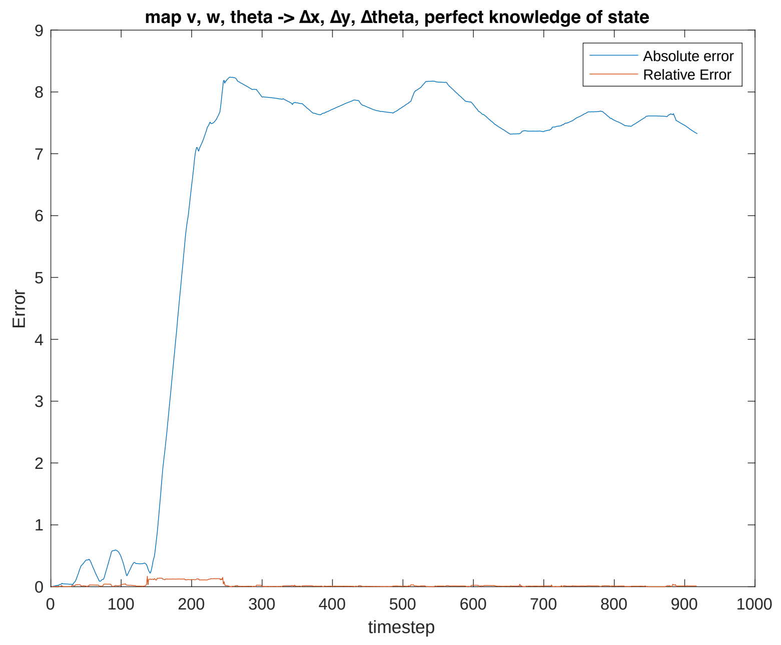

Figure 10: Local model with perfect state knowledge error plot.

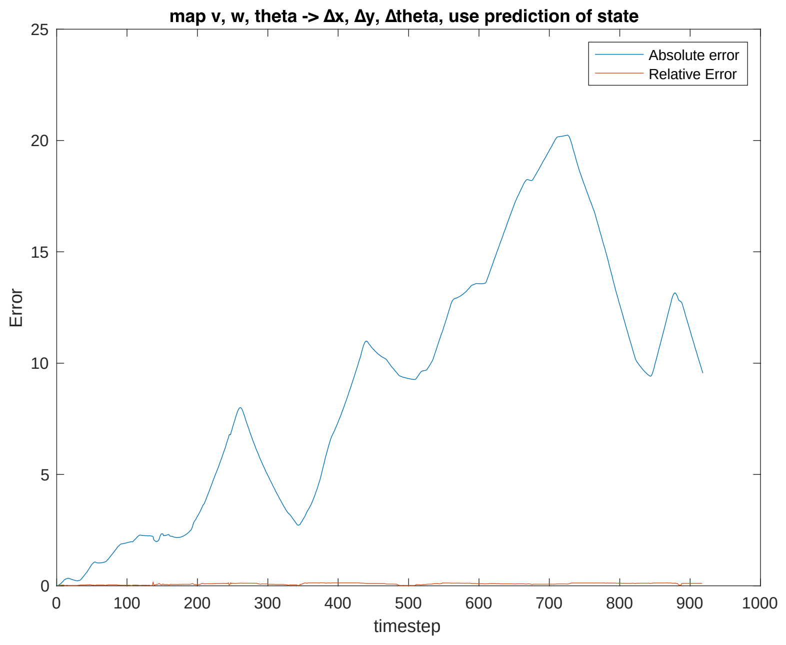

Figure 11: Local model without perfect state knowledge error plot.

Table 3: Local models global and local errors

	GLOBAL ERROR MEAN	GLOBAL ERROR VARIANCE	LOCAL ERROR MEAN	LOCAL ERROR VARIANCE
LOCAL MODEL (PERFECT STATE)	6.2652	8.0062	0.0242	0.0013
LOCAL MODEL (NOT PERFECT STATE)	8.9193	33.3578	0.0861	0.0013

Nonlinear input:

$$v_t, w_t, \theta_t, \sin(\theta_t), \cos(\theta_t) \rightarrow \Delta x_t, \Delta y_t, \Delta\theta_t$$

Lastly, I tried to compensate for the nonlinear terms of the motion model by including the sine and cosine of theta in the input. The plots are shown in figure 12 and 13 and the error measures in table 4.

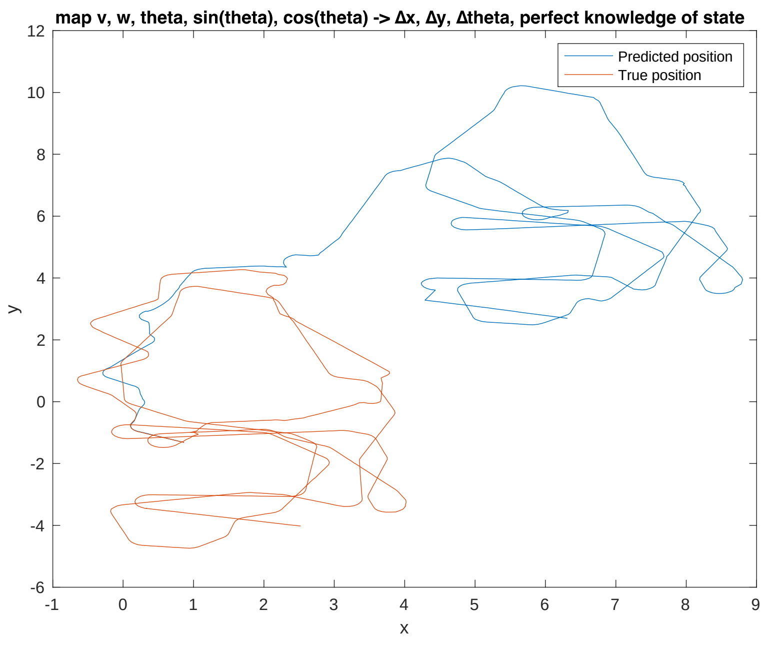

Figure 12: Local model with nonlinear terms and perfect state knowledge.

Table 4: Local model with nonlinear terms and perfect state knowledge local and global error.

	GLOBAL ERROR MEAN	GLOBAL ERROR VARIANCE	LOCAL ERROR MEAN	LOCAL ERROR VARIANCE
LOCAL MODEL (PERFECT STATE), NONLINEAR TERMS	6.7696	8.8824	0.0248	0.0015

If we compare figure 12 and 13 to figure 8 and 10, we can see that the nonlinear term do not help to get better results. This confirms the choice of the locally weighted linear regression model that I chose. Otherwise the nonlinear term should lead to improvement.

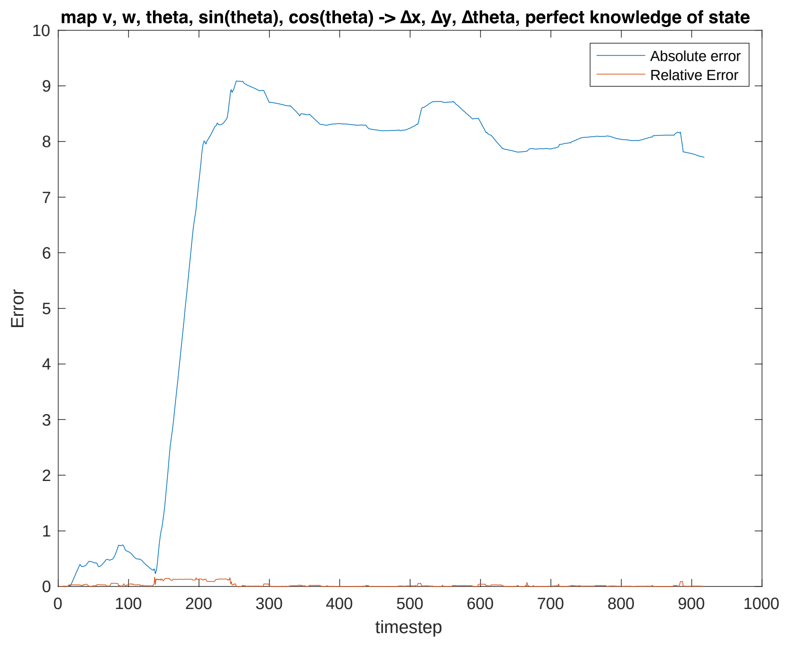

Figure 13: Local model with nonlinear terms and perfect state knowledge error plot.

Over all, the model learned from data was able to outperform the nonlinear model. This shows that learning can have advantages over solely modeling by also observing unmodeled dynamics or relations between input and output. Next to that, the locally linear approach was verified with the nonlinear terms not leading to any improvements. Nevertheless, this approach was not robust as well to local errors which lead to huge global drift over time, but could certainly improve the local motion model for the algorithm implemented in assignment 0.