SSY230 Learning dynamical systems using system identification

Project 1

Alfred Aronsson

April 23, 2024

1 Introduction

First project in SSY230 focused on estimating functions from noisy data.

Question 1

(a) In order to validate my linear regression function I compared it to case which is already known. If I have x values of 1 through 10 and likewise y values of 1 through 10. This is a simple straight line of the form: $y = x$. Therefore the linear regression should give me a linear term of 1. The function LinRegress in Matlab gave me the following result for the x and y values:

$$\hat{\theta} = 1 \tag{1}$$

This validates that the function LinRegress works correctly for a linear function of the form $y = x$.

I also want to validate that LinRegress works correctly for linear functions of the form $y = kx + m$. To do this I add 1 to my y values from before such that $y = \begin{bmatrix} 2 & \dots & 11 \end{bmatrix}^T$. To my regressor x, I will add a columns of 1s such that $x = \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 10 \end{bmatrix}^T$. These x and y values correspond to the linear function $y = x + 1$. When I enter the x and y values into my LinRegress function I get the following result:

$$\hat{\theta} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2}$$

These θ values validate that the function works correctly for linear functions of the form $y = kx + m$ aswell.

When it comes to validating the function when it comes to the variance in the regression we can begin with reviewing the variance analytically. We can assume that:

$$\dim x = 1$$
$$x_i = 1$$
$$\theta = 0$$
$$e_i \in N(0, 1)$$

With these assumptions the estimate of θ_0 becomes:

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3}$$

Further we know that:

$$y_i = \theta x_i + e_i \tag{4}$$

If we combine (3) and (4) and insert our values for θ and x we get:

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{5}$$

Now we want to analyze the variance of our estimator:

$$\text{Var}\left(\hat{\theta}_N\right) = E[\hat{\theta}_N\hat{\theta}_N^T] = E\left[\left(\frac{1}{N}\sum_{i=1}^{N} e_i\right)\left(\frac{1}{N}\sum_{j=1}^{N} e_j\right)^T\right] \tag{6}$$

We can then combine the sums:

$$\text{Var}\left(\hat{\theta}_N\right) = E\left[\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} e_j e_i\right] \tag{7}$$

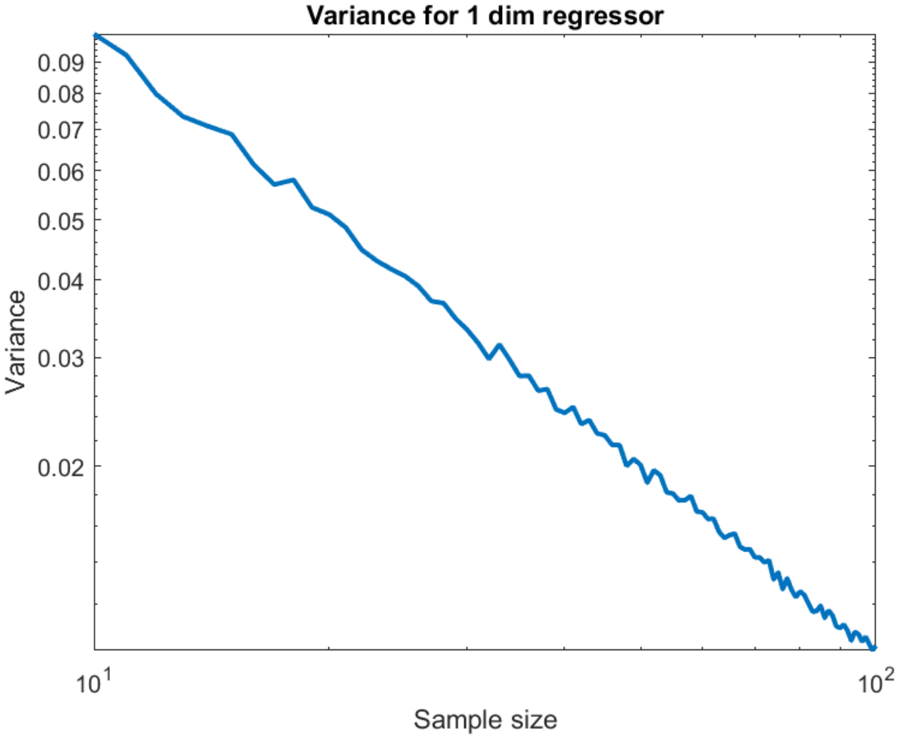
From (7) it follows that:

$$\text{Var}\left(\hat{\theta}_N\right) = E\left[\frac{1}{N^2}\sum_{i=1}^{N} e_j^2 + \cancel{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{i\neq j}^{N} e_j e_i}\right] \tag{8}$$

The second part cancels because e_i and e_j are independent of one another. The part that's left is the expectation of the white noise, which is simply the variance σ^2. All in all we get the result:

$$\text{Var}\left(\hat{\theta}_N\right) = \frac{1}{N^2}\sum_{i=1}^{N} E[e_i^2] = \frac{1}{N^2}\sum_{i=1}^{N} \sigma^2 = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N} \tag{9}$$

This can be validated in matlab using a Monte Carlo simulation. Here is the resulting plot of the simulation:



As can clearly be seen the variance behaves as predicted by the analytical solution which validates the function in the one dimensional case.

In the two dimensional case we can turn to equation (4.12) in S.S:

$$\operatorname{Var}\left(\hat{\theta}_N\right) = \sigma^2 (x^T x)^{-1} \tag{10}$$

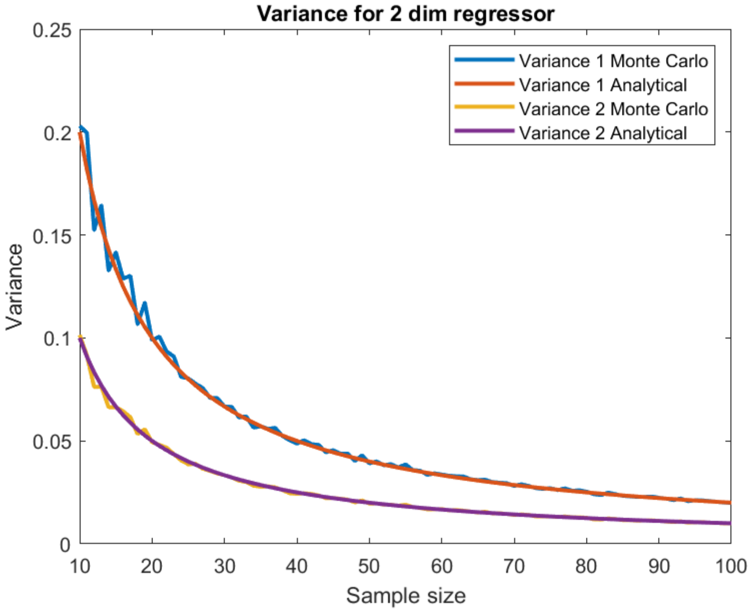
In the task definition θ is once again zero, and x is $x_i = \begin{bmatrix} 1 & 1 + (-1)^i \end{bmatrix}$. With this in mind (10) in matrix form will look like this:

$$\operatorname{Var}\left(\hat{\theta}_N\right) = \sigma^2 \left(\begin{bmatrix} 1 & 1 & 1 & 1 & \dots \\ 0 & 2 & 0 & 2 & \dots \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 0 \\ 1 & 2 \\ \vdots & \vdots \end{bmatrix} \right)^{-1} \tag{11}$$

This results in:

$$\operatorname{Var}\left(\hat{\theta}_N\right) = \sigma^2 \left(\begin{bmatrix} N & N \\ N & 2N \end{bmatrix}\right)^{-1} = \sigma^2 \begin{bmatrix} \dfrac{2}{N} & \dfrac{-1}{N} \\ \dfrac{-1}{N} & \dfrac{1}{N} \end{bmatrix} \tag{12}$$

Here we can clearly see that as in the case with one dimension the variances decrease with N. We can validate that the function works as intended for the two dimensional case by doing a similar Monte Carlo simulation as in the one dimensional case. Here is the resulting plot:



In the plot we can clearly see that the variance is reduced as with N as expected and that one variance is twice the size of the other. This validates that the function works correctly for two dimensional regressors aswell!

(b) In order to validate the function evalModel we can calculate the residuals of a known function and analyze if the results are reasonable. We can take the noise free data from task (a) with the values $x = \begin{bmatrix} 1 & \dots & 10 \end{bmatrix}^T$ and $y = \begin{bmatrix} 1 & \dots & 10 \end{bmatrix}^T$. Since the data is noise free we would expect that the residuals for a linear model are equal to 0. We can check this with our evalModel function in this way: $e = y - \hat{y}$. The result of this check is $e = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^T$, which validates that evalModel works correctly for linear functions.

(c) Given a regurilzed estimator that minimizes the cost function:

$$V(\theta) = \sum_{k=1}^{N} \left(y(k) - \theta \cdot x^T(k)\right)^2 + \lambda\theta^T \cdot \theta \tag{13}$$

We want to show that the regularized estimator (7) can be obtained through normal linear regression by appending the x and y vectors like this:

$$x_2 = \begin{bmatrix} x \\ \sqrt{\lambda} \cdot \mathbf{I}_x \end{bmatrix} y_2 = \begin{bmatrix} y \\ \mathbf{0}_y \end{bmatrix} \tag{14}$$

Proof. This can be shown by inserting x_2 and y_2 into a regular linear regression problem. We start with:

$$V(\theta) = \sum_{k=1}^{N} \left(y_2(k) - \theta \cdot x_2^T(k)\right)^2 = \sum_{k=1}^{N} \left(\begin{bmatrix} y(k) \\ \mathbf{0}_y \end{bmatrix} - \theta \cdot \begin{bmatrix} x(k) \\ \sqrt{\lambda} \cdot \mathbf{I}_x \end{bmatrix}^T \right)^2 \tag{15}$$

Which can be expressed in expanded matrix form as:

$$V(\theta) = \begin{bmatrix} y \\ \mathbf{0}_y \end{bmatrix}^T \begin{bmatrix} y \\ \mathbf{0}_y \end{bmatrix} - 2\theta \cdot \begin{bmatrix} x \\ \sqrt{\lambda} \cdot \mathbf{I}_x \end{bmatrix}^T \begin{bmatrix} y \\ \mathbf{0}_y \end{bmatrix} + \theta^T \cdot \begin{bmatrix} x \\ \sqrt{\lambda} \cdot \mathbf{I}_x \end{bmatrix}^T \begin{bmatrix} x \\ \sqrt{\lambda} \cdot \mathbf{I}_x \end{bmatrix} \cdot \theta \tag{16}$$

Doing the vector multiplications we get the following results:

$$V(\theta) = \underbrace{y^T y - 2\theta x y - \underbrace{\cancel{2\theta\sqrt{\lambda} \cdot \mathbf{I}_x \mathbf{0}_y}}_{=0} + \theta^T x^T x \theta}_{\text{Cost function for linear regression}} + \theta^T \underbrace{\sqrt{\lambda} \cdot \sqrt{\lambda}}_{=\lambda} \theta \tag{17}$$

Which can be expressed in a more familiar form as:
$V(\theta) = \sum_{k=1}^{N} \left(y(k) - \theta \cdot x^T(k)\right)^2 + \lambda\theta^T \cdot \theta$ □

In Matlab we can validate that the function works by choosing different λ values for a known regression and analyzing the results. We choose a standard regression of a straight line of the form $y = x$, with the familiar values, $x = \begin{bmatrix} 1 & \dots & 10 \end{bmatrix}^T$ and $y = \begin{bmatrix} 1 & \dots & 10 \end{bmatrix}^T$. Firstly if we make a regularized regression of the data with $\lambda = 0$ we would expect that the result is the same as for a regular regression. The result found for the case of $\lambda = 0$ is:

$$\hat{\theta}_{\lambda=0} = 1 \tag{18}$$

Now if we try to do a regularized linear regression of the data with a very high λ we expect to find that the θ is very close to zero. The result for the case of $\lambda = 10000$ is:

$$\hat{\theta}_{\lambda=10000} = 0.0371 \tag{19}$$

These results validate that the regularized regression works as intended.

(d) I want to verify my function polyfit by comparing it to data from a quadratic and cubic function I already know. The functions that known are:

$$f_{\text{Quadratic}}(x) = 1 + x + x^2 \tag{20}$$

$$f_{\text{Cubic}}(x) = 1 + x + x^2 + x^3 \tag{21}$$

From (20) and (21) I generated the following data:

$$\text{Quadratic: } x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \rightarrow y = \begin{bmatrix} 3 \\ 7 \\ 13 \\ 21 \\ 31 \\ 43 \\ 57 \\ 73 \\ 91 \\ 111 \end{bmatrix} \tag{22}$$

$$\text{Cubic: } x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \rightarrow y = \begin{bmatrix} 4 \\ 15 \\ 40 \\ 85 \\ 156 \\ 259 \\ 400 \\ 585 \\ 820 \\ 1111 \end{bmatrix} \tag{23}$$

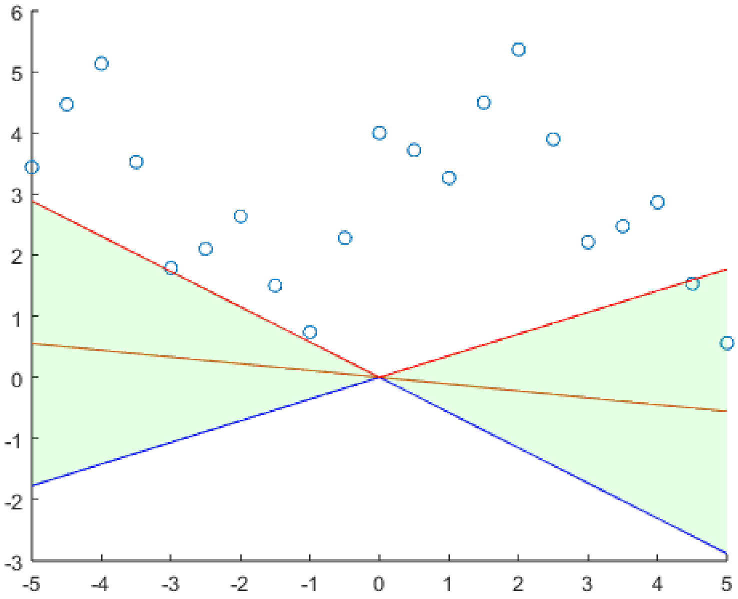
I entered (22) and (23) into the function polyfit and got the following results:

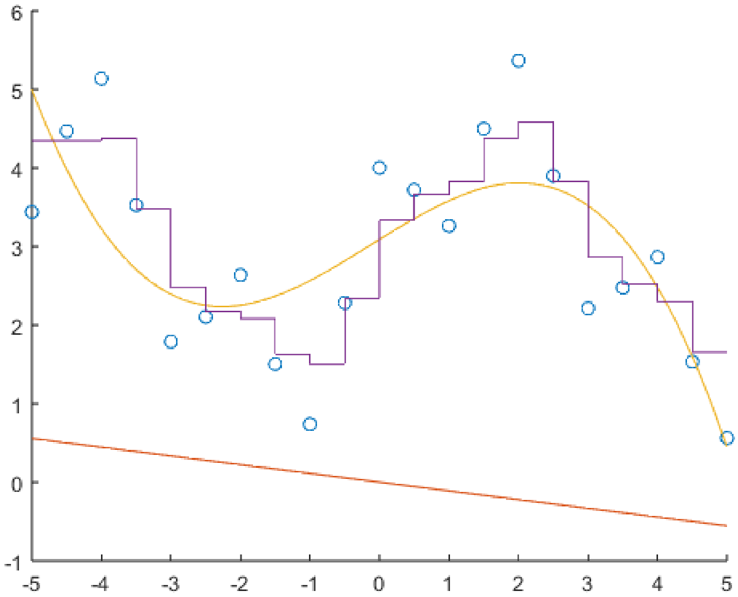
$$\hat{\theta}_{\text{Quadratic}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{24}$$

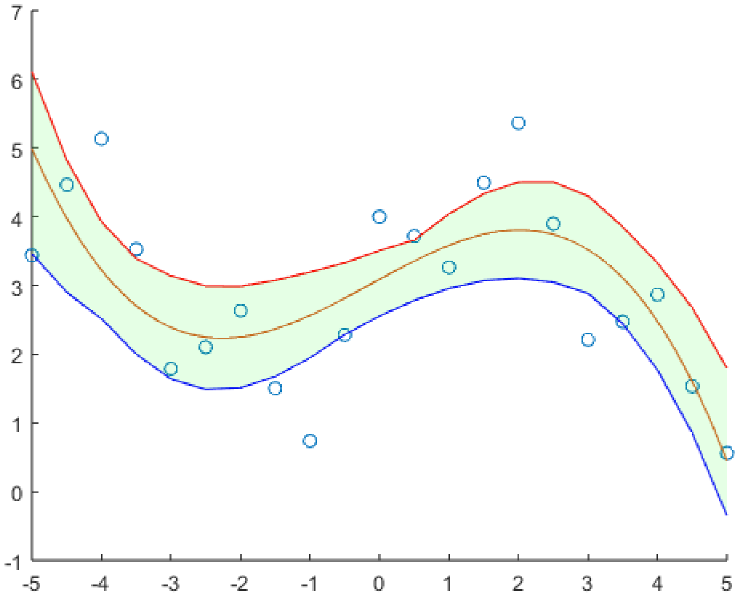
$$\hat{\theta}_{\text{Cubic}} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{25}$$

Which verifies that the function works correctly.

(e) In order to validate the plotModel function we are given a script, Test_plotModel that calls plotModel and produces 3 figures. These 3 figures can be compared to figures in the project definition. The figures produced are:







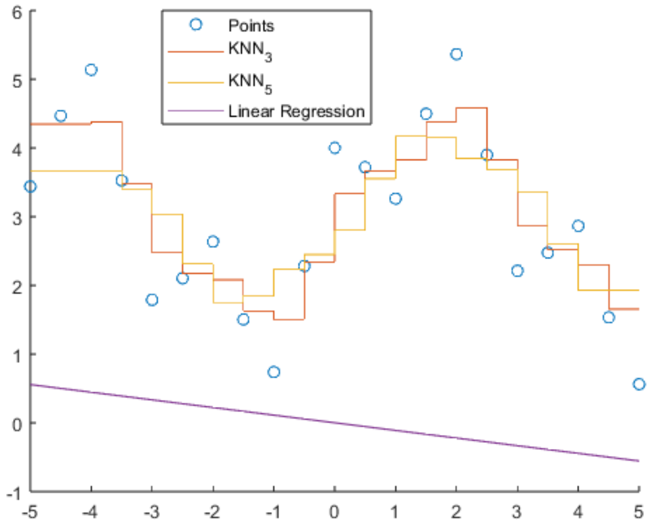
These figures does indeed look like the figures presented in the project definition which validates the function.

Question 2

(a) -

(b) -

(c) In order to test and verify the KNN functions I will reproduce the plots available on canvas. The plot produced by my functions knnRegress, evalModel and plotModel is the following:



The plot produced is the same as the plot produced in the canvas module, which verifies that my functions are correct.

Question 3.1

(a) As the task definition stated I generated training data and noise free validation data of different sizes using the given function linearData. I estimated linear regression models using the training data and evaluated the models using the validation data. Here are the results of the linear regression for sizes $N = \begin{bmatrix}10 & 100 & 1000 & 10000\end{bmatrix}$

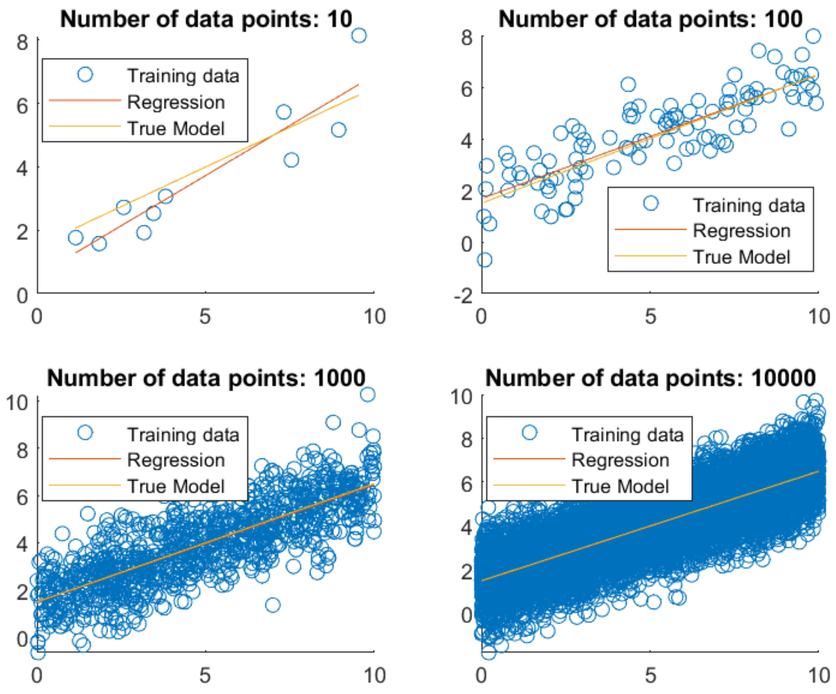
$$\hat{\theta}_{N=10} = \begin{bmatrix} 1.6320 \\ 0.3612 \end{bmatrix} \tag{26}$$

$$\hat{\theta}_{N=100} = \begin{bmatrix} 1.8650 \\ 0.4144 \end{bmatrix} \tag{27}$$

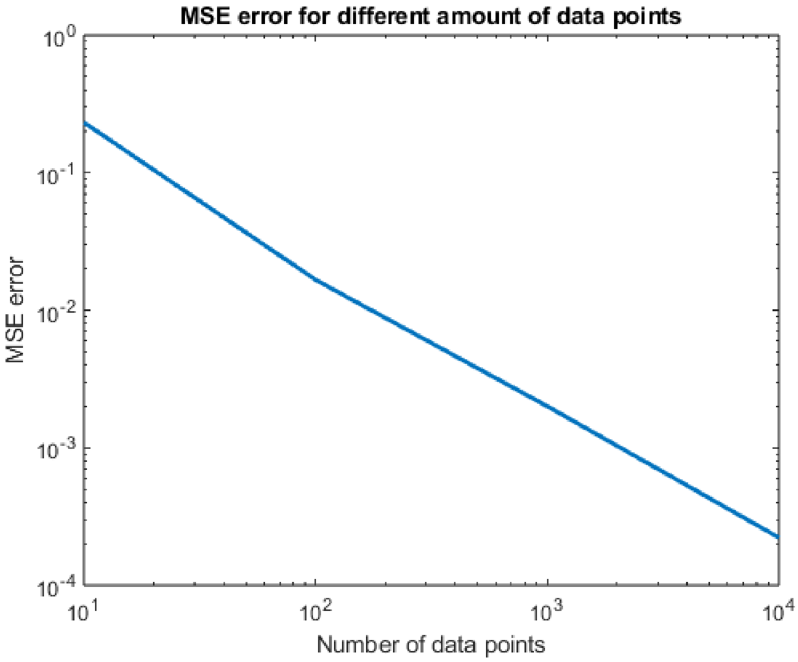
$$\hat{\theta}_{N=1000} = \begin{bmatrix} 1.5085 \\ 0.5015 \end{bmatrix} \tag{28}$$

$$\hat{\theta}_{N=10000} = \begin{bmatrix} 1.5049 \\ 0.4986 \end{bmatrix} \tag{29}$$

The linear regression models can be plotted together with the training data to give a visual insight to the models:



From the linear regression models and the validation data we can also analyze the model quality numerically by calculating the MSE. I calculated the MSE 100 times for each model and took the average. Here is a plot of the average MSE:



From (26) through (29) we can see that the models approach the real system $\theta_0 = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}$. This is supported visually through the plots where it can be observed that the regression fits the validation better for higher amount of data points. It is also observed numerically with the MSE rapidly droping to 0 for largre amount of datapoints.

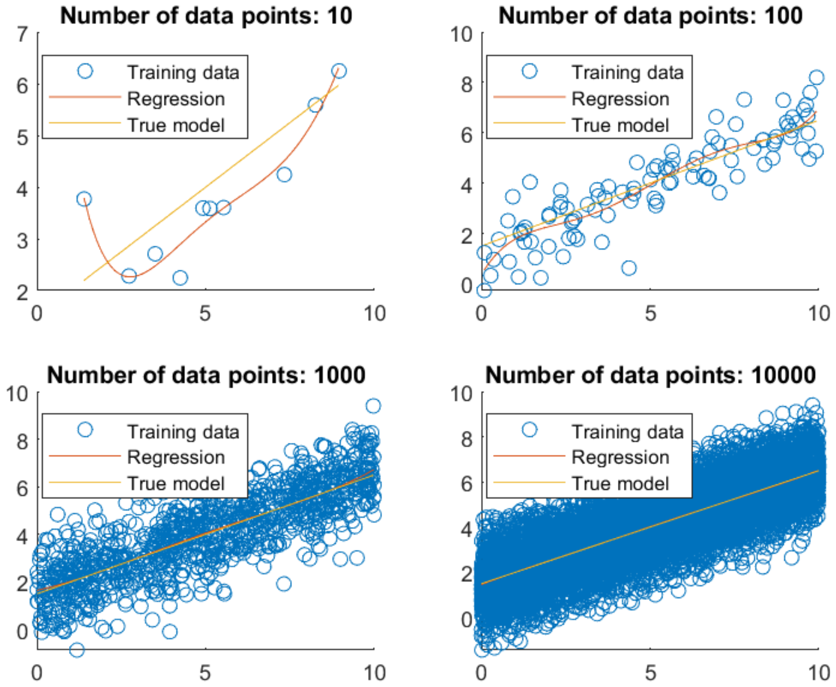
(b) If we repeat the experiment with a 5th degree polynomial and my polynomial regression function we get the following results, plots and MSE:

$$\hat{\theta}_{N=10} = \begin{bmatrix} 34.9925 \\ -29.9622 \\ 10.0392 \\ -1.5718 \\ 0.1194 \\ -0.0036 \end{bmatrix} \tag{30}$$

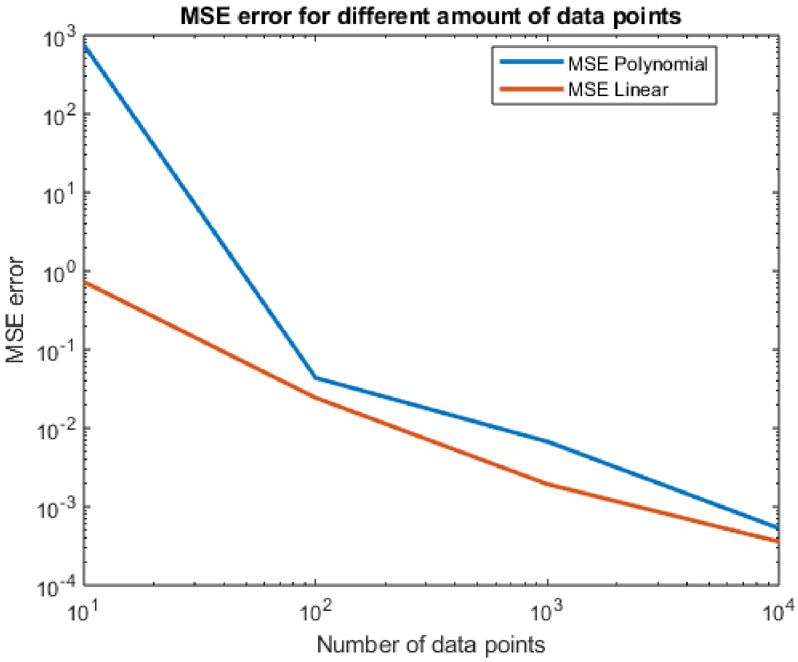
$$\hat{\theta}_{N=100} = \begin{bmatrix} 0.2681 \\ 2.3541 \\ -0.9858 \\ 0.2172 \\ -0.0207 \\ 0.0007 \end{bmatrix} \tag{31}$$

$$\hat{\theta}_{N=1000} = \begin{bmatrix} 1.5784 \\ 0.5373 \\ -0.1390 \\ 0.0575 \\ -0.0082 \\ 0.0004 \end{bmatrix} \tag{32}$$

$$\hat{\theta}_{N=10000} = \begin{bmatrix} 1.4948 \\ 0.5525 \\ -0.0310 \\ 0.0082 \\ -0.0009 \\ 0.0000 \end{bmatrix} \tag{33}$$



From (30) through (33), the plots and the MSE we can see that even though the regression is a 5th degree polynomial it still converges to the true system $\theta_0 = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}$ with the higher degree terms going to zero. We can compare this polynomial regression model with the linear regression model from task (a) by plotting both models MSE together:



We can clearly observe the the linear regression model has a higher quality than the polynomial regression for any number of data points.

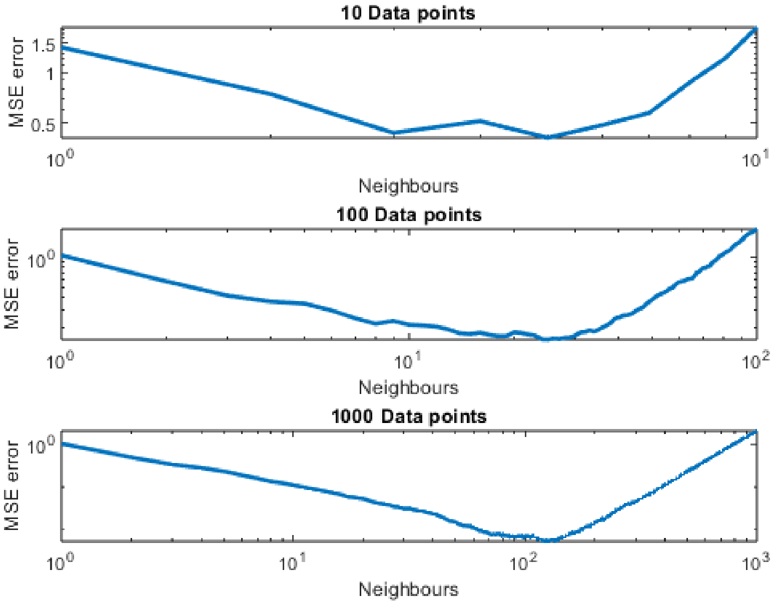
(c) In order to validate the variance using a monte carlo simulation I proposed a linear function with the true parameters $\theta_0 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$. Using these true parameter and a regressor with 100 samples I generated y values with some added noise. From the y values and my regressor I estimated a linear model and stored the estimated parameter values and the variance values. I repeated this 100 times. Then I calculated the variance of the parameter estimations and compared them to the mean of the model variances. This is the result:

$$\text{Emperical variance of intercept term} = 0.038894 \tag{34}$$
$$\text{Model variance of intercept term} = 0.040541 \tag{35}$$
$$\text{Emperical variance of slope term} = 0.000012163 \tag{36}$$
$$\text{Model variance of slope term} = 0.000011982 \tag{37}$$

This validates that the function computes the variances correctly.

(d) Firstly, I tried KNN models on models on data samples of different sizes but with the same noise variance. I tried the for the sizes $N = \begin{bmatrix} 10 & 100 & 1000 \end{bmatrix}$

all with the noise variance 1. In order to find out which amount of neighbours was optimal I generated models of every possible neighbour for the given data size. I then evaluated the MSE using the validation data for every model and plotted the result:



The optimal amount of neighbours was the following:

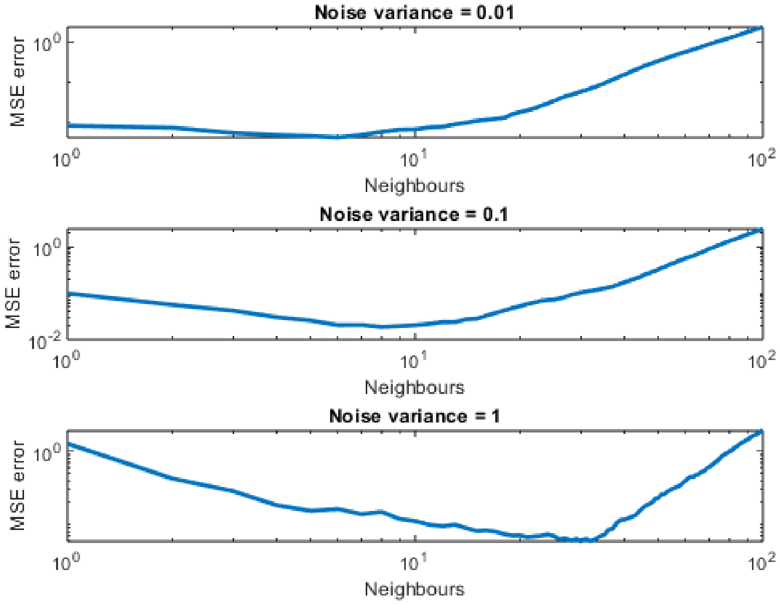
$$k_{N=10} = 5 \tag{38}$$

$$k_{N=100} = 25 \tag{39}$$

$$k_{N=1000} = 125 \tag{40}$$

From (38) through (40) we can see that the optimal amount of neighbours goes down (relative to model size) as the model size goes up.

We can conduct a similar experiment to analyze the optimal amount of neighbours for different noise variances. This time we choose a constant amount of data points, $N = 100$ and Var$= \begin{bmatrix} 0.01 & 0.1 & 1 \end{bmatrix}$. Here are the corresponding plots for this experiment:



The optimal amount of neighbours was the following:

$$k_{\text{Var}=0.01} = 6 \tag{41}$$
$$k_{\text{Var}=0.1} = 8 \tag{42}$$
$$k_{\text{Var}=1} = 30 \tag{43}$$

From (41) through (43) we can see that the optimal amount of neighbours goes up as the variance goes up.

Question 3.2

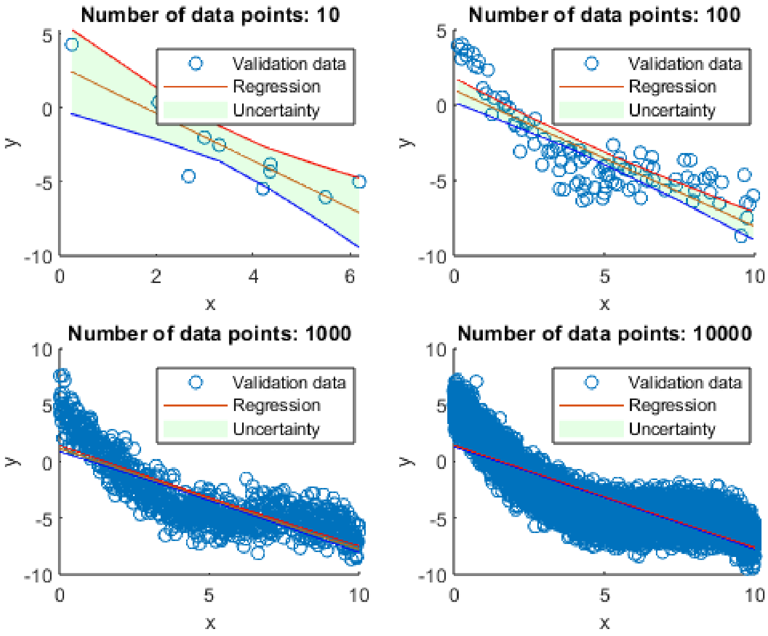
(a) If we repeat the experiment we did in task 3.1 (a) with the polyData and include the parameter uncertainty we get the following results, plots and MSE:

$$\hat{\theta}_{N=10} = \begin{bmatrix} 2.8433 \\ -1.6036 \end{bmatrix} \tag{44}$$

$$\hat{\theta}_{N=100} = \begin{bmatrix} 1.0067 \\ -0.9075 \end{bmatrix} \tag{45}$$

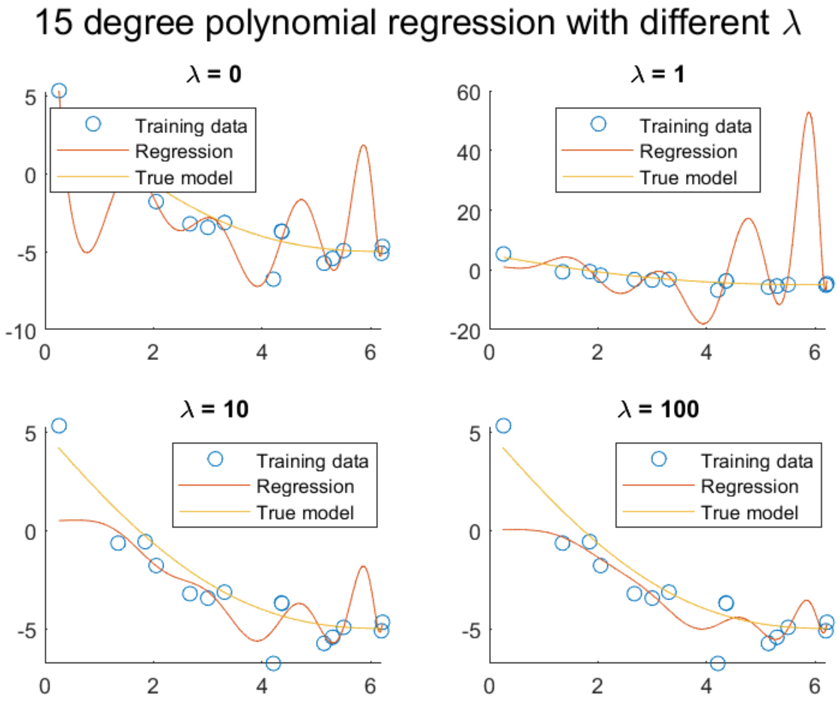
$$\hat{\theta}_{N=1000} = \begin{bmatrix} 1.1676 \\ -0.8934 \end{bmatrix} \tag{46}$$

$$\hat{\theta}_{N=10000} = \begin{bmatrix} 1.3990 \\ -0.9090 \end{bmatrix} \tag{47}$$



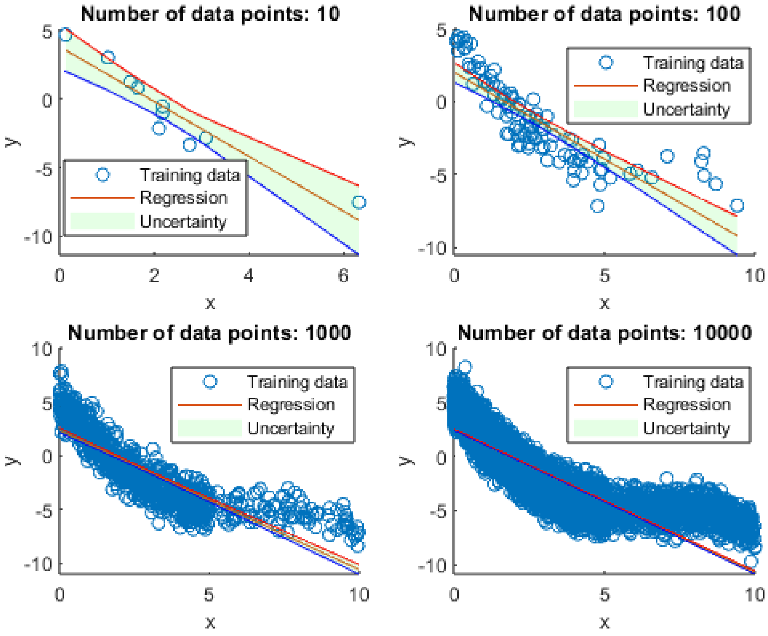
As can be clearly seen in the plots, the parameter uncertainty goes to 0 for large data sets. It is also be obvious that the linear regression does not converge to the correct function, as is evidenced both by the plots. That the parameter uncertainty is 0 does not means that regression model is good, it is simply a measure that indicates that the regression model is as good as it is going to get.

(b) In order to investigate if we can make a good model of a polynomial regression of high degree with regularization, we can generate a data set with 15 samples and fit a 15 degree polynomial to that data set. I generate 4 such models with different amount of regularization. The regularization levels i choose for the models where: $\lambda = \begin{bmatrix} 0 & 1 & 10 & 100 \end{bmatrix}$. Here are the 4 generated models depicted in plots:



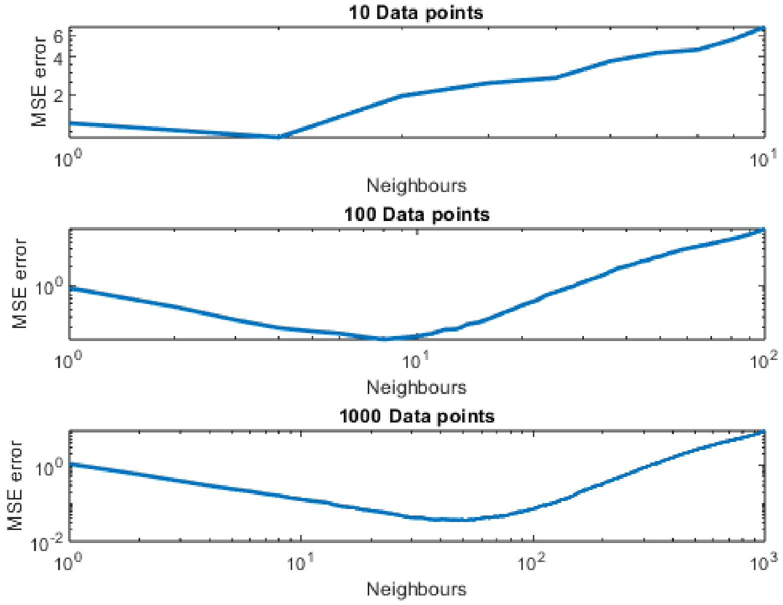
As can be expected the model with out regularization grossly over fits the data. With increased regularization the model quality improves.

(c) If we repeat the experiment done in task 3.2 (a) with this unsymmetrical data set we get the following result:



This is indeed different from task 3.1 (a). It is different because the unsymmetrical data set shifts the ”center of mass” of the data set causing the regression to disproportionally fit the data set. This is of course problematic because then the regression can not estimate the true nature of the data set. To remedy this you can sample the skewed part of the data set to generate a more symmetrical set, or you can use regression techniques other than linear regression that are not as sensitive to skewed data.

(d) We can repeat the experiment we did in task 3.1 (d) here but with our polynomial data set instead of the linear one. Here is the result for data sets of size $N = \begin{bmatrix} 10 & 100 & 1000 \end{bmatrix}$ with noise variance 1:



The optimal amount of neighbours was found to be:

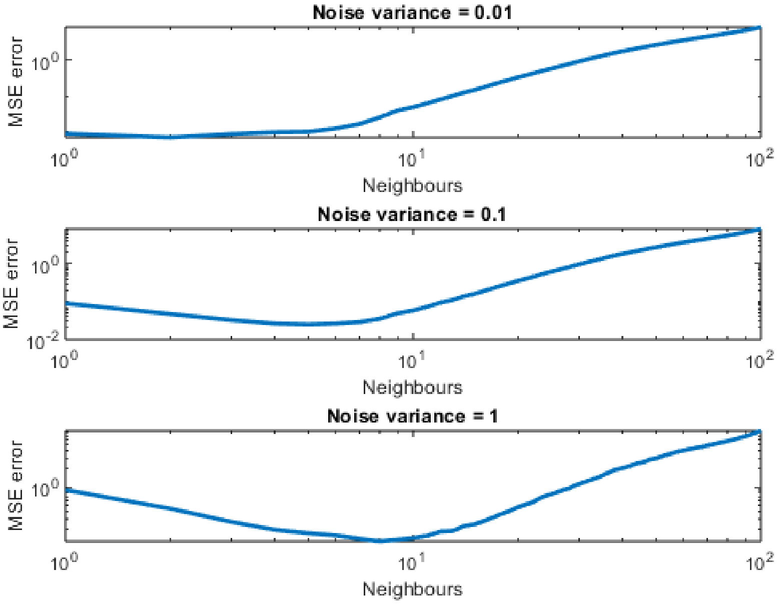
$$k_{N=10} = 2 \tag{48}$$

$$k_{N=100} = 8 \tag{49}$$

$$k_{N=1000} = 52 \tag{50}$$

We can identify the same trend here as with the linear data set, the relative amount of optimal neighbours goes down as the data sets get larger. It can be noted that this trend is even more prevalent here than in the case of a linear data set. This is because the polynomial data set is much more intricate.

Now we conduct the experiment with a data set of size $N = 100$ and noise variances Var=$\begin{bmatrix}0.01 & 0.1 & 1\end{bmatrix}$. Here are the results:



The optimal amount of neighbours was found to be:

$$k_{\text{Var}=0.01} = 2 \tag{51}$$

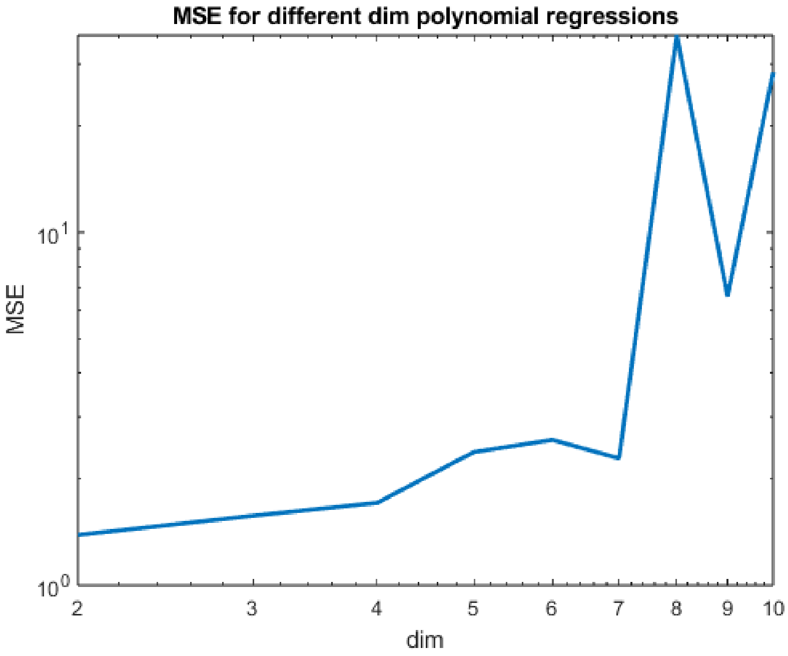
$$k_{\text{Var}=0.1} = 5 \tag{52}$$

$$k_{\text{Var}=1} = 8 \tag{53}$$

Here we can also spot the same trend as was found in the linear data set. As the variance goes up, the optimal amount of neighbours goes up.

Question 3.3

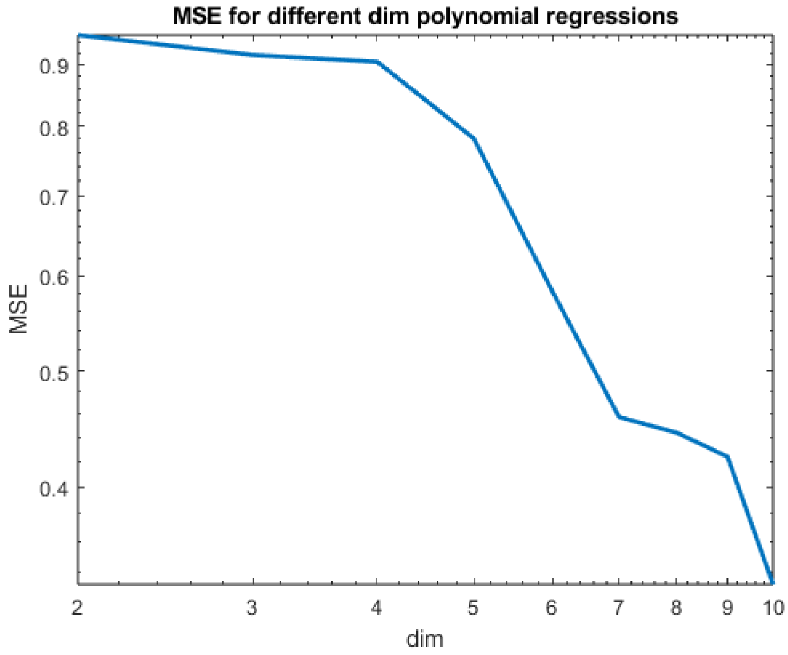
(a) In order to find the best polynomial regression model for a data set with sample size 50 and noise variance 0.4, I generated polynomial models for order 2 to 10. I then calculated the MSE for these models 100 times and calculated the mean MSE. This is the result:



$$\text{Best dim} = 2 \tag{54}$$

$$\text{Best MSE} = 1.3877 \tag{55}$$

If we extend the sample size to 1000 we get the following results:



$$\text{Best dim} = 10 \tag{56}$$

$$\text{Best MSE} = 0.3319 \tag{57}$$

The reasons that we can find a better model with more data than with less data are among many:

- **More information**: With more information it possible to more closely capture the form of the data set.
- **Reduced variance**: As proved in task 1 (a) larger data sets have less variance which enhances model quality.
- **Model Complexity:** With larger data sets you can use higher degree models with lower risk of over-fitting.

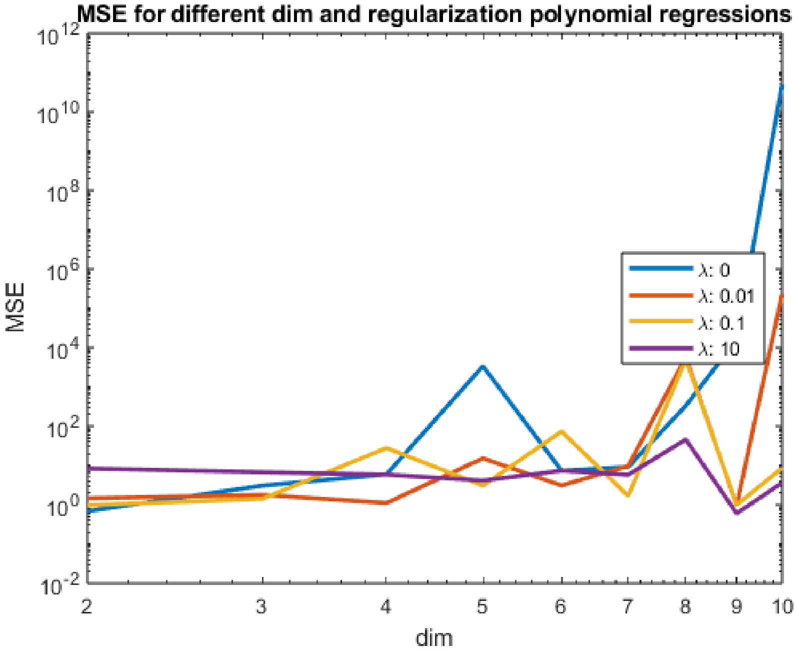
We can investigate if we can get a better model with reduced variance by running the experiment again but now with noise variance 0.2 with 1000 samples. The result is now:

$$\text{Best dim} = 10 \tag{58}$$

$$\text{Best MSE} = 0.2788 \tag{59}$$

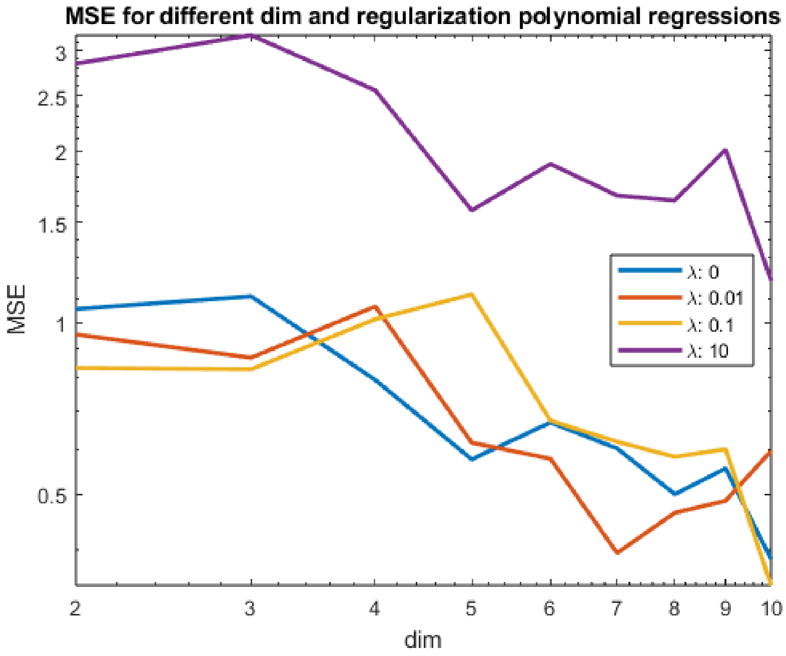
With less variance the model quality gets better!

(b) In order to find out if its better to have a high degree polynomial model with some regression or a low degree polynomial with no or little regularization I will first repeat the experiment conducted in task 3.2 (b) with a data set of size 10. Here are the results:



Here we can see that the model with the best quality is a second degree polynomial without regularization.

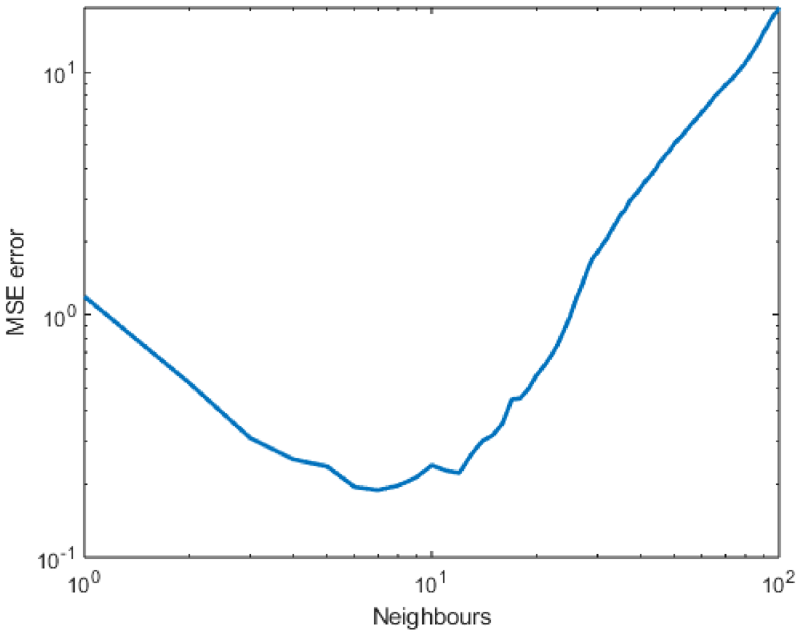
Now lets repeat the experiment with a data set of size 100. Here are the results:



Here we can see that the best model was a polynomial of size 10 with a regularization of $\lambda = 0.1$.

According to the two experiments, lower degree polynomials with out regularization is better for small data sets and higher degree polynomials with regularization are better for larger data sets.

(c) We can determine if KNN models are better than polynomial models by comparing the MSE's of the best polynomial model and the best KNN model for a data set of size 100. From task (b) we know that the best polynomial is a 10th degree polynomial with 0.1 regularization. In order to find the best KNN model we repeat the experiment done in task 3.1 (d) and 3.2 (d):



The optimal amount of neighbours was found to be:

$$k_{N=100} = 7 \tag{60}$$

Now we can compare the MSE of the polynomial model and the KNN model:

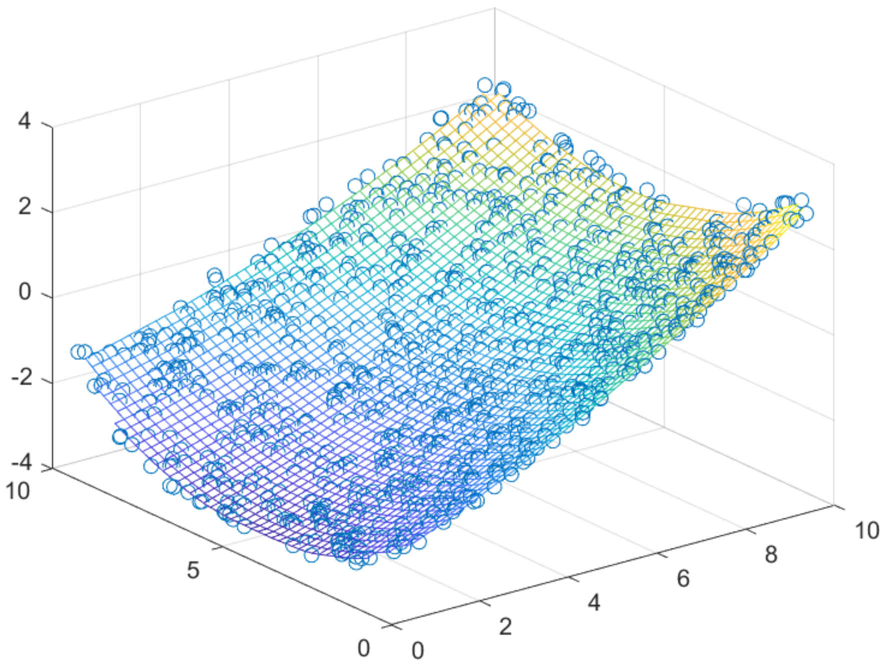
$$\text{MSE}_{\text{Poly}} = 0.4830 \tag{61}$$

$$\text{MSE}_{\text{KNN}} = 0.3543 \tag{62}$$

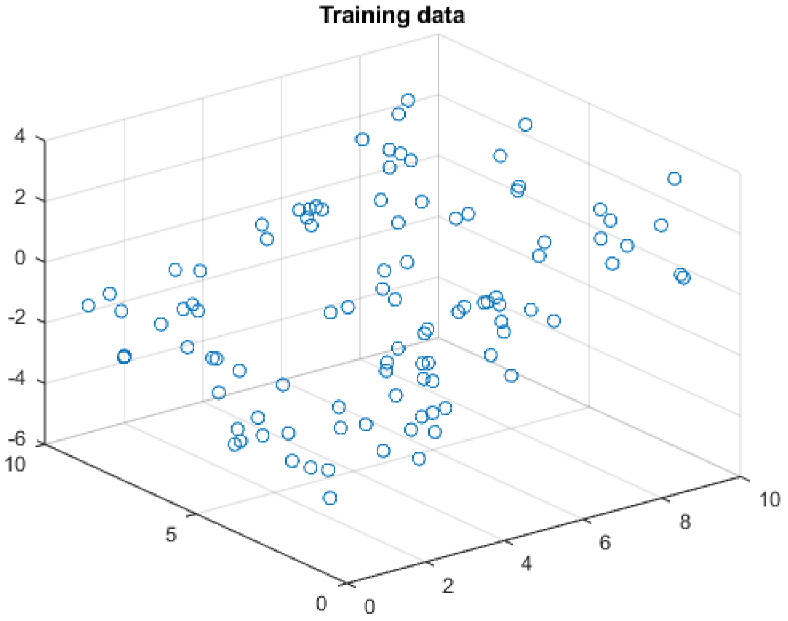
According to this experiment KNN models are better on this data set.

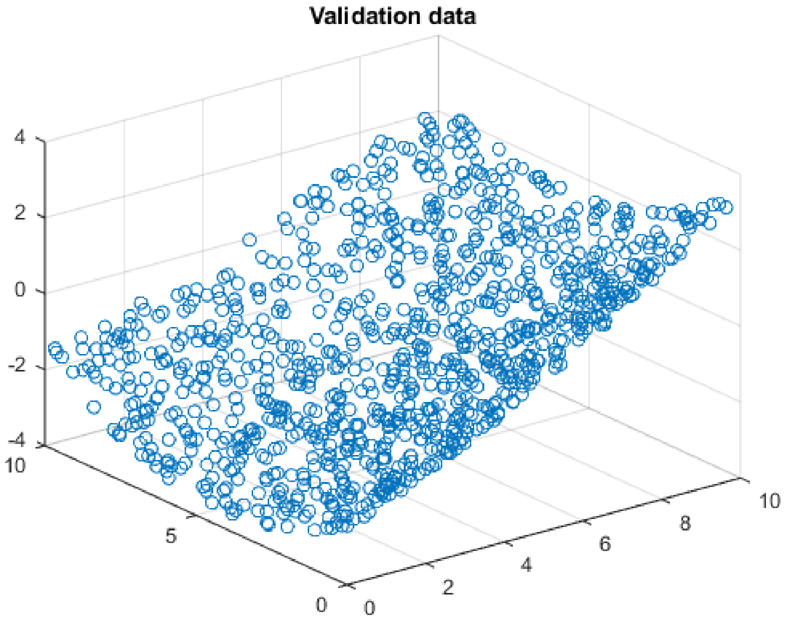
Question 4.1

(a) Here is the desired plot:



(b) Here are the traing data and validation data plotted:





(c) The linear regression model had the following model quality when testing it on the validation set:

$$\text{MSE} = 1.8789 \tag{63}$$

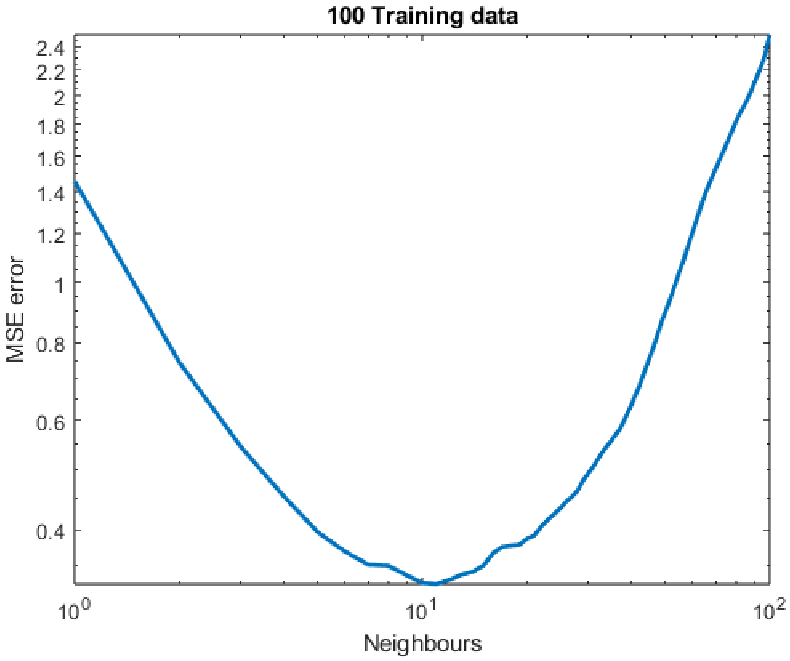
Due to the stochastic nature of the training data the MSE changes slightly each time you run the script but with a sample size of 100 this change is rather small.

(d) The polynomial models had the following model quality when testing it on the validation set:

$$\text{MSE for 2nd degree model} = 0.0541 \tag{64}$$
$$\text{MSE for 3rd degree model} = 0.0977 \tag{65}$$
$$\text{MSE for 4th degree model} = 0.1424 \tag{66}$$
$$\text{MSE for 5th degree model} = 0.6433 \tag{67}$$

The same logic follows here that due to the stochastic nature of the training data the MSE values changes slightly every time the script is run. However the order of the model quality is always the same for the models. The model with degree 2 is always best etc.

(e) In order to find the optimal amount of neighbours I repeated the experiment conducted in task 3.1 (d) 3.2 (d) etc. Here is the result for a sample size of 100:



The optimal k was found to be:

$$k_{100} = 11 \tag{68}$$

$$\text{KNN MSE} = 0.3272 \tag{69}$$

We can compare (69) to (64) and observe that the polynomial model is much better. After some testing I found that this changes when the sample size is 14. For sample sizes below 15 KNN is better.