**2.0 Function Estimation and Time- series Prediction**

For this exercise, the SVM toolbox is used.

**2.1 The Support Vector Machine for Regression**

A artificial data has been created with the help of 20 different data points using the User interface ‘uiregress’. We start by adding 20 points to the user interface. For the chosen data points the added points output the polynomial function of degree 2. And by creating a different dataset using different interface, with different types of kernels.

In the following figures, we display the results for linear, polynomial (degree 2), RBF and linear B-Spline (degree 2). Although theoretically, a polynomial kernel of degree 2 would be the best, we see that other kernels (RBF and linear B-Spline) perform similarly. However, the B-Spline kernel overfits the data.

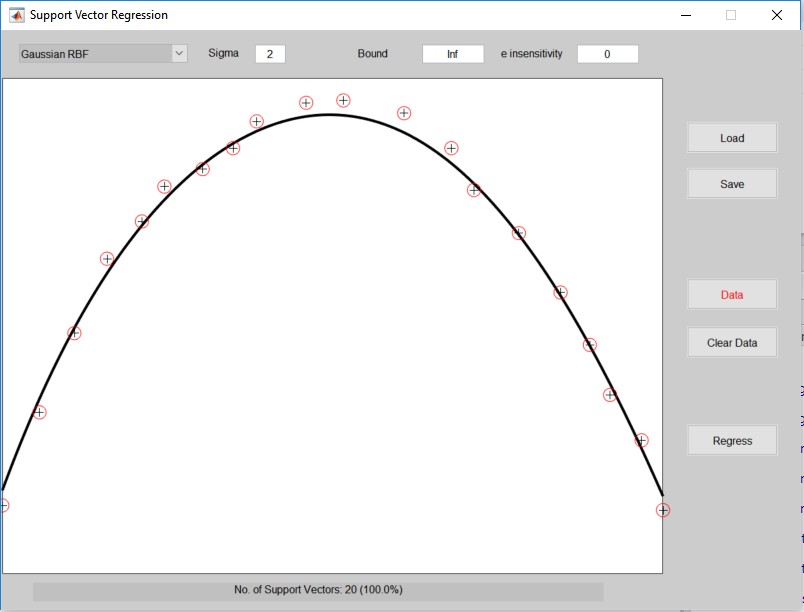
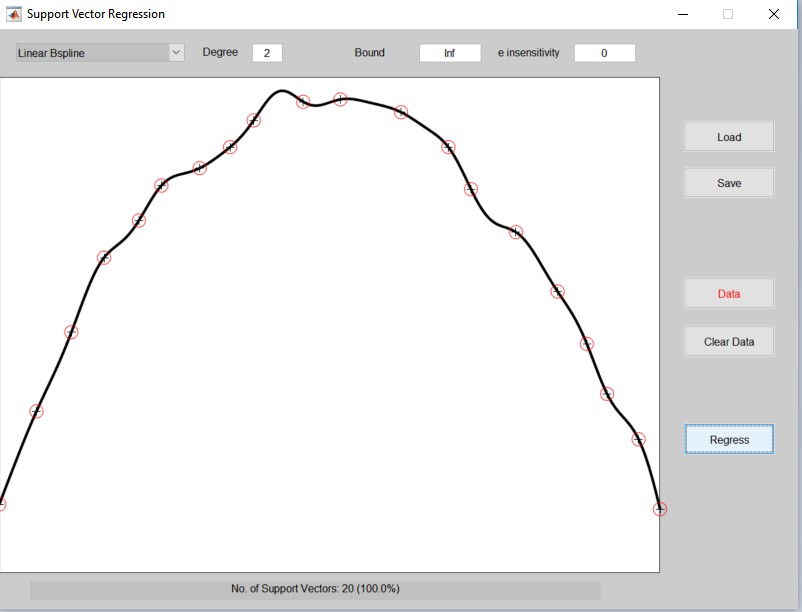
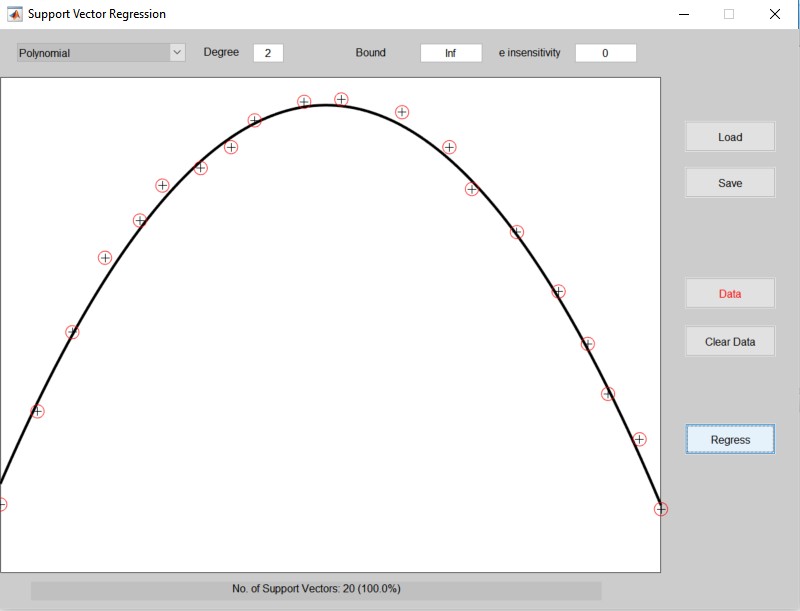
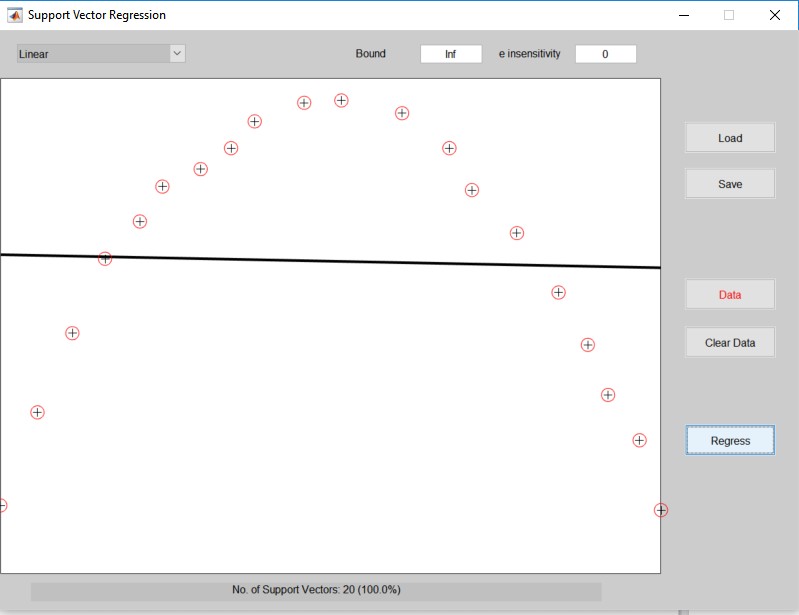


Fig-1- Regression results for linear, polynomial, RBF and linear B-Spline kernels

Figure 2 depicts a comparison of a linear and RBF kernel on approximately linear data. but for linear data with noise, a linear kernel will deliver a better performance on unseen samples, as RBF or more complicated kernels will clearly overfit the data.

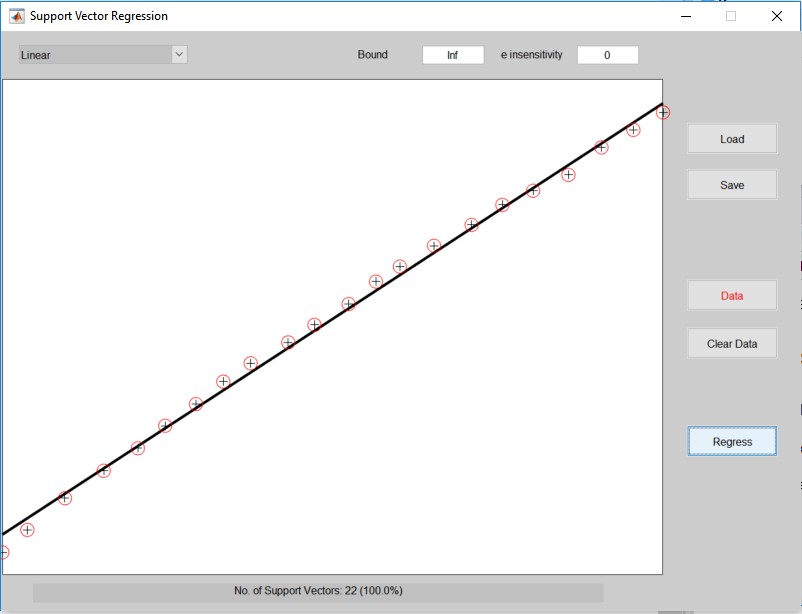
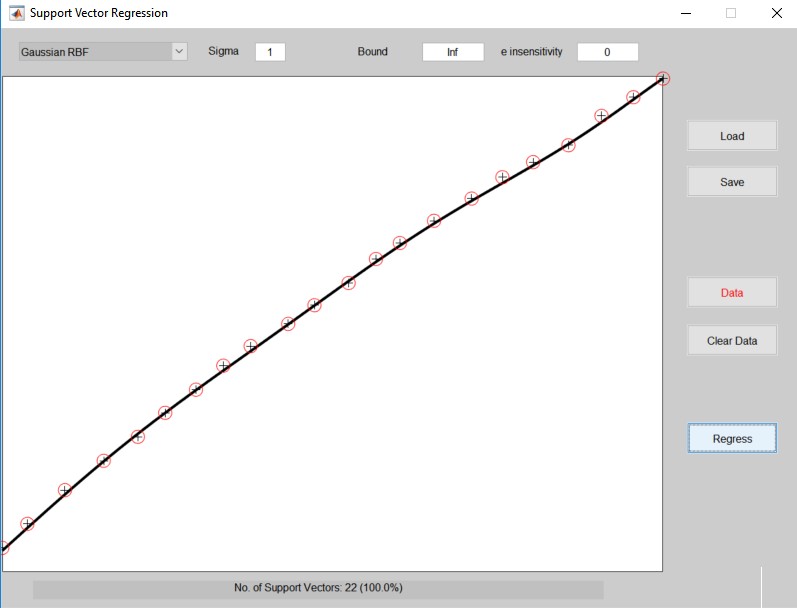


Figure 2: Comparison of linear and RBF kernels on approximately linear data

Both parameters (Bound and e) have an influence on the **flatness** of the regression curve. Figures 3 and 4 show a linear kernel on linear data for increasing values of e and Bound.

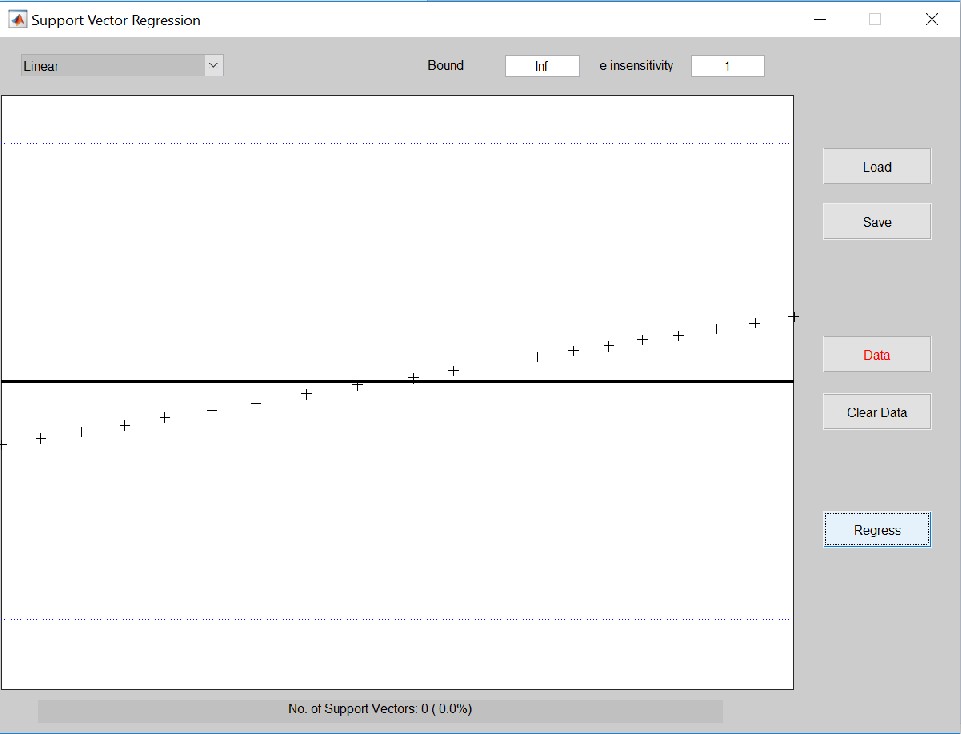
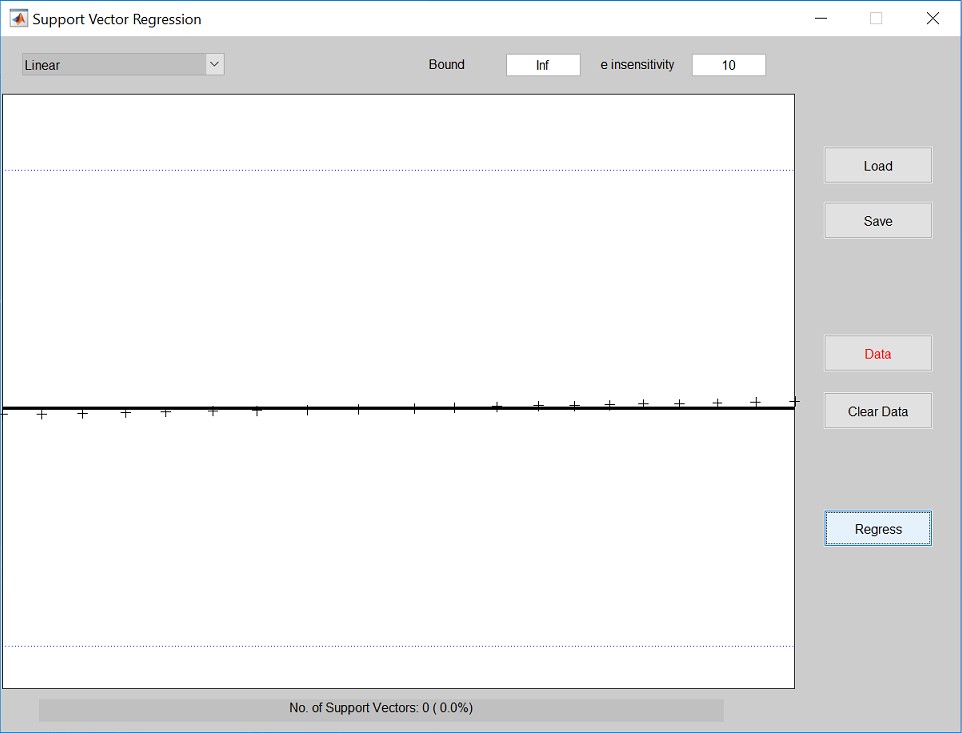
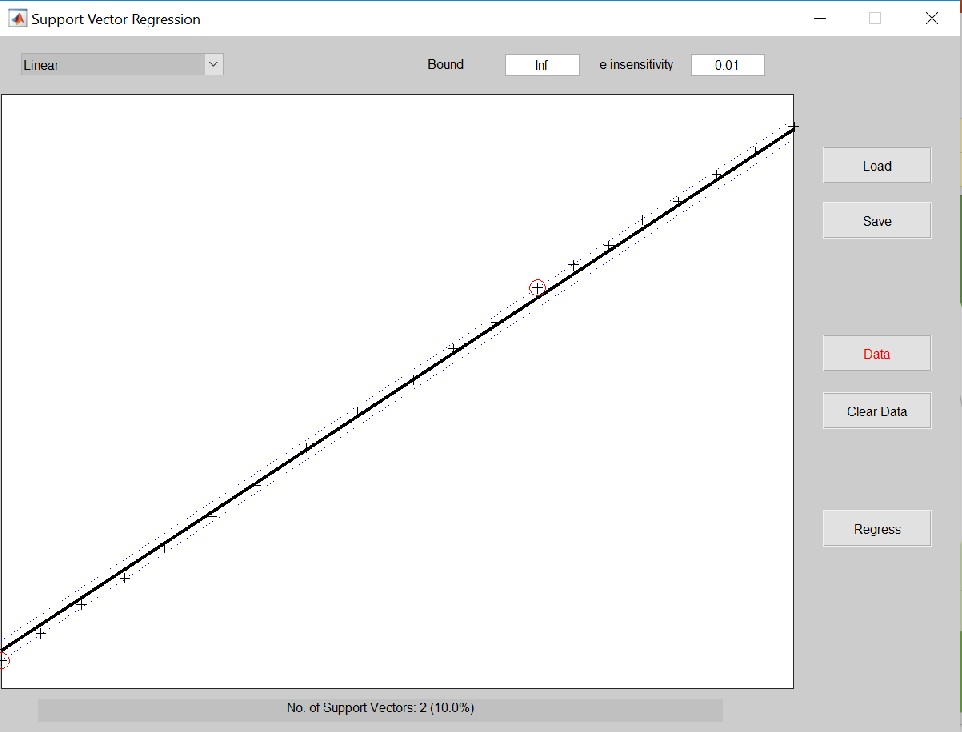
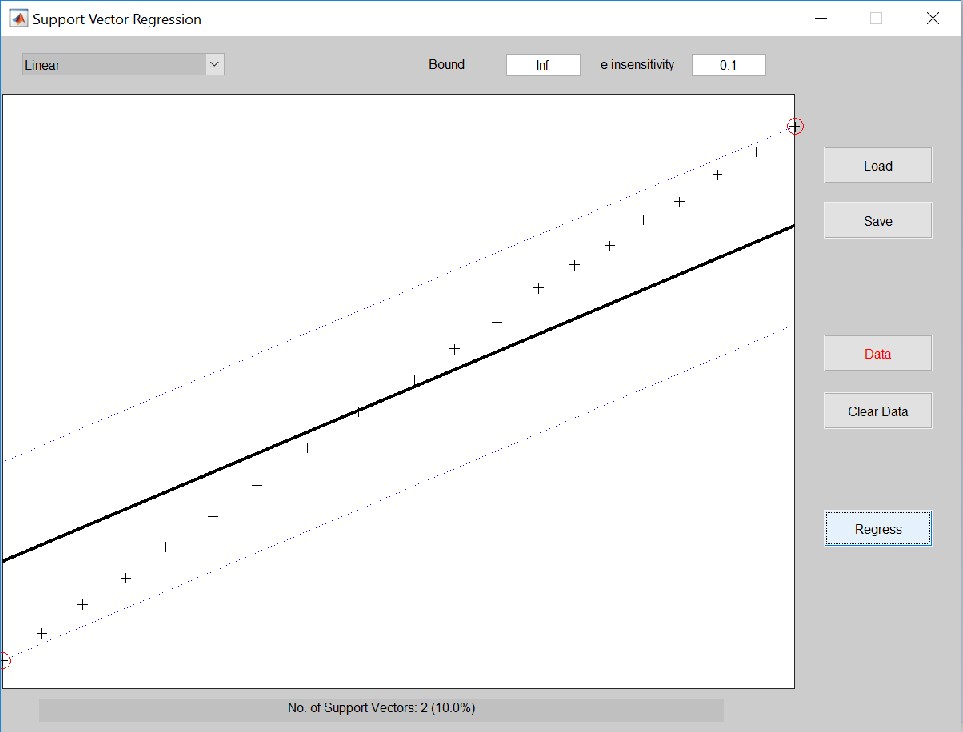


Figure 3: Linear kernel for increasing values of e, e.g. 0.01, 0.1, 1, 10 (Bound=∞)

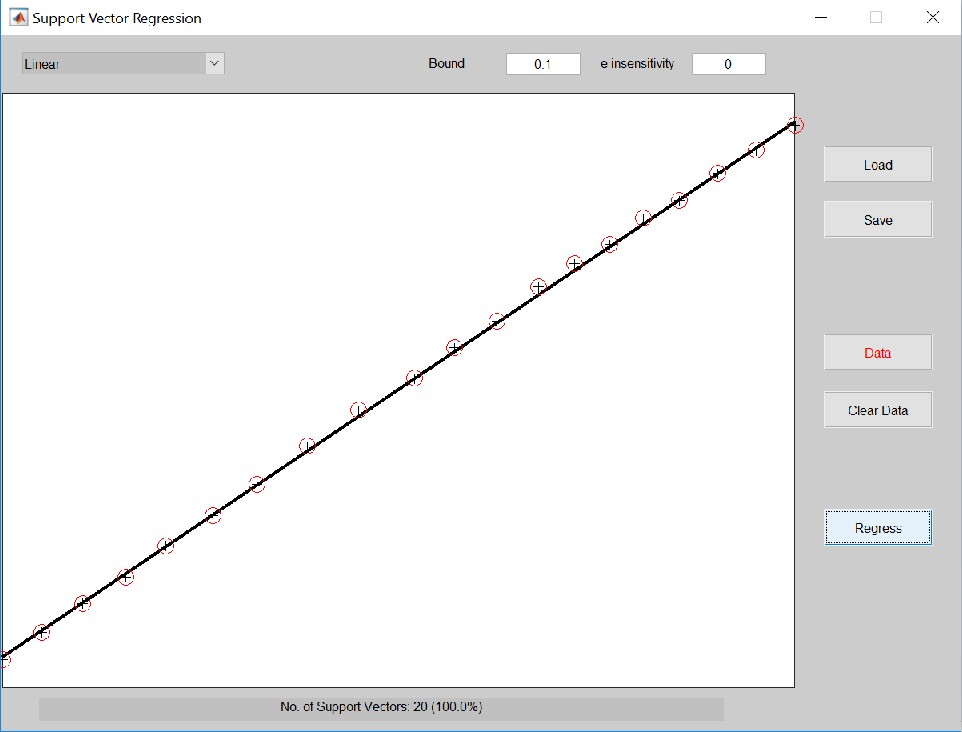
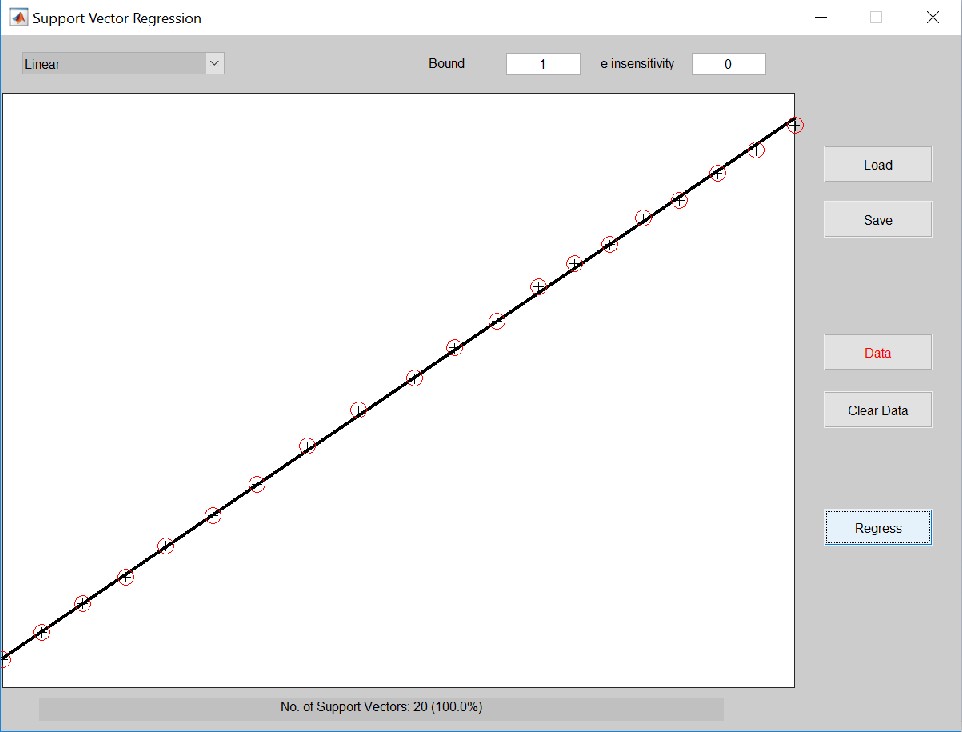
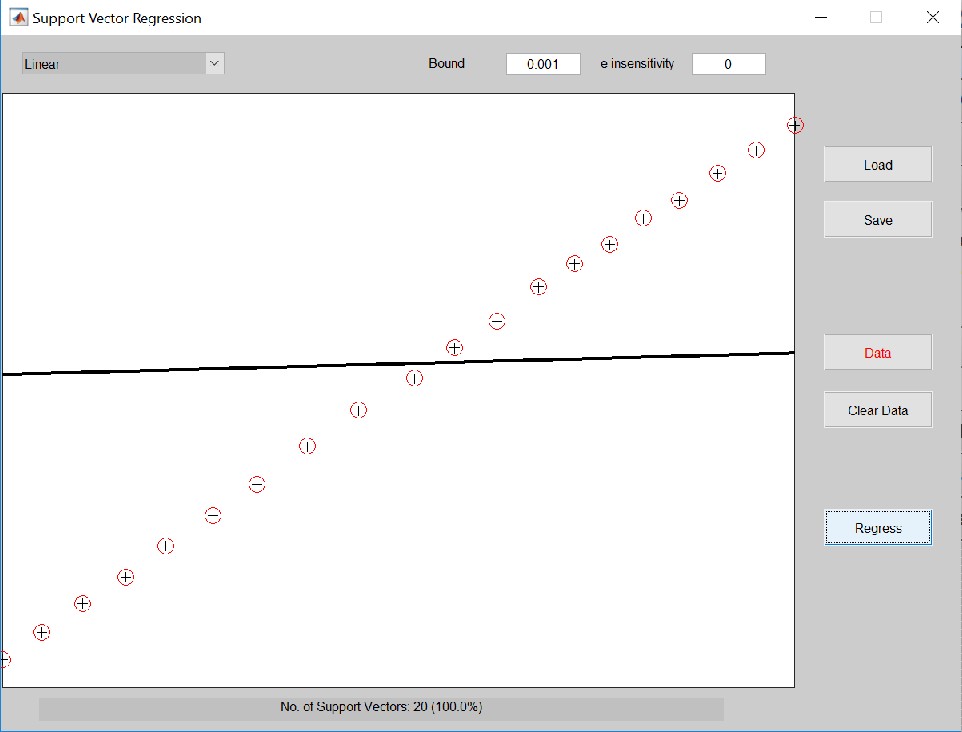
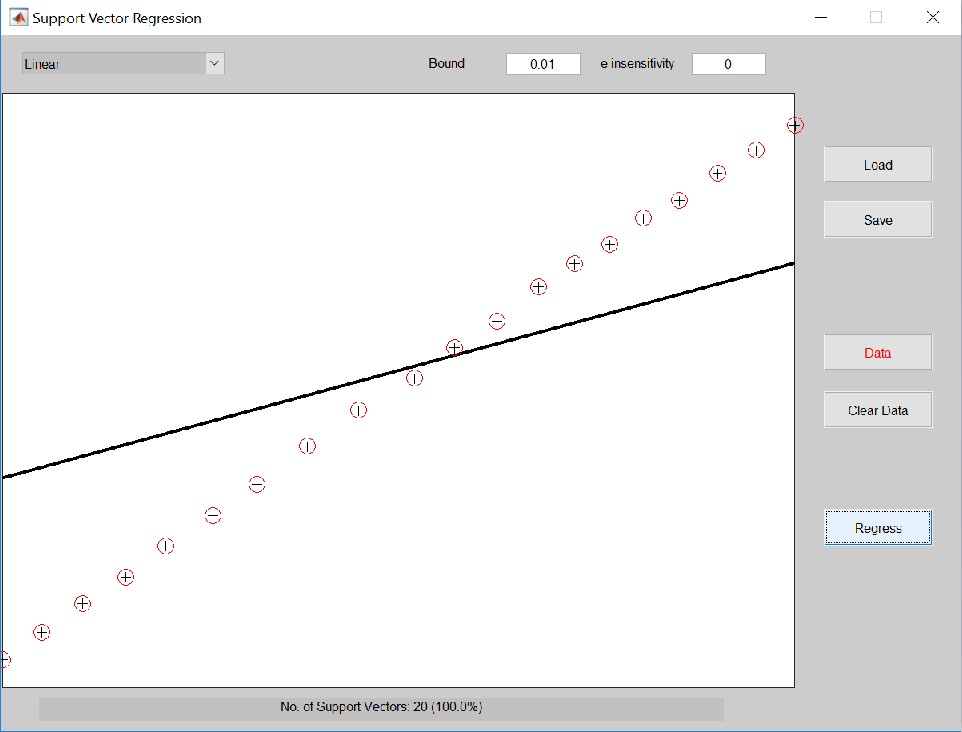


Figure 4: Linear kernel for increasing values of Bound, e.g. 0.001, 0.01, 0.1, 1 (e=0)

Parameter e controls the width of the e-insensitive zone (related to the loss function), used to fit the training data. The higher e, the higher the number of data points that have loss function equal to 0. Therefore, the value of e affects the number of support vectors used to construct the regression function (points for which loss function is not 0). The higher e, the fewer support vectors are selected. High values of e result in more flat estimates. Low values of e result in more accurate estimate on the training set. e must therefore be chosen to reflect the data in some way. Choosing a too low e might result in overfitting, while choosing a too high e might result in a too general (too flat) model.

Parameter C (Bound) determines the trade off between the model complexity and the degree to which deviations larger than e are tolerated in the optimization formulation. For example, if C is too large (infinity), then the objective is to minimize the empirical error only (error on the training set), without regard to model complexity.

Sparsity comes in for non null values of e ( in the equations below). In the formulation of the optimization problem of SVM regression, the Lagrange multipliers of the dual problem vanish (they have to equal 0 for a solution to exist, linked to the KKT conditions) for the points within the e-insensitive zone. Therefore, sparsity appears as some Lagrange multipliers equal 0 and not all points are needed to characterize the regression function. The points associated to the non vanishing coefficients are called Support Vectors.

The general formulation (Vapnik) of SVM regression is as follows:

For training data {xi, yi}N

i=1

, map a linear function in the feature space such that y**w**(xi) =

**w**T φ(xi)) + β0. From the structural risk minimization principle, the formulation can be

rewritten as a minimization problem:

subject to

1 T

minw,ξ 2 w

w + c

N

X

i=1

ξi (1)

yi − y**w**(xi) ≤ + ξi, i = 1, ..., N and ξi ≥ 0, i = 1, ..., N (2)

where ξi is a slack variable and the parameter defining the loss function. For least squares SVM regression, the minimization problem is written by:

1 T

minw,β0 ,e 2 µw

such that yi = y**w**(xi) + ei, i = 1, ..., N .

1

w + 2 γ

N

X

i=1

e2 (3)

i

We can rewrite e2 by (yi − (wT φ(xi) + b))2 so that the above minimization can be rewritten

i

by

1 T

minw,β0 ,e 2 µw

1

w + 2 γ

N

X

i=1

(yi − (wT

φ(xi) + b)2), (4)

which implicitly corresponds to a least squares fit regression (without the first part). Compared to the more general support vector regression framework, the least square fit does use all the points to fit the regression, while SVM regression leads to a sparse solution where only a limited number of support vectors determines the fit.

**2.2 A Simple Example: The sinc**

In this section, We simulate a cardinal sine function with noise and we train using RBF kernel in order to approximate the function. We use a training and a test set. We train the model for arbitrary values of γ (100) and σ2 (1) and visualize the results in the following figure.

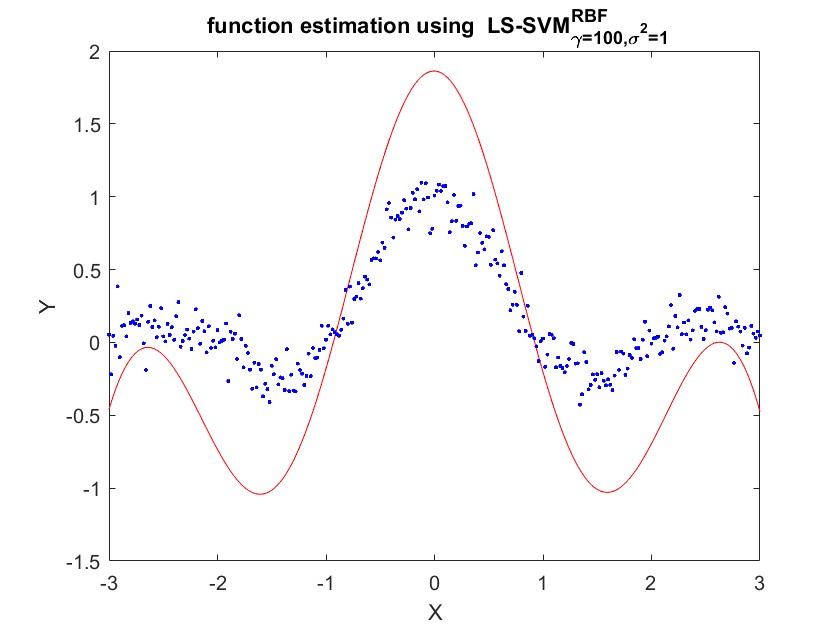


Figure 5: RBF kernel for regression on the training set

The results of applying the trained model on the test set are calculated and visualized on

Figure 6.

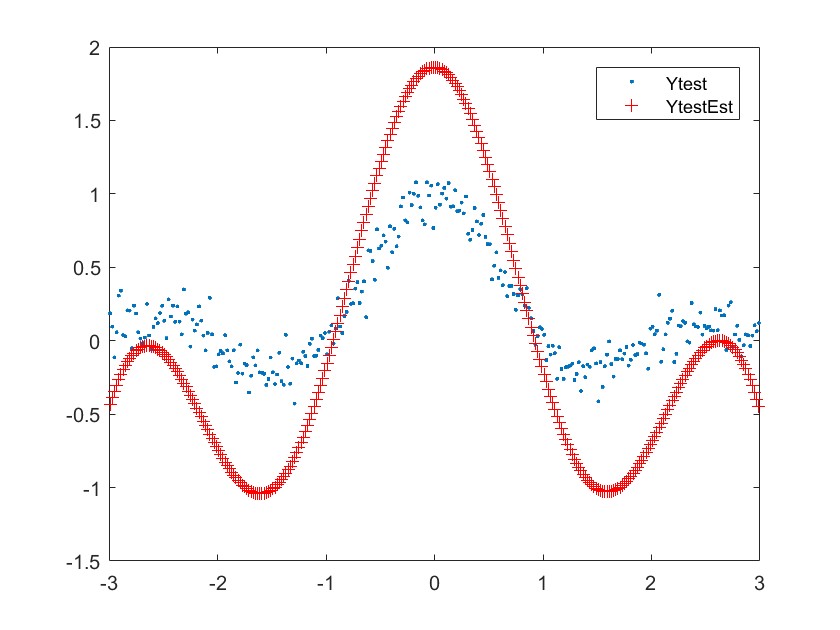


Figure 6: RBF kernel for regression on the test set

We have used arbitrary values of the hyper parameters. We also run a function estimation for a range of values for γ and σ2 and the following Figures highlights the estimated functions for 6 cases on the training and the test sets. It is obvious that it is possible to optimize the parameters according to a certain criteria so that the function estimated is the closest from the true function. However, as the hyper-parameters are continuous, there might be an infinite numbers of cases leading to similar mean squared error.

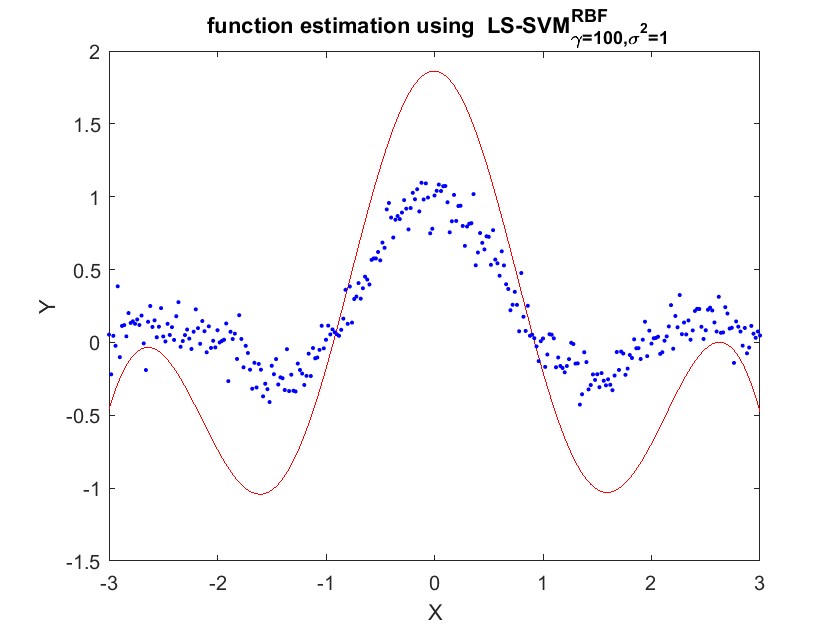
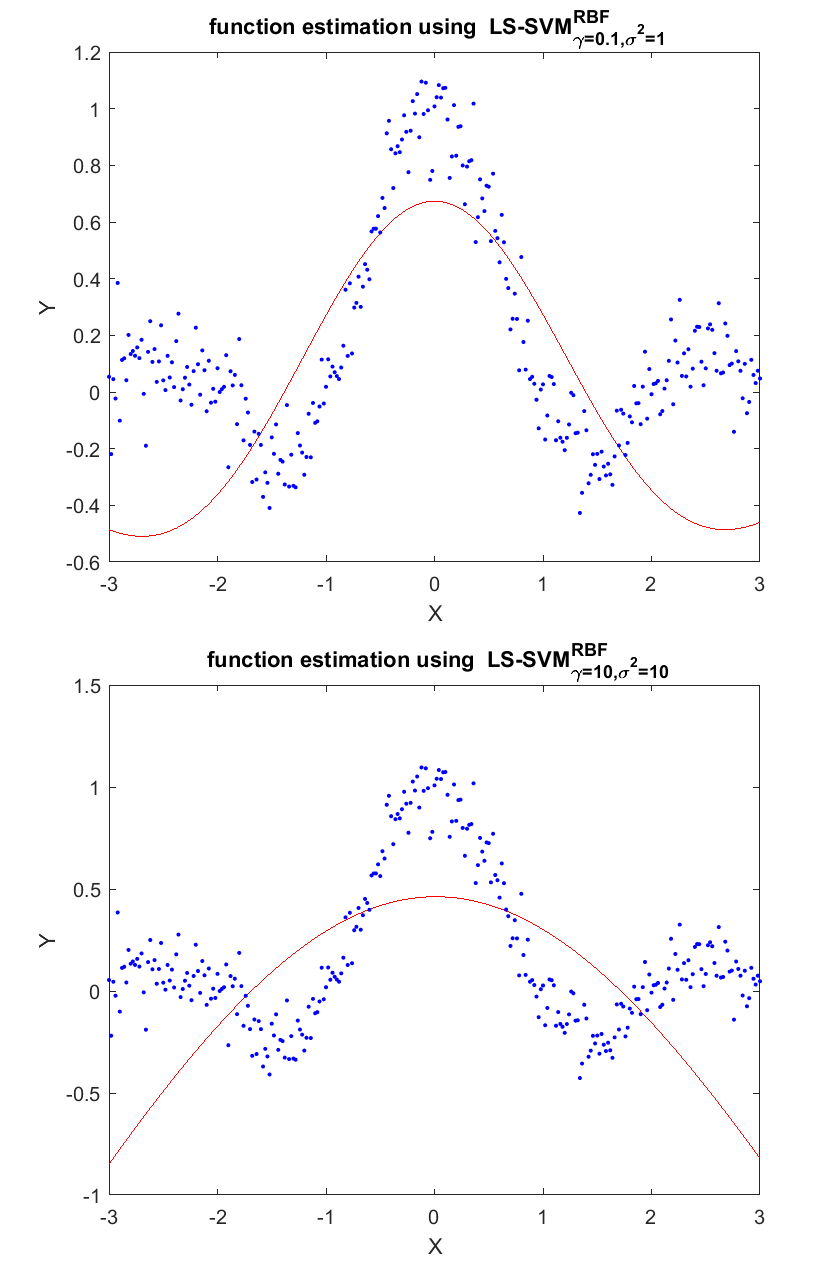
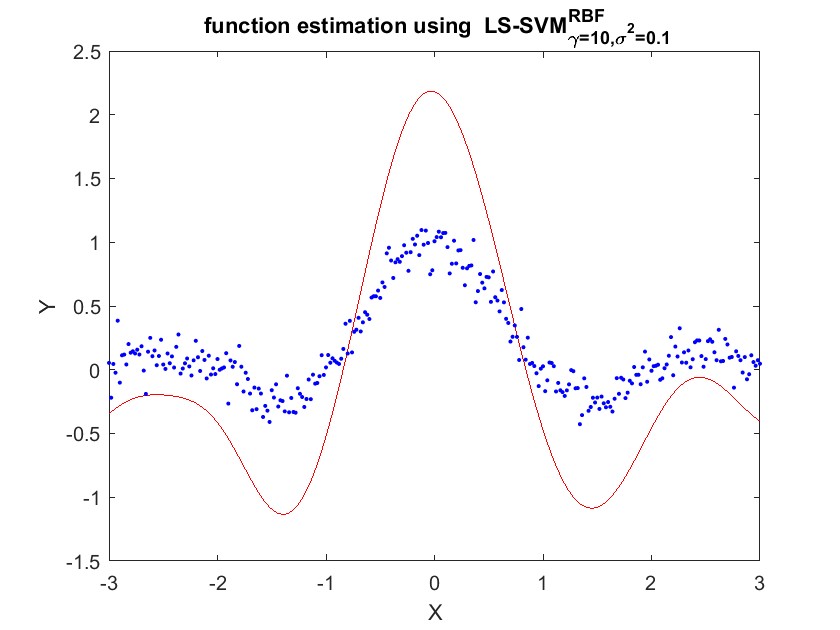
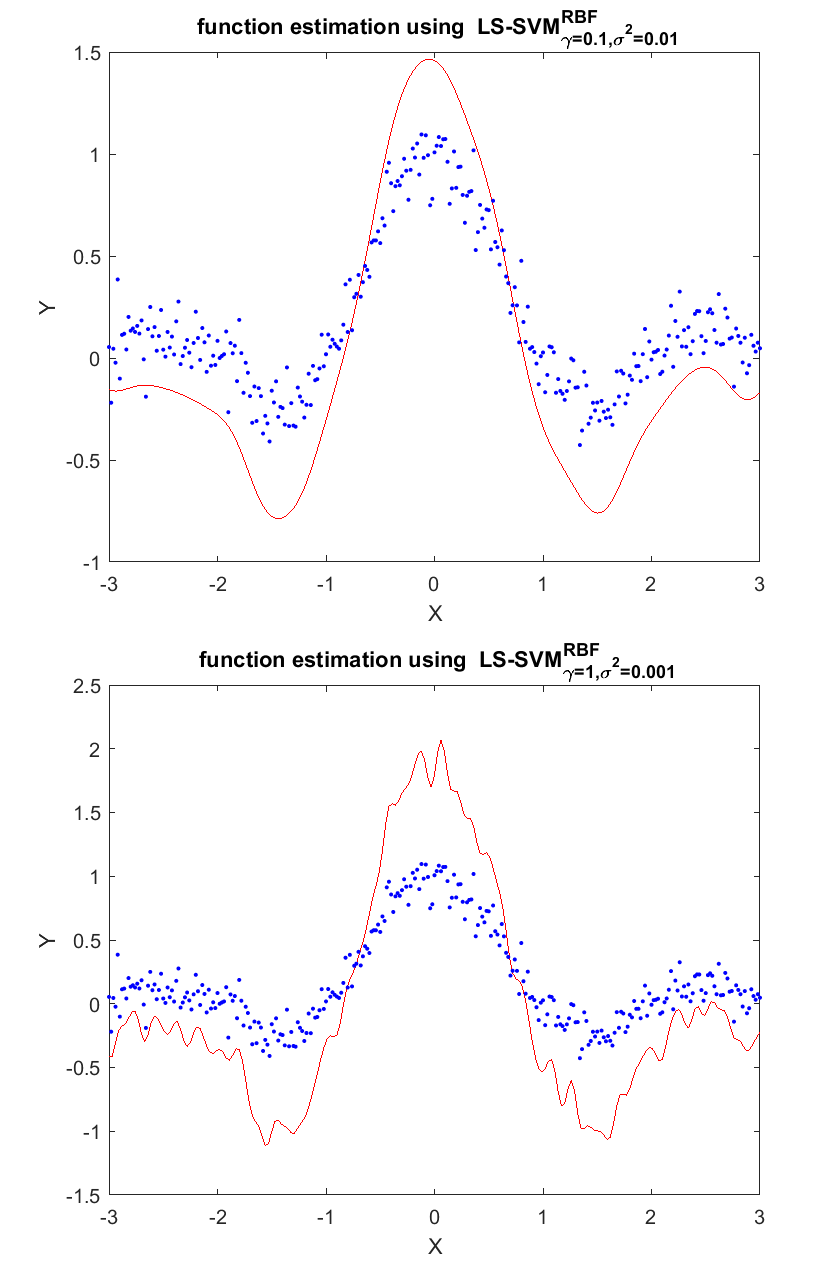


Figure 7: RBF kernels for different values of γ and σ2 on the training set

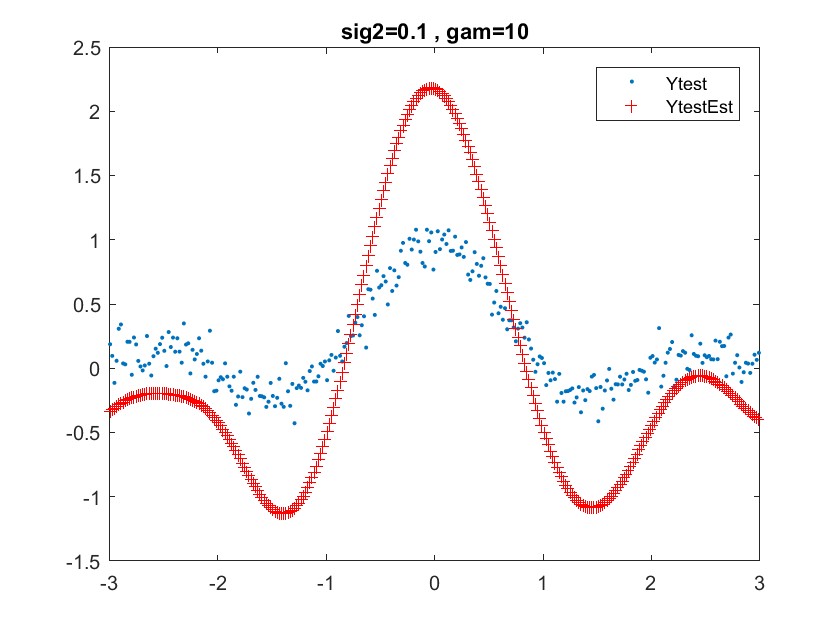
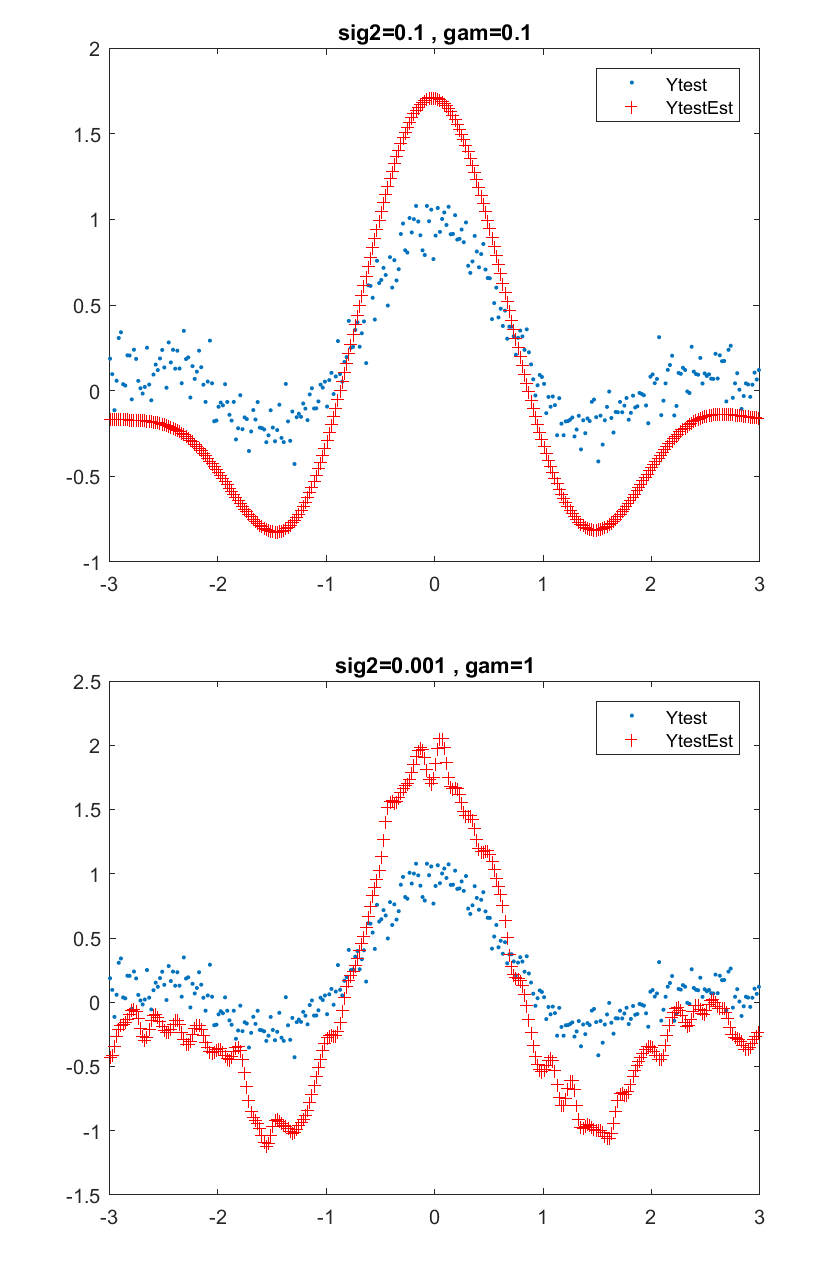
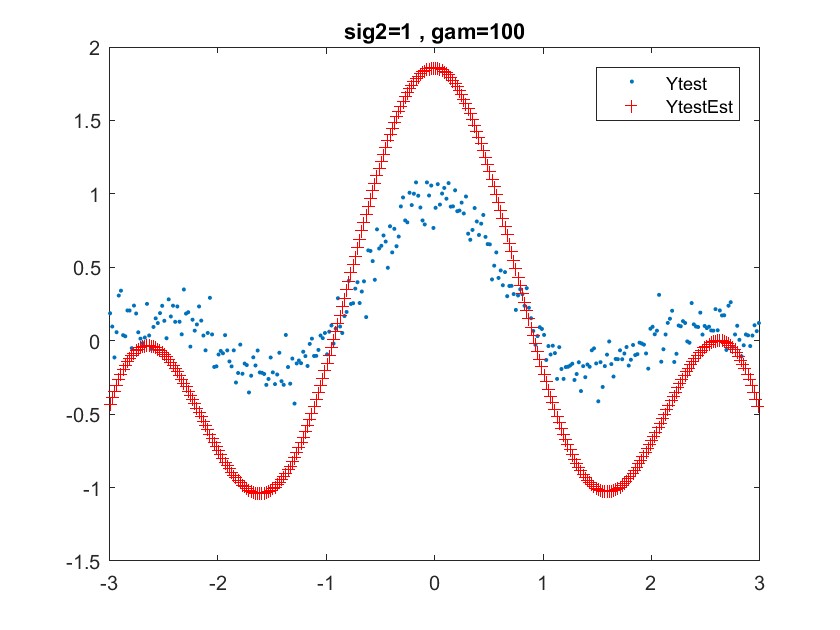
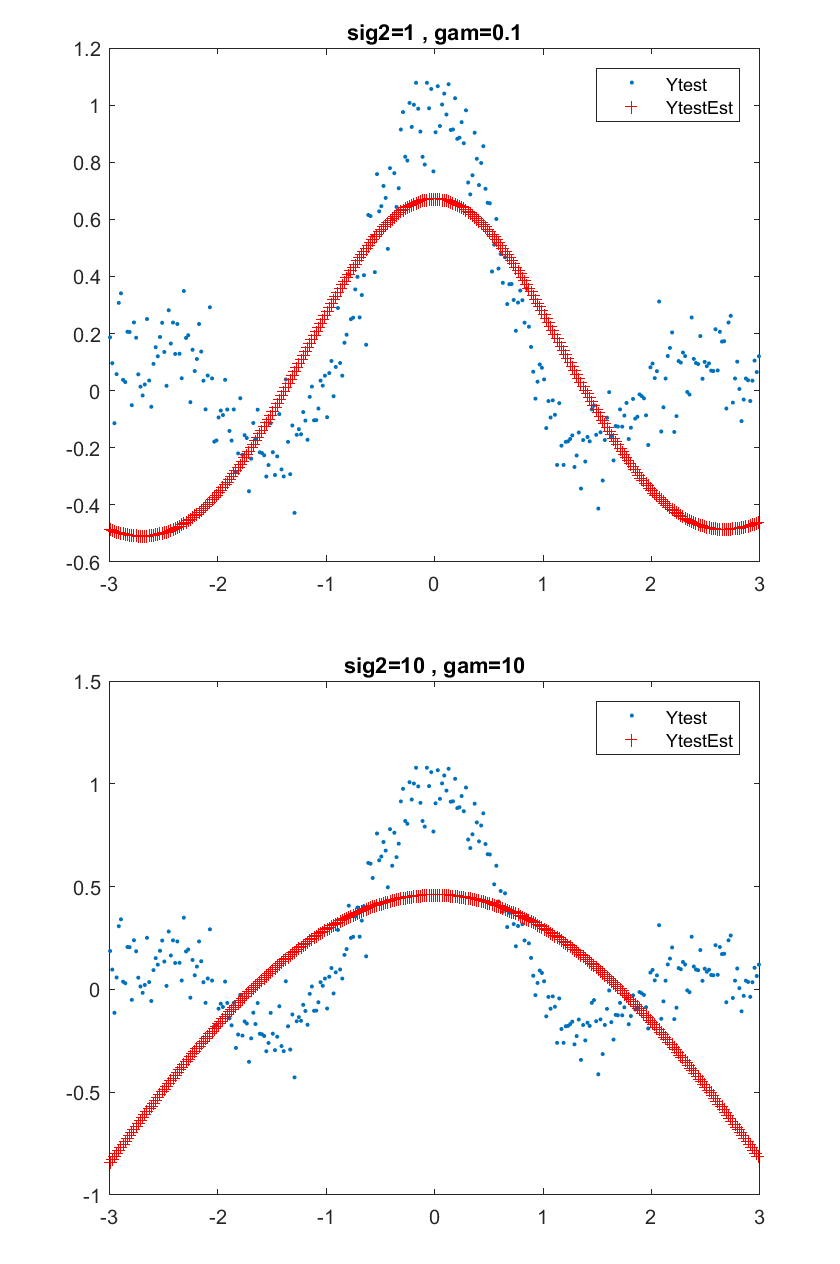


Figure 8: RBF kernels for different values of γ and σ2 on the test set

**2.3 Hyper-parameter Tuning**

In this section, we study the tuning algorithm used to optimize the parameters of the RBF kernel (Figure 8). We run the tuning algorithm several times and compare the obtained values of the hyper parameters. This is depicted in Figure 5.

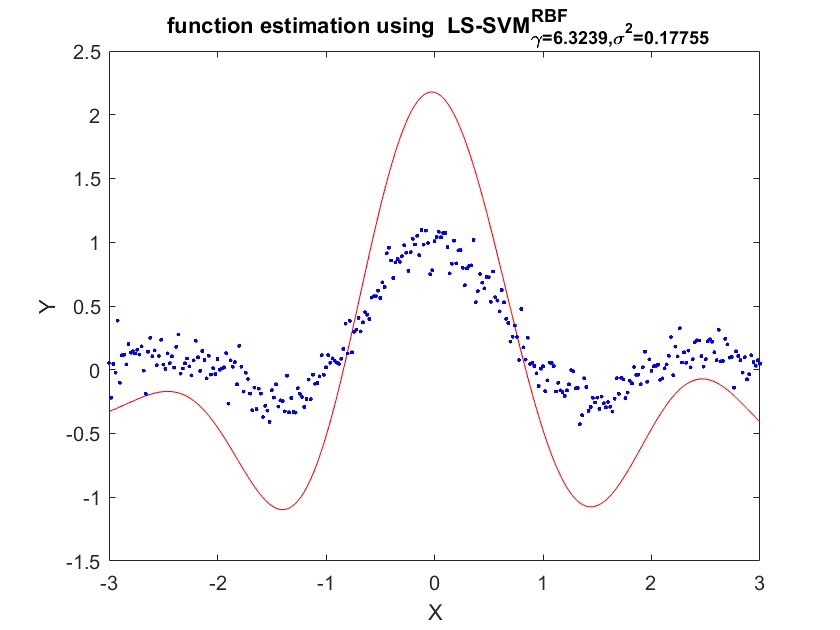


Figure 9: Optimized estimated function (gridsearch)

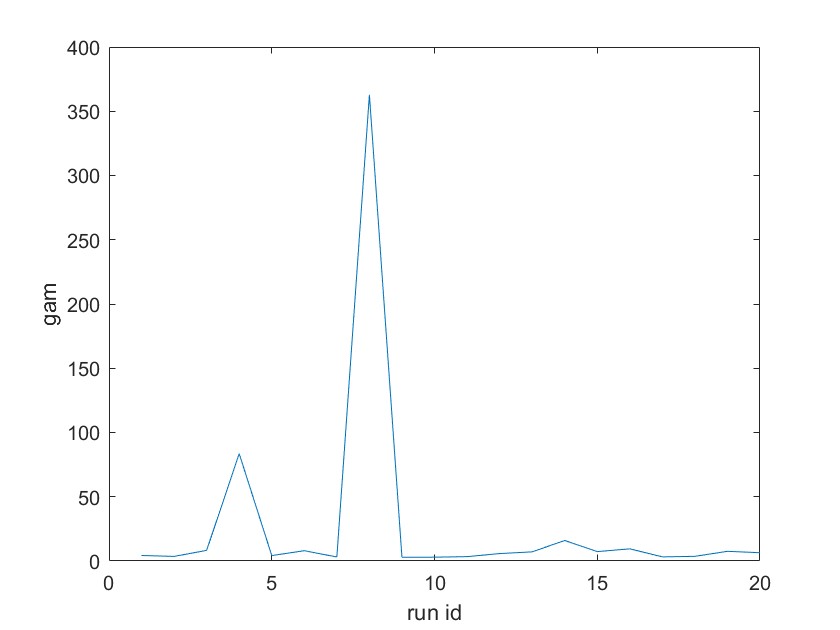
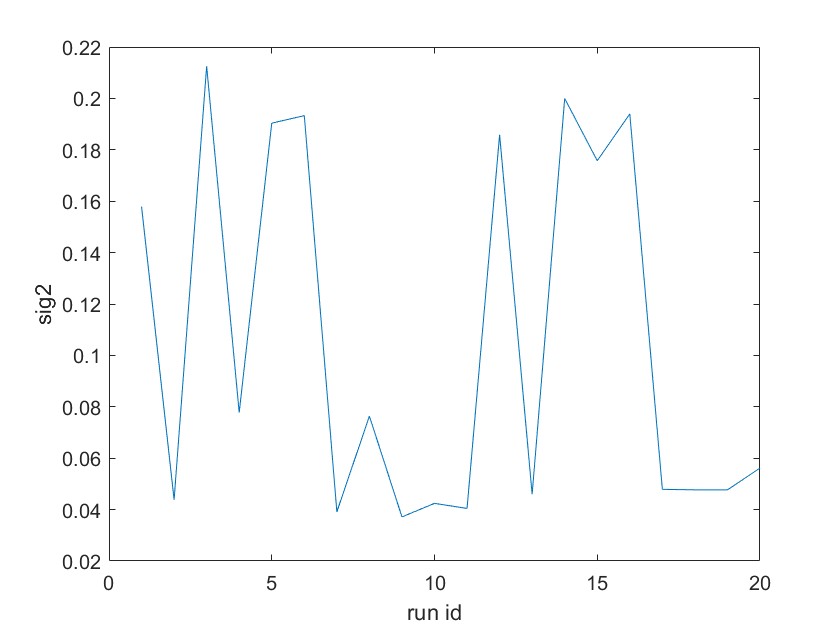


Figure 10: Optimized values of γ and σ2 when running the tuning algorithms several times

(gridsearch)

We observe that the estimated optimal parameters change a lot from one run to another but have little influence on the cost of the hyper parameters (Figure 10). We also observe that although optimal parameters can change, the estimated regression doesn’t vary a lot (Figure11).

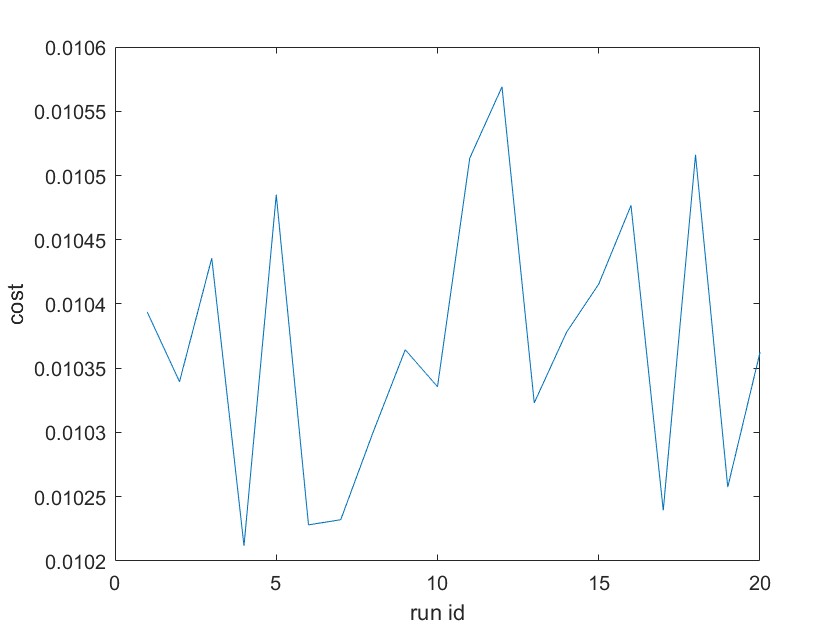
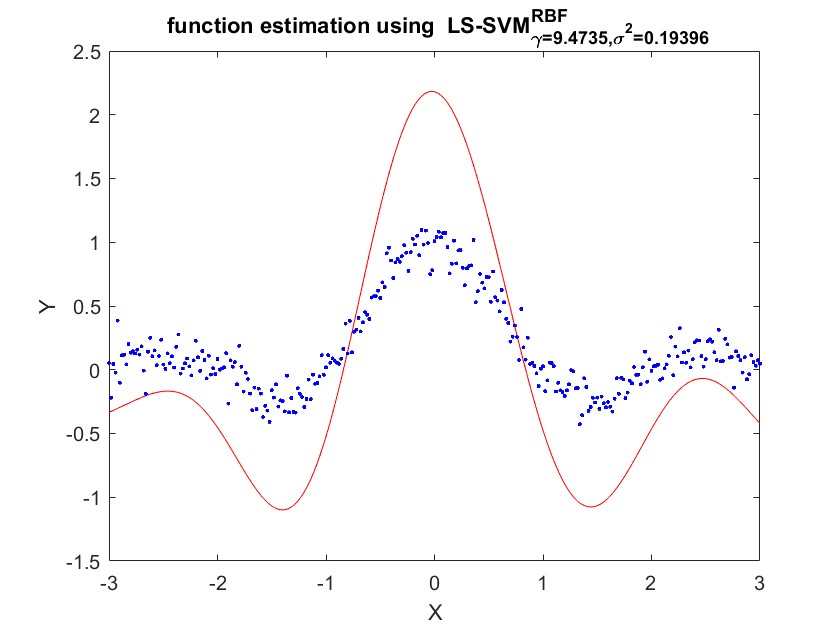
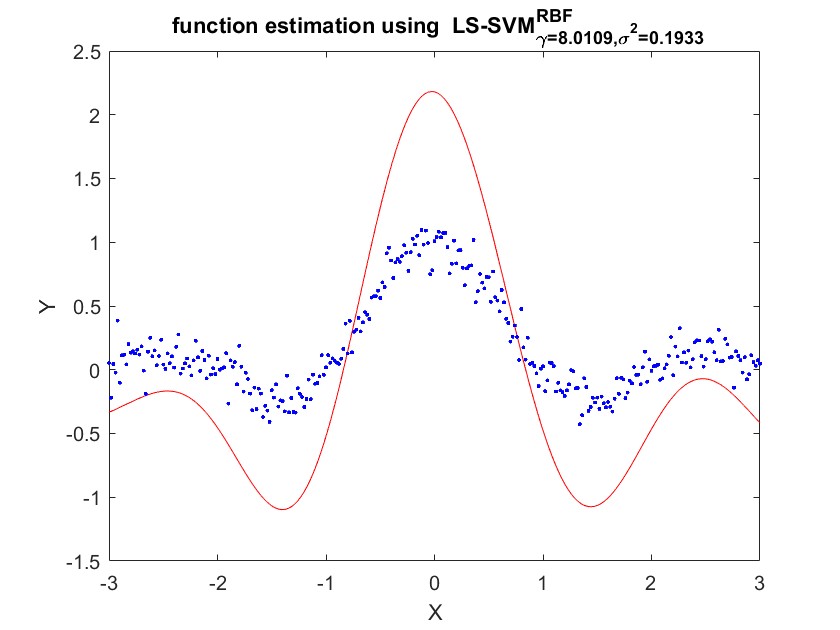


Figure 11: Cost of hyper parameters when running the tuning algorithms several times



(gridsearch)

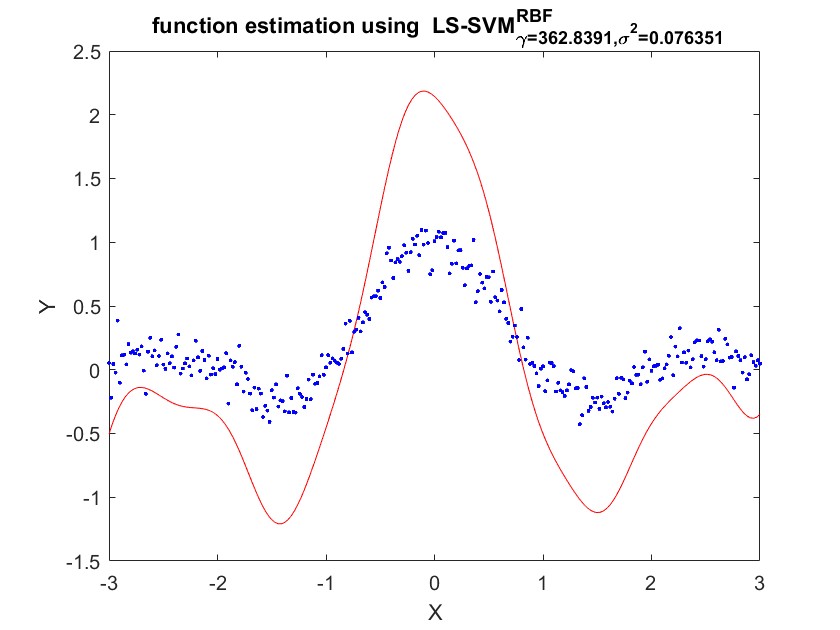
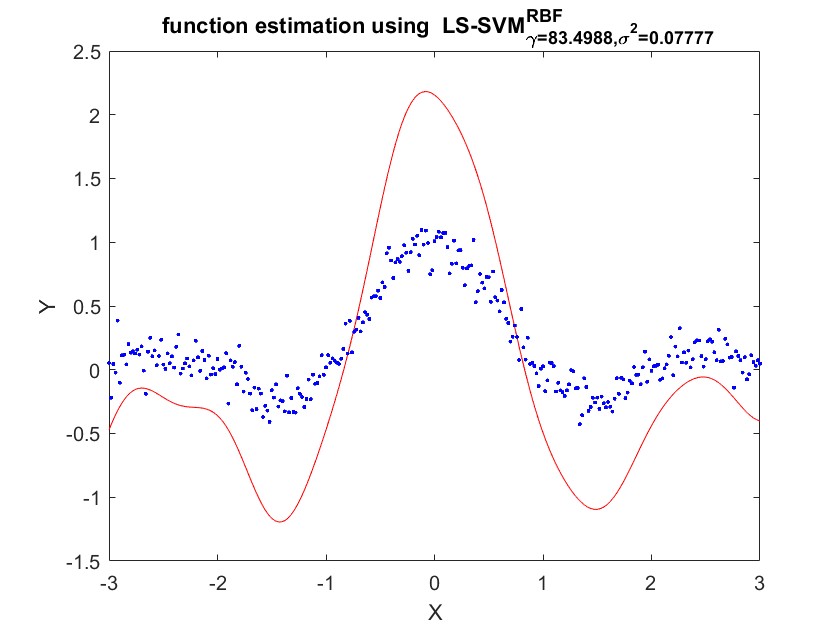


Figure 12: Estimated function when running the tuning algorithm several times (gridsearch)

When we change the optimization procedure in the tuning function to simplex and re-estimate the optimal model, we obtain the following result.

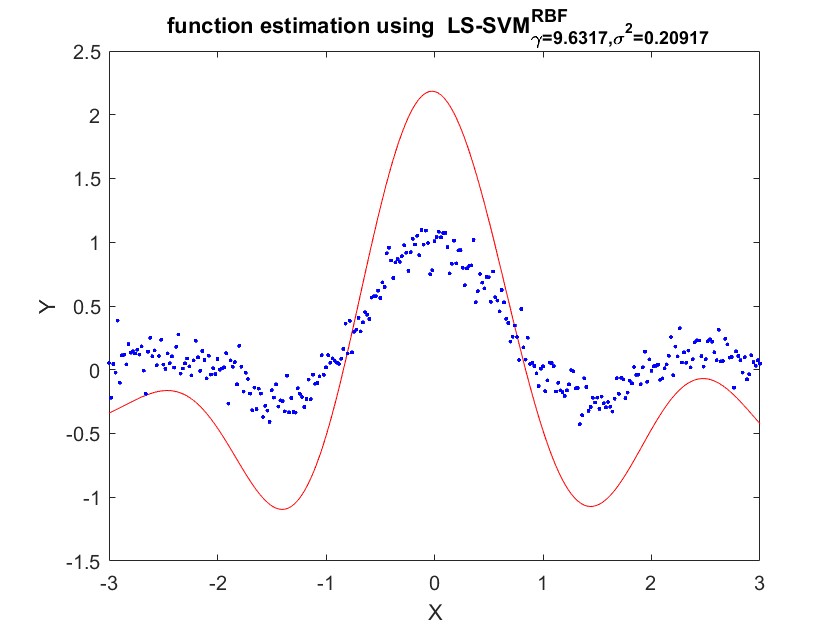


Figure 13: Optimized estimated function (simplex) Again, we run the tuning algorithm and obtain the following pictures.

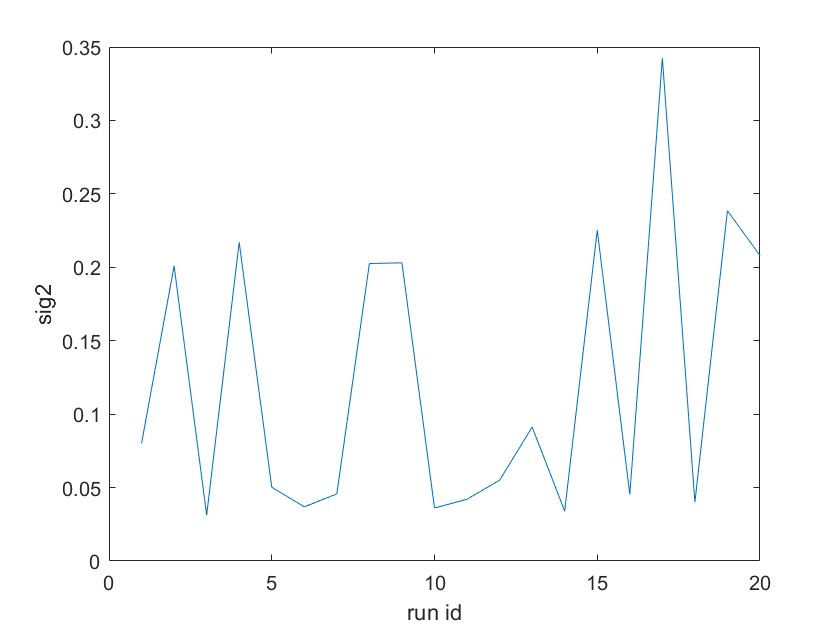
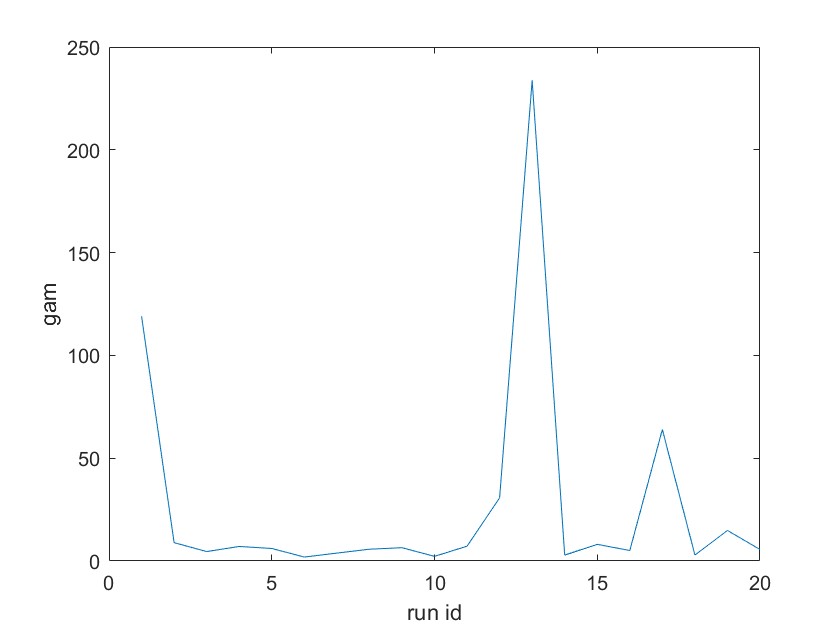


Figure 14: Optimized values of γ and σ2 when running the tuning algorithms several times

(simplex)

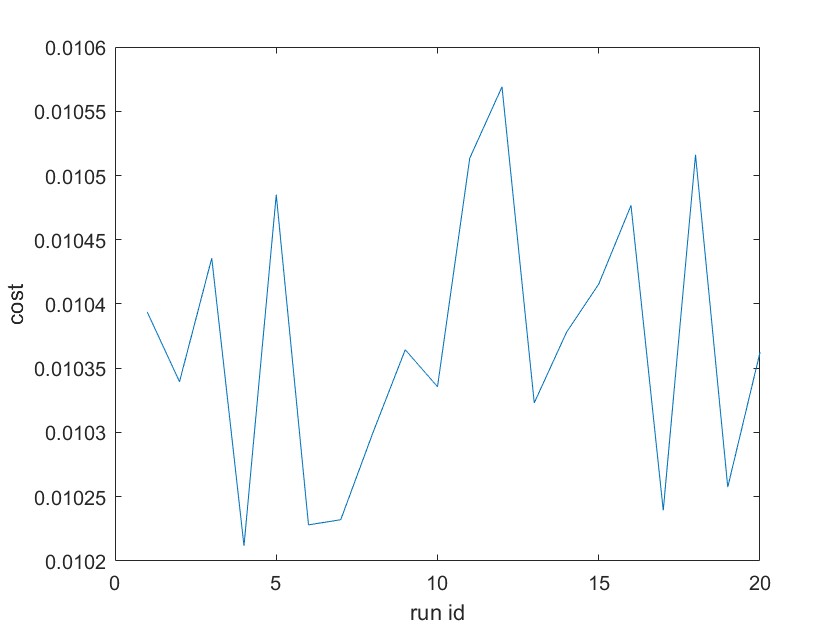


Figure 15: Cost of hyper parameters when running the tuning algorithms several times

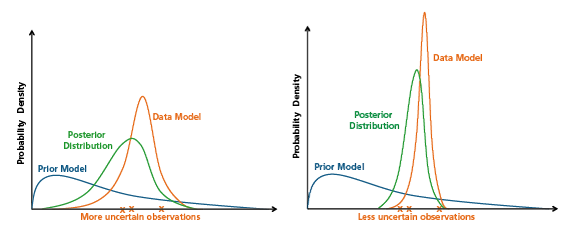
(simplex)

Again, optimized values vary heavily, but the estimated functions are very similar. For a given starting point (calculated based on csa), gridsearch exhaustively considers all parameter combinations for a grid around the starting point. The start-values determine the limits of the grid over parameter space. The Nelder-Mead method or downhill simplex method is a commonly applied numerical method used to find the minimum or maximum of an objective function in a multidimensional space. It is applied to nonlinear optimization problems for which derivatives may not be known. The method uses the concept of a simplex, which is a special polytope of n + 1 vertices in n dimensions. The method approximates a local optimum of a problem with n variables when the objective function varies smoothly and is unimodal.

**1.4 Application of the Bayesian Framework**

The Bayesian framework now can be used to tune and to analyze the LS-SVM regressor. The basic result from the Bayesian framework for the LS-SVM is the derivation of the probability that the data points are generated by the given model. This is called the posterior probability. This probability criterion is expressed as a number. There are 3 variants: the posterior with respect to the model parameters α and b, the posterior with respect to the regularization constant and the posterior with respect to the choice of the kernel and its parameter.

This three-levels principle can be schematized by the following picture:



**Fig-Schematic of a Bayesian "update" of a prior model from a data model describing the observations as relatively uncertain (left panel) and relatively certain (right panel), resulting in posterior distributions with more and less uncertainty, respectively.**

Provided data set D, a model H with parameter vector w and so-called hyper-parameters or regularization parameters θ, Bayesian inference is constructed with 3 levels of inference:

• In level 1, for a given value of θ, the first level of inference infers the posterior distribution of w by Bayesian rule:

p(w | D, θ, H ) ∝ p(D | w, H )p(w | θ, H ) (5)

• The second level of inference determines the value of θ, by maximizing

p(θ | D, H ) ∝ p(D | θ, H )p(θ | H ) (6)

• The third level of inference in the evidence framework ranks different models by examining their posterior probabilities:

p(H | D) ∝ p(D | H )p(H ) (7)

For regression, the error bars can be computed using Bayesian inference (see Figure 17)

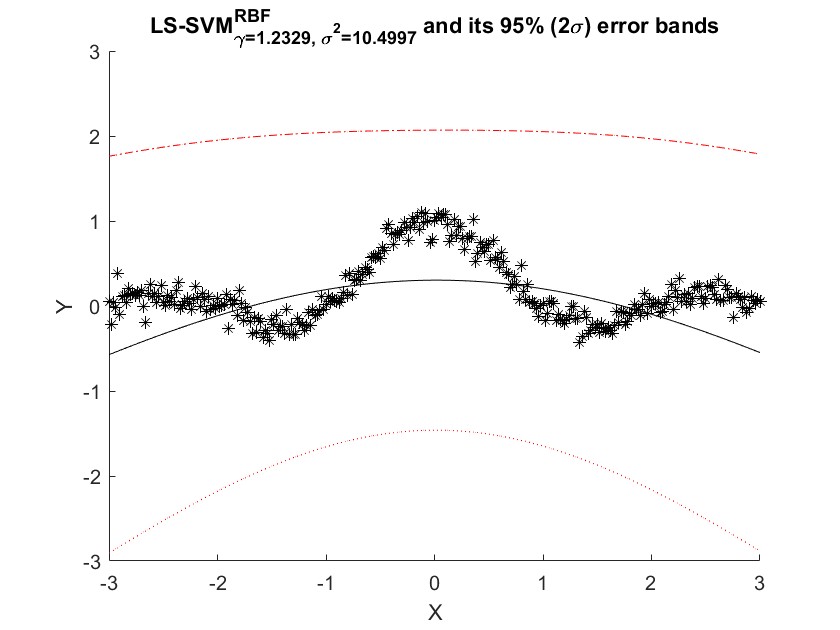


Figure 17: Error bars for Bayesian inference

For classification, it is also possible to get probability estimates. We load the Iris data-set and calculate the posterior class probabilities. Results are displayed at the following figure:

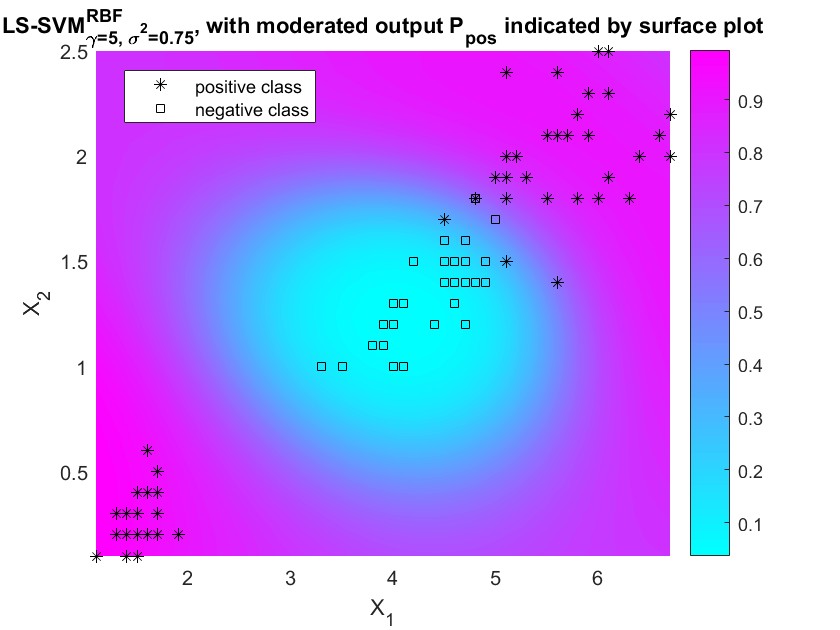
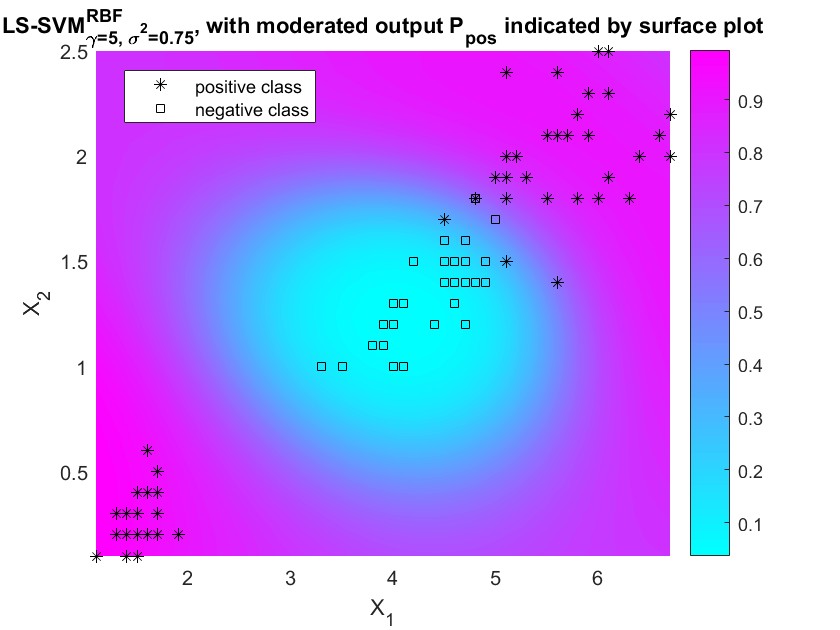
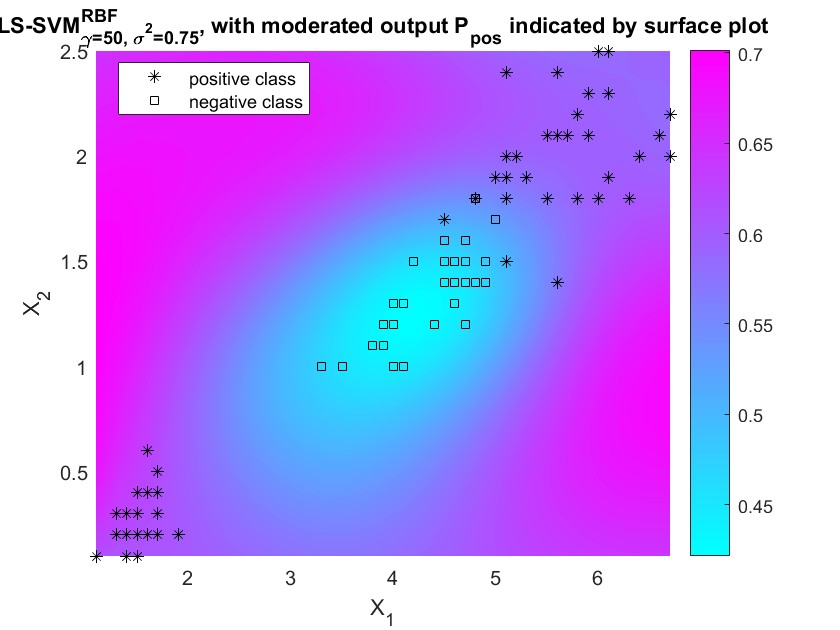
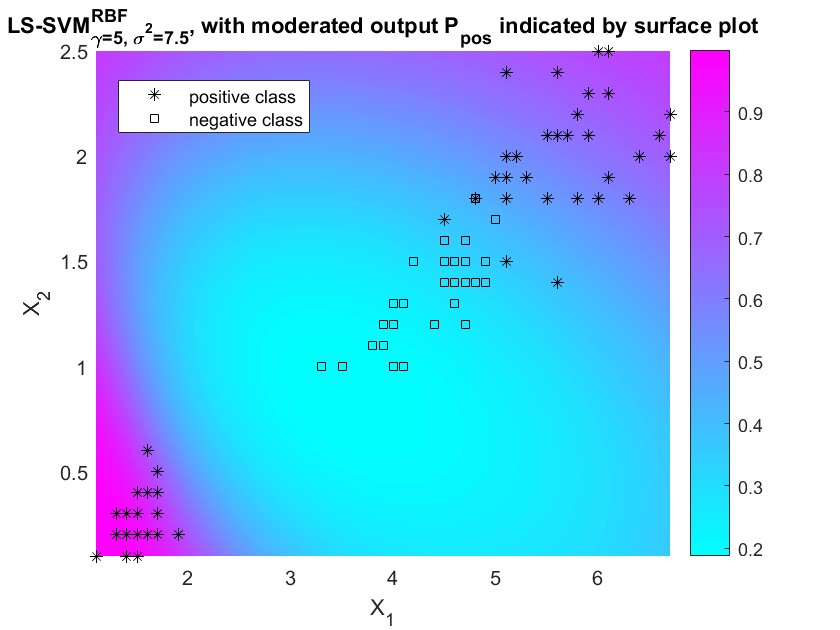
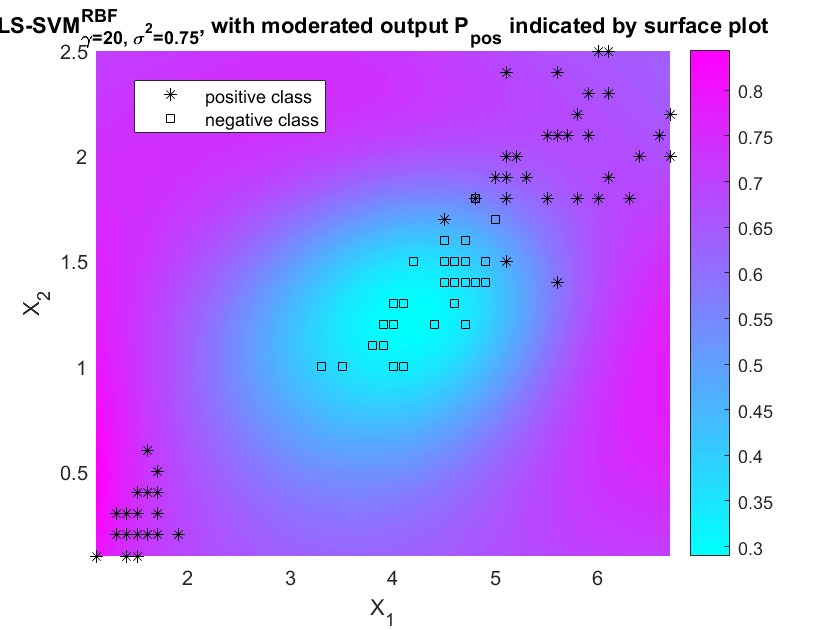


Figure 18: Posterior class probabilities

Colors represent the probability that a point belongs to the positive class. Figure 19 represents the same probabilities when we change the values of the hyper-parameters. Obviously, for different values, the probabilities change. As the values move away from the optimal values, we can observe that the obtained posterior probabilities don’t match the empirical labels

**Figure 19: Posterior class probabilities for different values of the hyper-parameters**



The Bayesian framework can also be used to select the most relevant inputs by Automatic Relevance Determination (ARD). In a backward variable selection, the third level of inference of the Bayesian framework is used to infer the most relevant inputs of the problem. Only the most explanatory variables will be selected.

ARD assigns a different weighting parameter to each dimension in the kernel and optimizes this using the third level of inference. Per the used kernel, one can remove inputs corresponding the larger or smaller kernel parameters. In each step, the input with the largest optimal σ2 is removed (backward selection). For every step, the generalization performance is approximated by the cost associated with the third level of Bayesian inference.

An easy way to represent the results is to plot the cost associated with third level of inference for each feature (Figure 20). Features 1 and 3 have the higher cost (the generalization performance is approximated by the cost associated with the third level of Bayesian inference.) and are therefore selected.

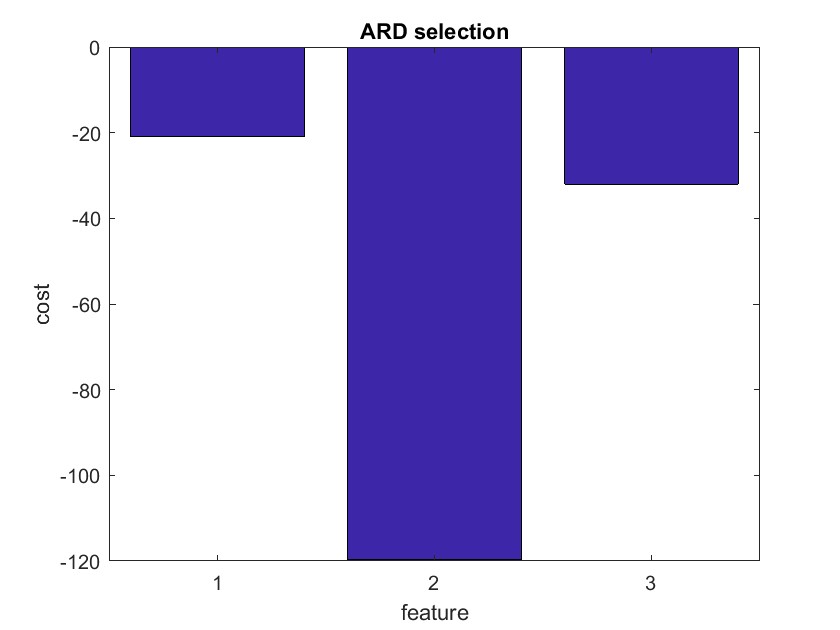


Figure 20: ARD selection

Cross-validation can be used instead of the Automatic Relevance Determination. Selecting each possible combination of variables, tuning the hyper-parameters and calculating the cost function of the cross-validation procedure leads to the same result (selecting the number of variables that correspond to the lowest cost function of the cross-validation procedure). To fully mimic the backward selection procedure, we can start with all possible variables, exclude sequentially only one variable, select the model that correspond to the lowest cost and start again the backward procedure.

**1.5 Robust Regression**

In this section, we study the impact of outliers on the regression. In situations where the data is corrupted with non-Gaussian noise or outliers, it becomes important to incorporate robustness into the estimation.

The Figure 21 highlights the impact of outliers on the estimation of the regression.

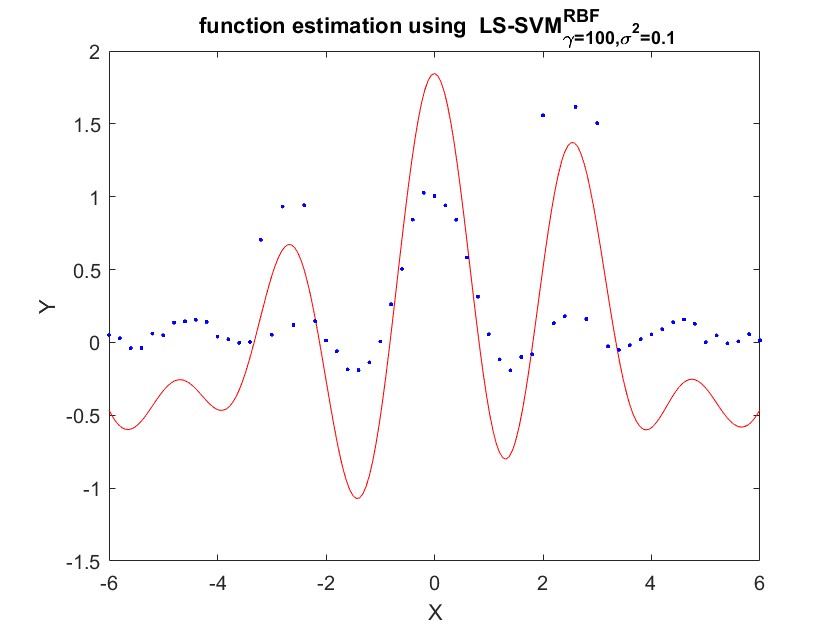
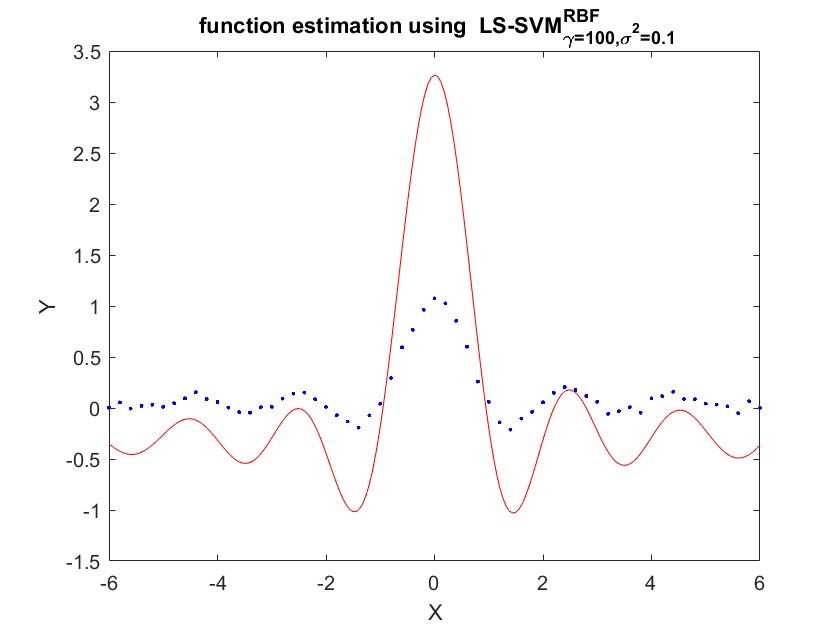


Figure 21: Regression with outliers (left) and without (right)

Outliers impact the regression locally (where outliers are). Learning observations with outliers without awareness may lead to fitting those unwanted data and may corrupt the approximation function. This will result in the loss of generalization performance in the test phase.

Outliers are dealt with in SVM regression by using robust cross-validation and smoothing the impact of outliers. Figure 22 shows the result of a robust cross-validation and the use of a smoothing function for the outliers. We can see that the outliers don’t impact the regression anymore (or at least to a lesser extent).

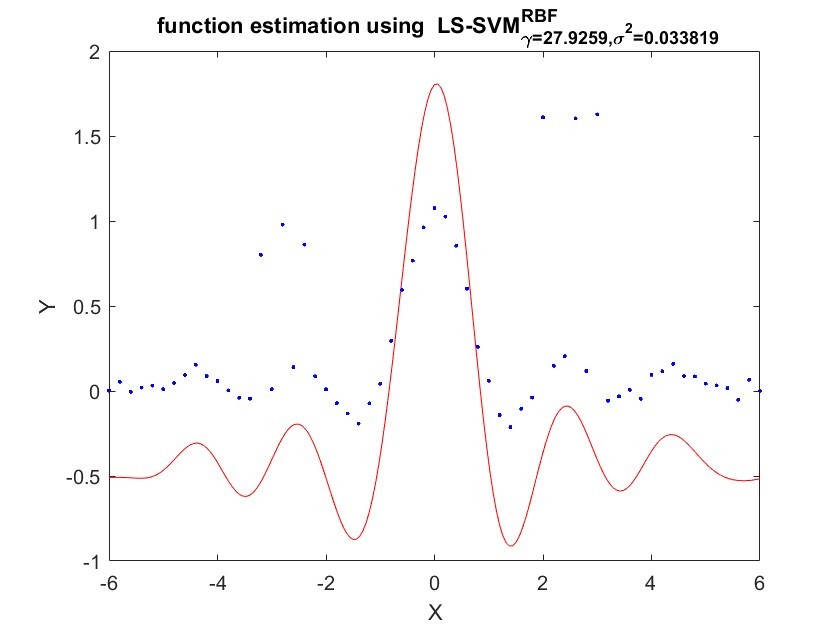


Figure 22: Robust cross-validation and outliers smoothing

Both mean squared error (MSE) and mean absolute error (MAE) are used in predictive modeling. MSE has nice mathematical properties which makes it easier to compute the gradient. However, MAE requires more complicated tools such as linear programming to compute the gradient. Because of the square, large errors have relatively greater influence on MSE than do the smaller error. Therefore, MAE is more robust to outliers since it does not make use of square.

Different weighting functions can be used in robust regression. Figure 23 describes the four weighting functions available with the tuning algorithm for robust regression. As an example, the weighting function Hampel completely ignores points further than a certain threshold (b2).

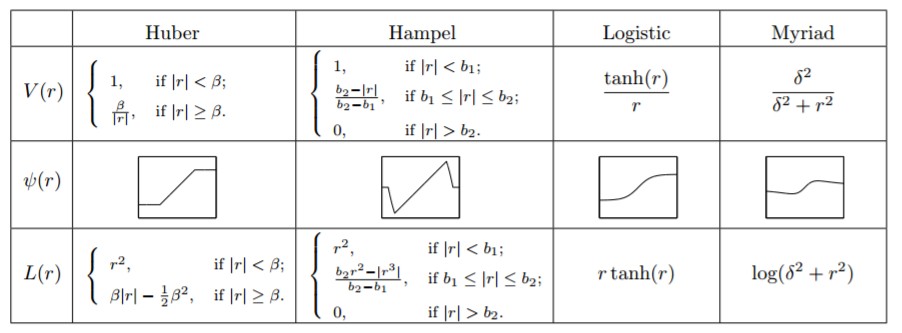


Figure 23: Weighting functions for robust regression

The following figure represents the results of different weighting functions for robust regression (top-left

= Huber, top-right = Hampel, bottom-left = Logistic, bottom-right = Myriad). Comparison of the results of the different weighting functions is made difficult due to the instability of the tuning algorithm (optimal values are different every time).

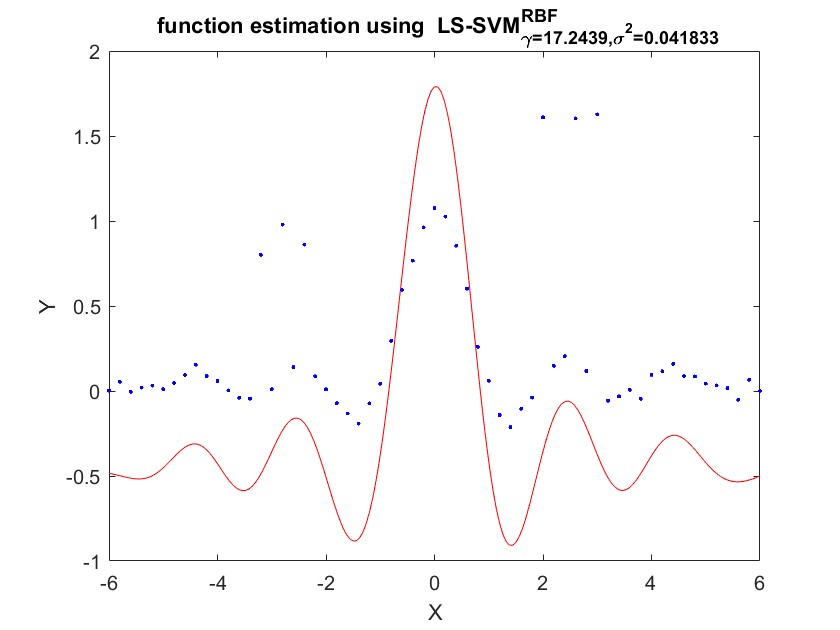
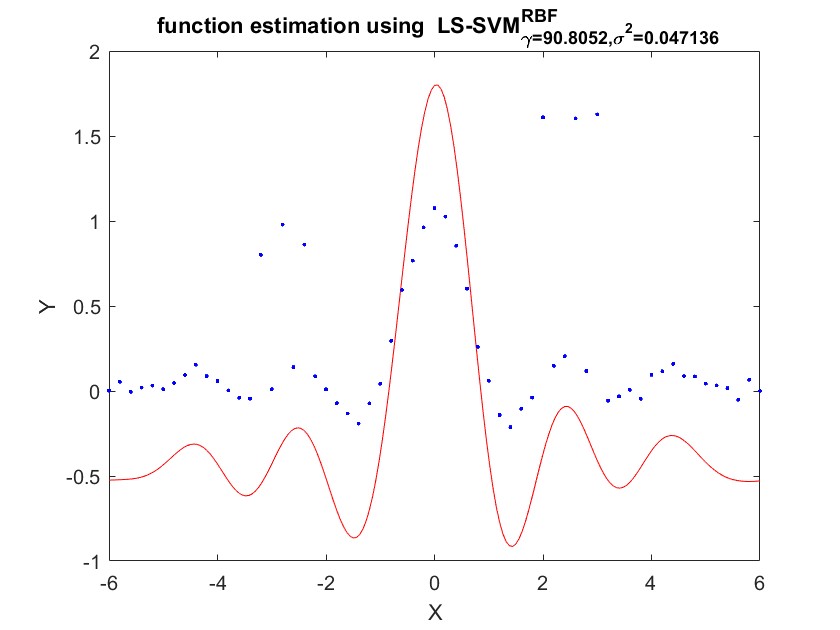
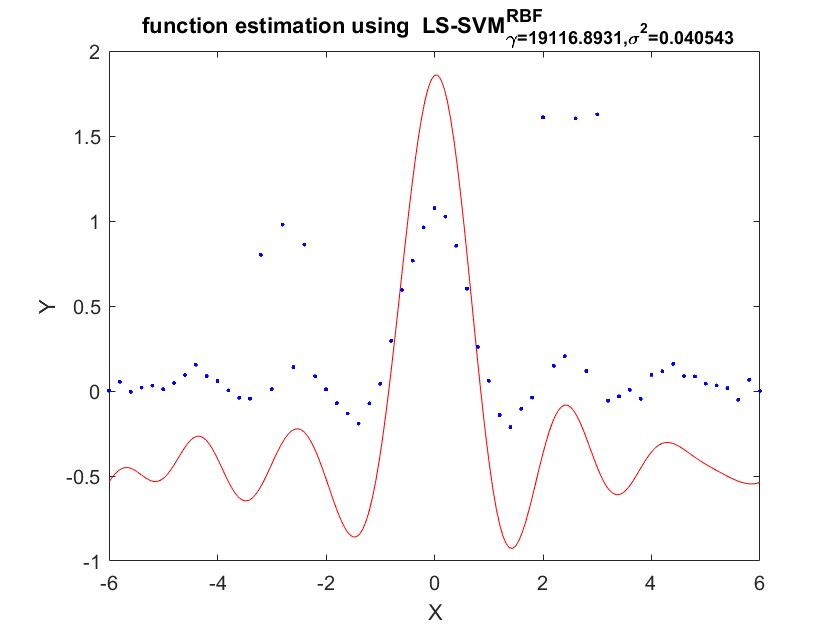
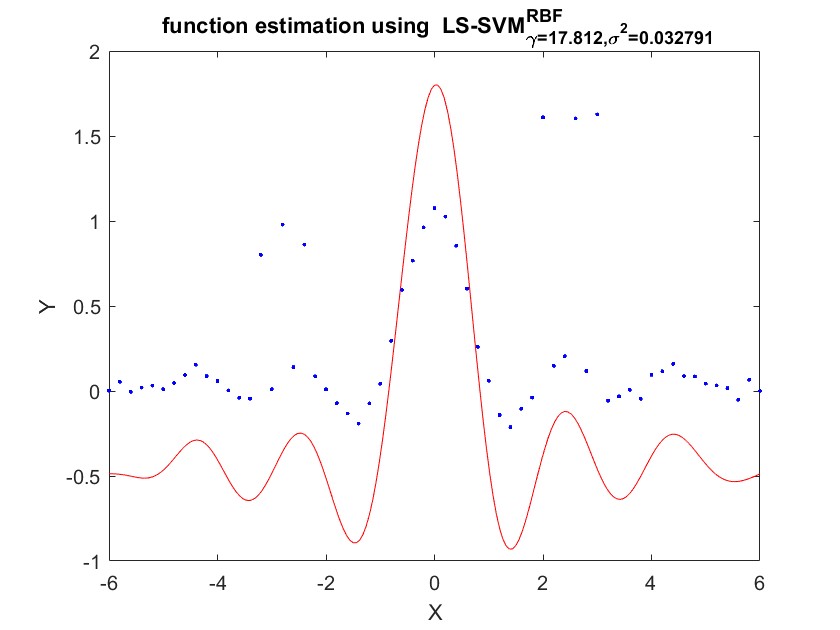


Figure 24: Results of different weighting functions for robust regression

**2.6 Homework Problem**

2.6.1 Introduction: Time-series Prediction

In this section, we investigate the use of SVM regression for time series model. The linear auto-regressive (AR) model of a process Zt with t = 1, 2, ..., ∞ is given by

z˜t = a1zt−1 + a2zt−2 + ... + anzt−n (8)

with ai ∈ R for i = 1, ..., n and n the model order. The nonlinear variant (NAR) is described as:

z˜t = f (zt−1, zt−2, ..., zt−n) (9)

The time-series identification can be written as a classical black-box regression modeling problem:

y˜t = f (xt) (10)

with yt = zt and xt = [zt−1, zt−2, ..., zt−n].

Any provided sequence Z can be mapped into a regression problem. By using the command windowize, we can transform the logmap data into the required format.

A very important prerequisite is to normalize the data. Support vector machines usually works best with normalized data (like other machine learning techniques). After training an initial model with random values for the hyper-parameters, we optimize the hyper-parameters by using the cross-validate approach. The Figure 25 displays the result of the prediction for the test set with order = 10. Like the hyper-parameters, order can be optimized to minimize the mean squared error between the prediction and the test set. We run the above procedure for a range of order from 1 to 15 and calculate the mean squared error for each order. The mean squared error as a function of order and the prediction for order = 7 (minimizing MSE) is displayed in Figure 26. It is obvious that the order of the model has an impact on the quality of the prediction (also visually on the prediction itself, not displayed here).

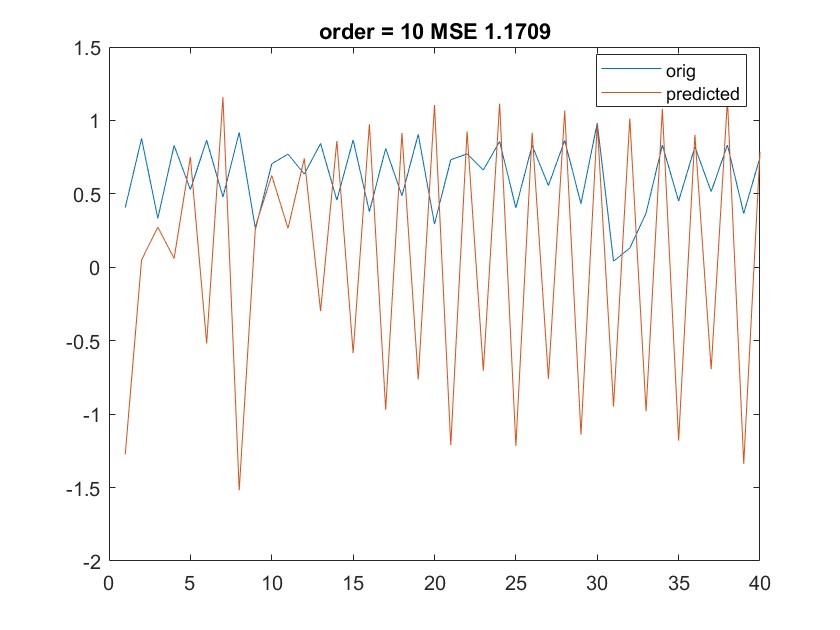


Figure 25: Prediction on the test set for order = 10

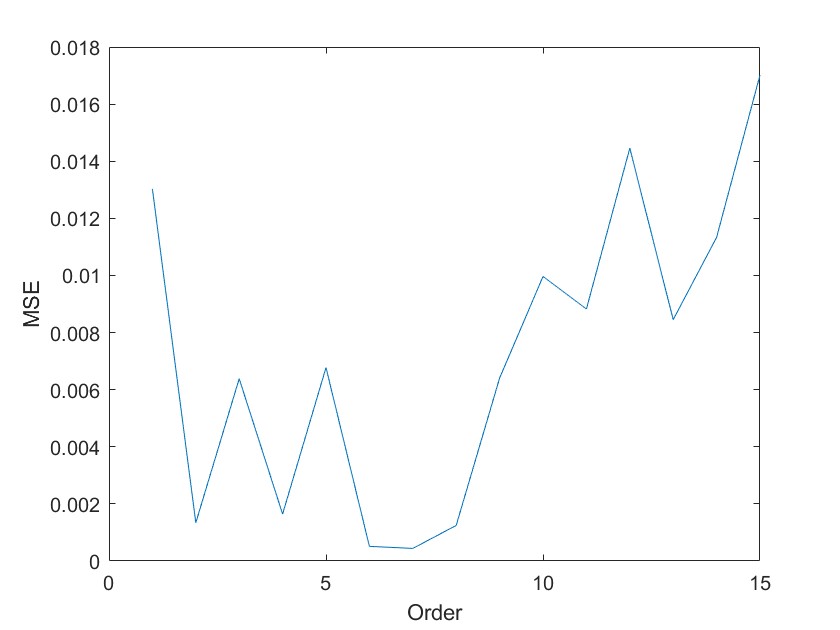
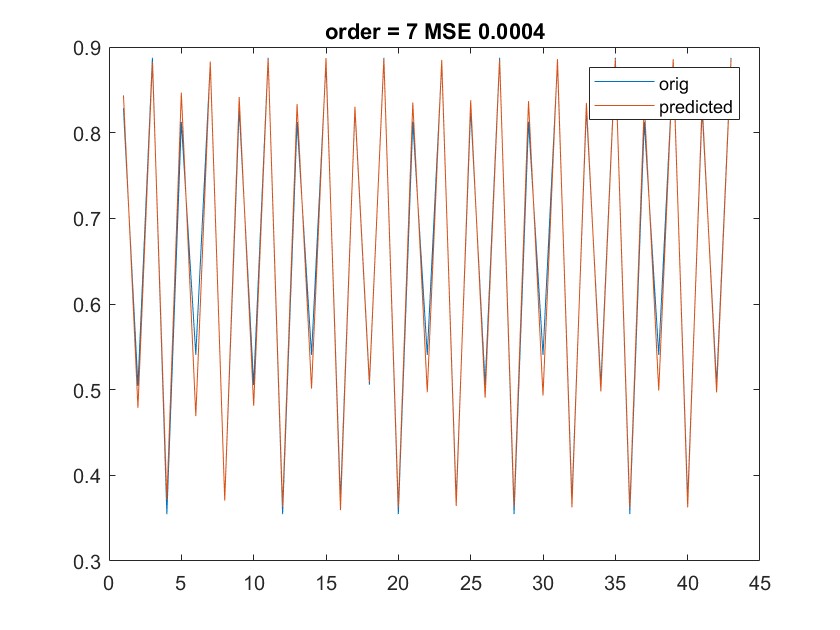


Figure 26: MSE as a function of order and prediction on the test set for order = 7

**2.6.2 Application: Santa Fe Laser Dataset**

The Santa Fe Laser data set is shown in Figure 29. The training set has 1000 points and the test set has 200 points.

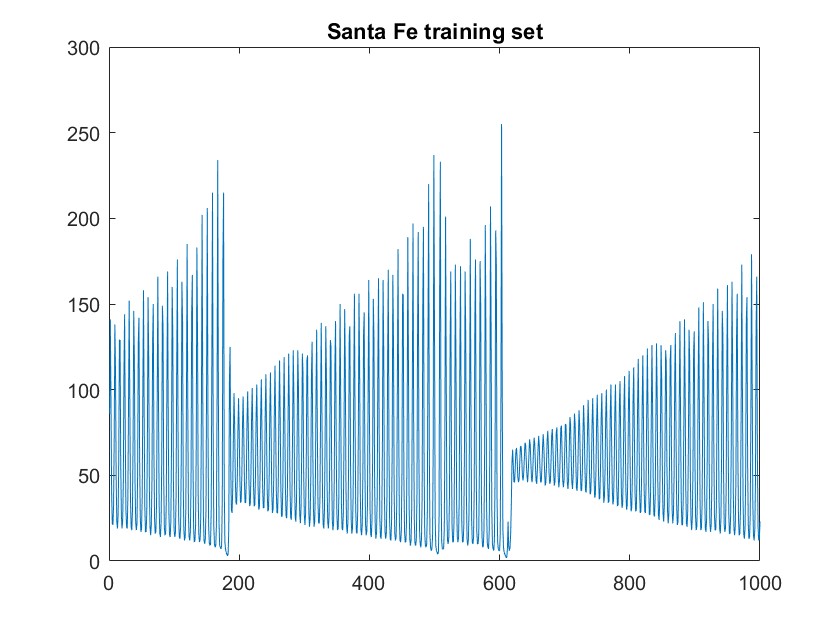
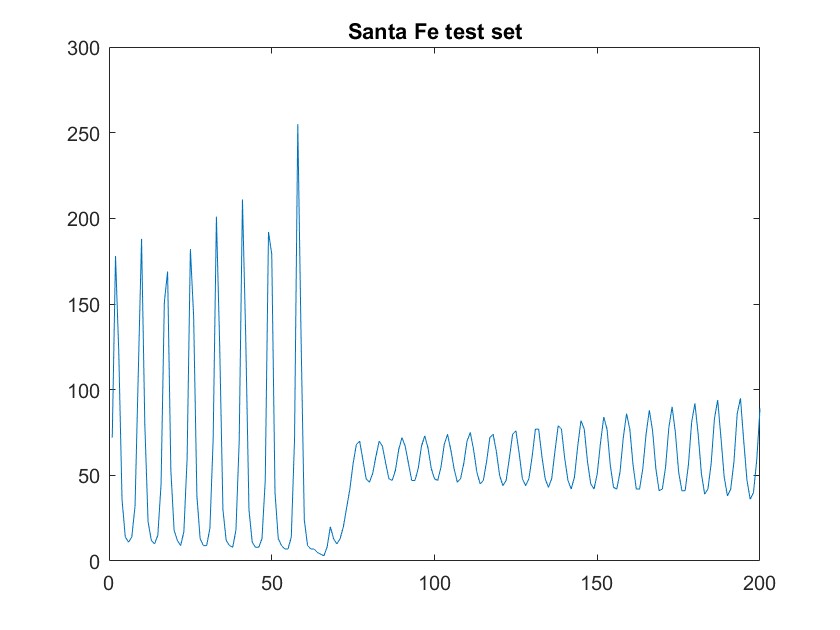


Figure 27: Santa Fe data set

First, we try to fit a model with order = 50. Again, we tune the parameters (γ and σ2) on the training set. The results are displayed in Figure 28. Prediction with order = 50 doesn’t capture the change of amplitude that occurs around point 15. From point 60, we also see that oscillations of the test set and the prediction are not in phase anymore. Therefore, order = 50 is not a good choice for the model.

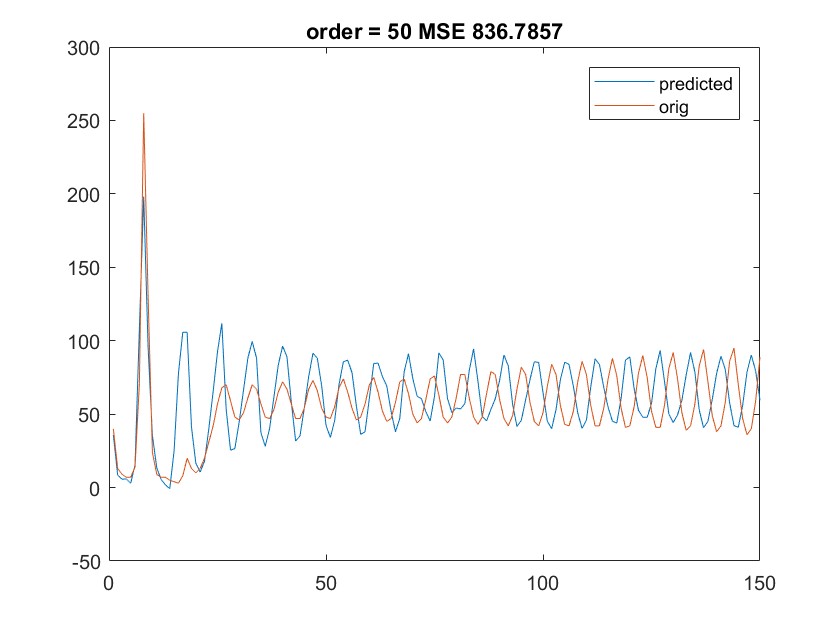


Figure 28: Prediction on the test set for order = 50

In the next step, we try different model orders on the test set and observe the mean squared error (MSE). The results are plotted in Figure 29. The best results are given for an order = 30. The difficulty with the test set is that around point 60, the amplitude of the oscillations change suddenly and it is difficult for the trained model to predict this change. We also noticed that the results of the tuning of the hyper-parameters are not very stable.

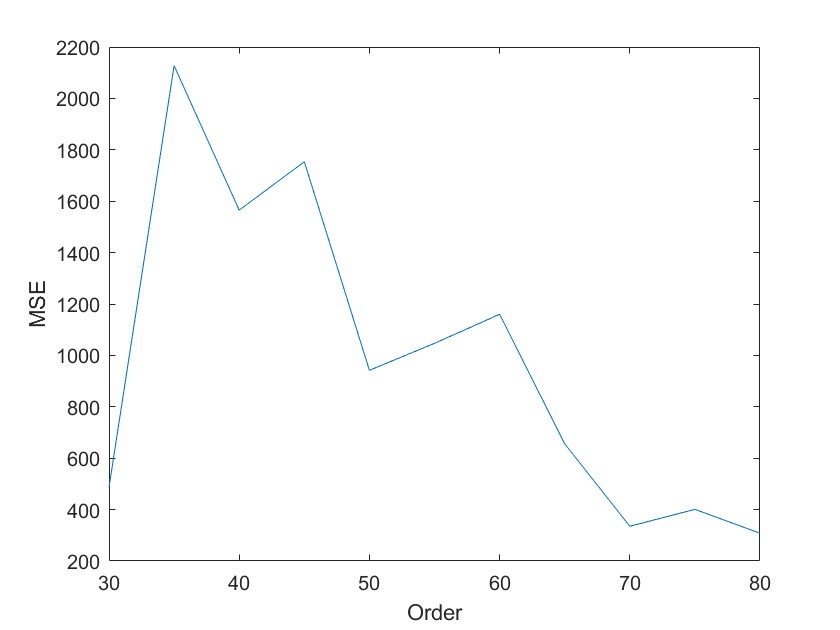
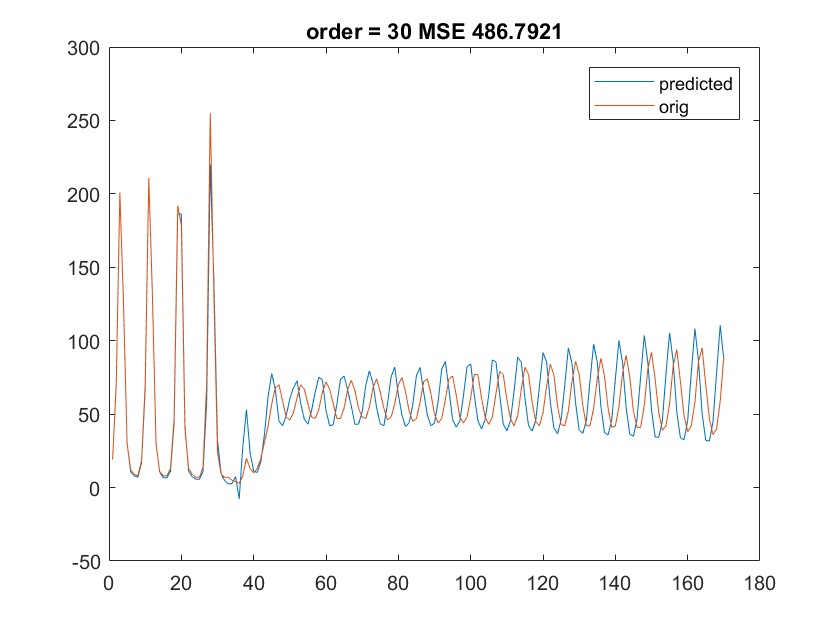


Figure 29: Results of prediction for order = 30 and MSE as a function of order

Other order (70, 80) gave interesting results, but the interpretation is flawed due to the fact that for such orders, the sudden change of behavior lies then in the test sample to predict and is actually not part of the prediction.

It is not correct to use this test set in order to choose the model order and report the error on the same set. A separate validation set should be extracted from the training set and

could be used to find the optimal model order. Alternatively Bayesian ARD could be used in order to determine the order and select the most relevant inputs.