Warsaw University of Technology

FACULTY OF
ELECTRONICS AND INFORMATION TECHNOLOGY

Assignment B: approximations of functions

Mahmoud Elshekh Ali
Student index number: 323930

May 15, 2024

Course ENUME spring semester 2023/2024

Supervisor dr inż. Jakub Wagner

Contents

1 Mathematical Symbols And Notations

- N Number of samples of $f_{13}(x)$ and its approximation $\hat{f}\ (x;\mathbf{p})$.
- K Number of terms in the approximation function $\hat{f}\ (x;\mathbf{p})$.
- x Independent scalar variable (defined as $x_n \in [-x_{max}, x_{max}]$).
- $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}$
  Independent vector variable (defined as $\mathbf{x} = [-x_{max}, x_{max}]$).
- y Dependent scalar variable (defined as $f_{13}(x_n) = y_n$).
- $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}$ Dependent vector variable (defined as $\mathbf{y} = \begin{bmatrix} f_{13}(x_1) \\ f_{13}(x_2) \\ f_{13}(x_3) \\ \vdots \\ f_{13}(x_N) \end{bmatrix}$).
- $\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_K \end{bmatrix}$ vector of approximation function parameters.

2 Introduction

- The purpose of this experiment, of which this document is the result, is the application and practice of methods of approximating functions on the basis of a set of their discrete values. There are many such methods, and there is a large scope and variety to each of them. However, of particular interest to this experiment is the "least-squares" method. Assume that we have a set of points representing some unknown function $f(x)$, the least-squares method of approximation is the process of finding some function $\widehat{f}\ (x;\mathbf{p})$, which best approximates the function $f(x)$ based on the least-squares approximation criterion **[1]**:

$$J_2(\mathbf{p}) = \Sigma_{n=1}^{N}(\widehat{f}\ (x;p_0,p_1,\ldots,p_K) - f(x))^2 \tag{1}$$

By best approximates $f(x)$, we mean that it minimizes the criterion. The function $\widehat{f}\ (x;\mathbf{p})$ is a sum of linearly independent functions, multiplied by a set of parameters, i.e., $\widehat{f}\ (x;\mathbf{p}) = \Sigma_{k=1}^{K}p_k\phi_k(x)$. The functions $\phi_k(x)$ can take on many forms. For example, they may be algebraic polynomials, trigonemetric polynomials, rational functions, etc. Any one of those may be used to approximate an unknown function, however some may prove more or less useful depending on the application. When trying to find a function which best approximates another, using the least-squares method, and given a set of discrete values of the approximated function, the next step is to find the vector of parameters $\mathbf{p} = \{p_0,p_1,\ldots,p_K\} \subset \mathbb{R}$ which minimizes the least-squares approximation criterion. This is done analytically by finding the values of $\mathbf{p}$ for which $\frac{\delta J_2(\mathbf{p})}{\delta p_k} = 0$, or, written in vector form **[1]**:

$$\mathbf{\Phi}^T \cdot \mathbf{\Phi} \cdot \mathbf{p} = \mathbf{\Phi}^T \cdot \mathbf{y} \Rightarrow \mathbf{p} = (\mathbf{\Phi}^T \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{y} \tag{2}$$

Where: $\mathbf{\Phi} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) & \cdots & \phi_K(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \phi_3(x_2) & \cdots & \phi_K(x_2) \\ \phi_1(x_3) & \phi_2(x_3) & \phi_3(x_3) & \cdots & \phi_K(x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \phi_3(x_N) & \cdots & \phi_K(x_N) \end{bmatrix}$, and $\mathbf{y} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_N) \end{bmatrix}$

Having such a vector $\mathbf{p}$ satisfying equation (2), and having chosen the form of the approximation functions, we finally have the desired approximation function $\widehat{f}\ (x;\mathbf{p}) = \Sigma_{k=1}^{K}p_k\phi_k(x)$.

3 Methodology and results of experimentation

3.1 Methodology, assumptions, limitations, etc.

- Some important notes regarding the least-squares method are that, for example, the least-squares approximation, and this extends to some degree for other approximation methods as well, is good for approximating functions which do not experience rapid changes (functions whose derivatives are continuous everywhere). More generally, the least-sqaures method is good for approximating functions whose behaviour is relatively simple. Also, the least-squares method can be said to be somewhat ill conditioned **[1]**, as for small changes in the data, relatively large changes may be observed in the results of the approximation. This makes the least-squares method sensitive to data points that are outliers, as they can skew the approximation. Furthermore, often times in practical settings the set of discrete values representing the approximated function has within it some error that results from the imperfections of measurement instruments. This error is referred to as measurement error, and, coupled with the sensitivity mentioned previously, can in some cases lead to innacurate approximations.

- The experiment is performed, numerically, using the *MATLAB* environment. The detailed description of the implementation of the experiment in *MATLAB* is provided in a later section, the full implementation is provided in the appendix

3.2 Goals of the experiment

- The aim of this experiment is to find the approximation function, using the least-squares method and trigonemetric polynomials, approximating an unknown function $f_{13}(x)$. Both functions, $f_{13}(x)$ and $\hat{f}\ (x; \mathbf{p})$, are evaluated over the domain $x = [-x_{max}, x_{max}]$, using samples. Thus the approximation function is of the form:

$$\hat{f}\ (x; \mathbf{p}) = \hat{f}\ (x; a_0, a_1, \ldots, a_K, b_1, b_2, \ldots, b_K) = \\ \Sigma_{k=0}^{K} a_k cos(kx) + \Sigma_{k=1}^{K} b_k sin(kx) = a_0 + \Sigma_{k=1}^{K} a_k cos(kx) + b_k sin(kx) \quad (3)$$

The experiment is split into 4 tasks, some of which are split into a collection of subtasks. The tasks are as follows:

1. Develop a program for estimating the set of parameters **p** which minimize the least-square approximation criterion. Assume the samples of

the approximated function used to find the approximation function are exact (there is no measurement error).

- Test the developed program for a range of different values for x_{max} and for K (the number of the terms in the approximation function, see equation (3)).

2. Determine the dependence of the mean-square approximation error on the number K. The following is used to calculate the mean-square approximation error:

$$e = \sqrt{\frac{1}{N}\Sigma_{n=1}^{N}(\hat{f}\ (x; p_0, p_1, \ldots, p_K) - f(x))^2} \tag{4}$$

   Assume the samples of the approximated function used to find the approximation function are exact (there is no measurement error).

   - Observe how the dependence changes for different values of x_{max}.

3. Determine the dependence of the expected value of the mean-square approximation error $\bar{e}$ on x_{max} and on the variance of the measurement error σ^2. Assume that the samples of the approximated function are corrupted with some error $\Delta\widetilde{y}_n$, which is normally distributed. The error is modelled as follows:

$$\widetilde{y}_n = f_{13}(x) + \Delta\widetilde{y}_n \quad \text{for} \quad n = 1, 2, \ldots, N \tag{5}$$

   - For more reliable results, repeat the experiment many times, each time with a different set of randomly generated error. Calculate the expected value using the following equation:

$$\bar{e} \ = \sqrt{\frac{1}{R}\Sigma_{r=1}^{R} e_r^2} \tag{6}$$

4. Compare three methods of calculating the vector $\mathbf{p}$ using *MATLAB*'s built-in operator \. The methods are:

$$\mathbf{p} = pinv(\mathbf{\Phi}) \cdot \mathbf{y} \tag{7}$$

$$\mathbf{p} = \mathbf{\Phi} \backslash \mathbf{y}; \tag{8}$$

$$\mathbf{p} = (\mathbf{\Phi'} \cdot \mathbf{\Phi}) \backslash (\mathbf{\Phi'} \cdot \mathbf{y}); \tag{9}$$

3.3 Experimentation process for numeric results

This section will detail the process by which the numerical results were obtained, and the experiment was performed. The steps are as follows:

- Task 1:

  1. The function $f_{13}(x)$ is provided in a file seperate from the implementation of the experiment. The function is provided in a file format that does not allow for the viewing of its contents.
  2. The numerical implementation of experiment begins with setting the values for the constants used in the experiment, those being N and K, x_{max} (see page 3 for details).
  3. Vectors of data are generated. The vector is of the form $x = [-x_{max}, x_{max}]$, and has N elements. The vector is generated using *MATLAB*'s built-in functions. The elements of the vector are equidistant. The vector is stored in the vector variable **x**.
  4. The function $f_{13}(x)$ is applied to the elements of the data vector. The results are stored in the vector variable **y**.
  5. The matrix $\mathbf{\Phi}$, as defined in the introduction, is generated using a user-defined function, defined in a file seperate from the implementation of the experiment. The matrix variable **M** stores the matrix generated.
  6. The vector of parameters **p**, which minimizes the least-squares approximation criterion (see equation (1)), calculated using equation (2), is generated. The results are stored in the vector variable **p**.
  7. Using the obtained results for the vector **p** and the data vector, samples of the approximation function $\hat{f}(x; \mathbf{p})$ are generated using equation (3). The results are stored in the vector variable **y_approx**.
  8. Finally, figures 1,2 and 3 are generated using *MATLAB*'s extensive plotting capabilities. The least-squares approximation criterion is also calculated. The result is stored in the scalar variable j.

- Task 2:

  1. Calculation of vector of samples of the mean-square approximation error, each element being calculated for a different value of K. The results are stored in the vector variable **E**. This is repeated for each method.

2. Generation of figures 4.

- Task 3:

  1. Generation of vectors of sample values for x_{max} and for the variance of the measurement error σ^2 for use in showing the dependace of the expected value of the mean-square approximation error on these values. The vectors are stored in the vector variables **X_max** and **sig2** respectively. The vector **X_max** ranges from $[10^{-2}, 10]$, the vector **sig2** ranges from $[10^{-12}, 10^{-2}]$.
  2. Generation of matrix of values of the expected value of the mean-square approximation error $\bar{e}$. Each entry of the matrix is calculated for a unique combination of x_{max} and σ^2, $x_{max,j}$ and σ_i^2 respectively, where $i, j = 1, 2, 3, \ldots, N$. Each entry of the matrix is calculated as the expected value of 100 individual mean-square approximation errors, each calculated for a different set of measurement error, using the same values of x_{max} and σ^2, using equation (6). Each individual mean-square approximation error is calculated after introducing error according to the model shown in equation (5), and using equation (4); these are calculated for the data vector x. The resulting matrix is stored in the matrix variable **T**. The matrix calculated takes the following form: **T** $= \begin{bmatrix} \bar{e}_{1,1} & \bar{e}_{1,2} & \bar{e}_{1,3} & \cdots & \bar{e}_{1,N} \\ \bar{e}_{2,1} & \bar{e}_{2,2} & \bar{e}_{2,3} & \cdots & \bar{e}_{2,N} \\ \bar{e}_{3,1} & \bar{e}_{3,2} & \bar{e}_{3,3} & \cdots & \bar{e}_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{e}_{1,N} & \bar{e}_{N,2} & \bar{e}_{1,2} & \cdots & \bar{e}_{N,N} \end{bmatrix}$

     where $\bar{e}_{i,j}$ is the expected value of the mean-square approximation error calculated using $x_{max,j}$ and σ_i^2.
  3. Generation of figure 5.

- Task 4:

  1. Generation of vector of parameters **p**. The vector is generated three times, each time using one of the three methods mentioned in equations (7),(8) and (9). The results of the method using equation (7) is stored in the vector variable **p1**. The results of the method using equation (8) is stored in the vector variable **p2**. The results of the method using equation (9) is stored in the vector variable **p3**.

2. During the generation of the vector variables **p1**, **p2**, **p3** a timer is set to measure the rutime required for the calculations. A seperate timer is set for each method. This measurement procedure is repeated a number of times decided by a scalar variable t. The results of the measurements are stored in the vector variables **m1**, **m2**, **m3**. The measurements of the runtime of the first method are stored in **m1**, and the second method measurement results in *m2*, the third method measurement results are stored in **m3**.
3. Generation of figures 6,7.

3.4 Results of experimentation

3.4.1 Obtained approximation function and parameters

Figures 1.a - 1.c, 2.a - 2.c and 3.a - 3.c show the developed samples of the approximation functions (developed using the calculated set of parameters **p** which minimizes the least-squares criterion) as well as the set of samples of the approximated function $f_{13}(x)$.

3.4.2 Obtained dependences of the mean-square approximation error on number of terms in the approximation function

Figures 4.a - 4.c show the dependences calculated after calculating the set of parameters **p** which minimizes the least-squares criterion, which is in turn used in calculating the approximation function and then the mean-square error, using three different methods:

Method 1:
Using *MATLAB*'s built-in function *pinv(**A**)*: $\mathbf{p} = pinv(\mathbf{\Phi}) \cdot \mathbf{y}$
Method 2,3: Using *MATLAB*'s built-in operator \:
Method 2: $\mathbf{p} = \mathbf{\Phi} \backslash \mathbf{y}$
Method 3: $\mathbf{p} = (\mathbf{\Phi}^T \cdot \mathbf{\Phi}) \backslash (\mathbf{\Phi}^T \cdot \mathbf{y})$

3.4.3 Obtained dependence of the expected value of the mean-square approximation error on the limit of the data vector and the variance of the measurement errors

Figure 5 shows the dependence of the expected value of the mean-square approximation error E_e on the limit of the data vector x_{max} and the variance of the measurement errors σ^2.

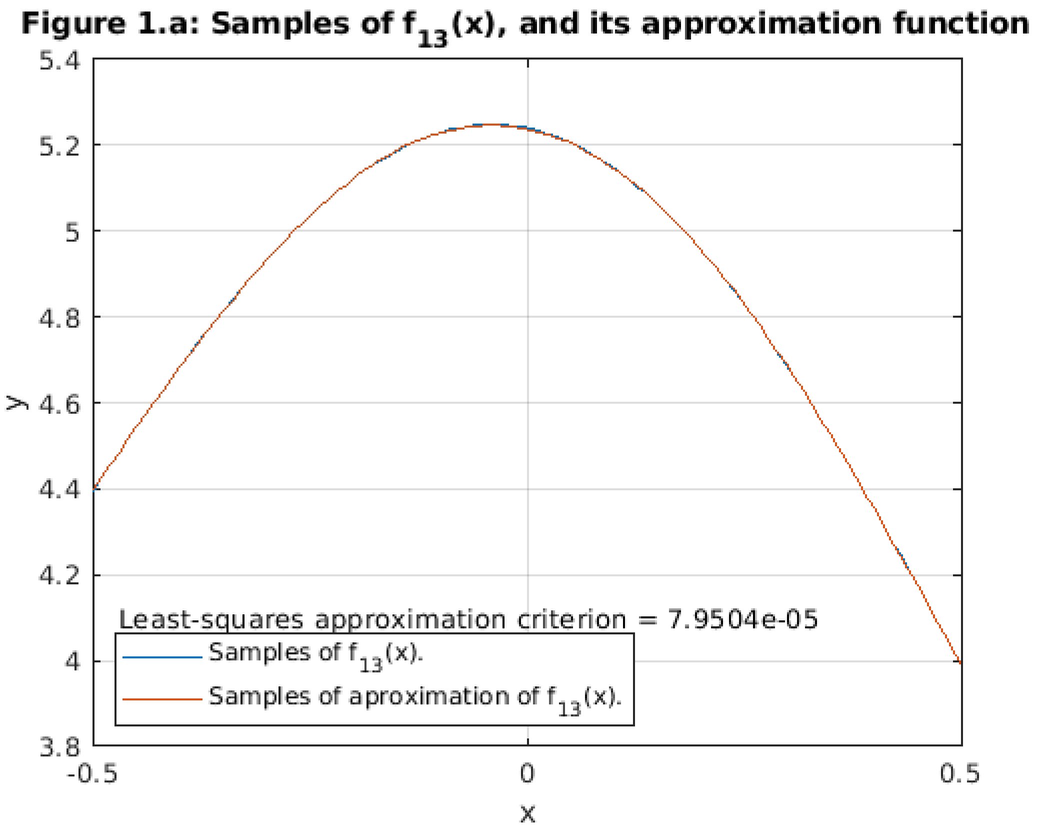

Figure 1.a: Plot of the approximated function $f_{13}(x)$ and its approximation $\widehat{f}\ (x; \mathbf{p})$. K is set to 2, x_{max} is set to 0.5 .

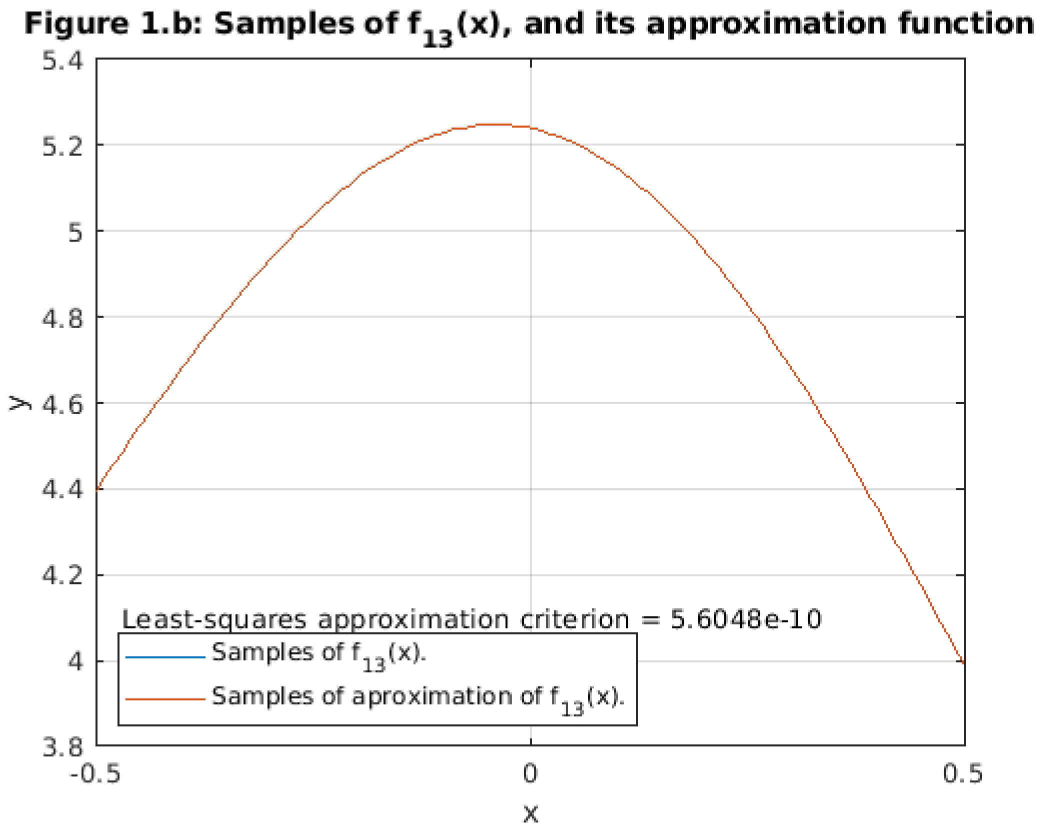

Figure 1.b: Plot of the approximated function $f_{13}(x)$ and its approximation $\widehat{f}\ (x;\mathbf{p})$. K is set to 4, x_{max} is set to 0.5 .

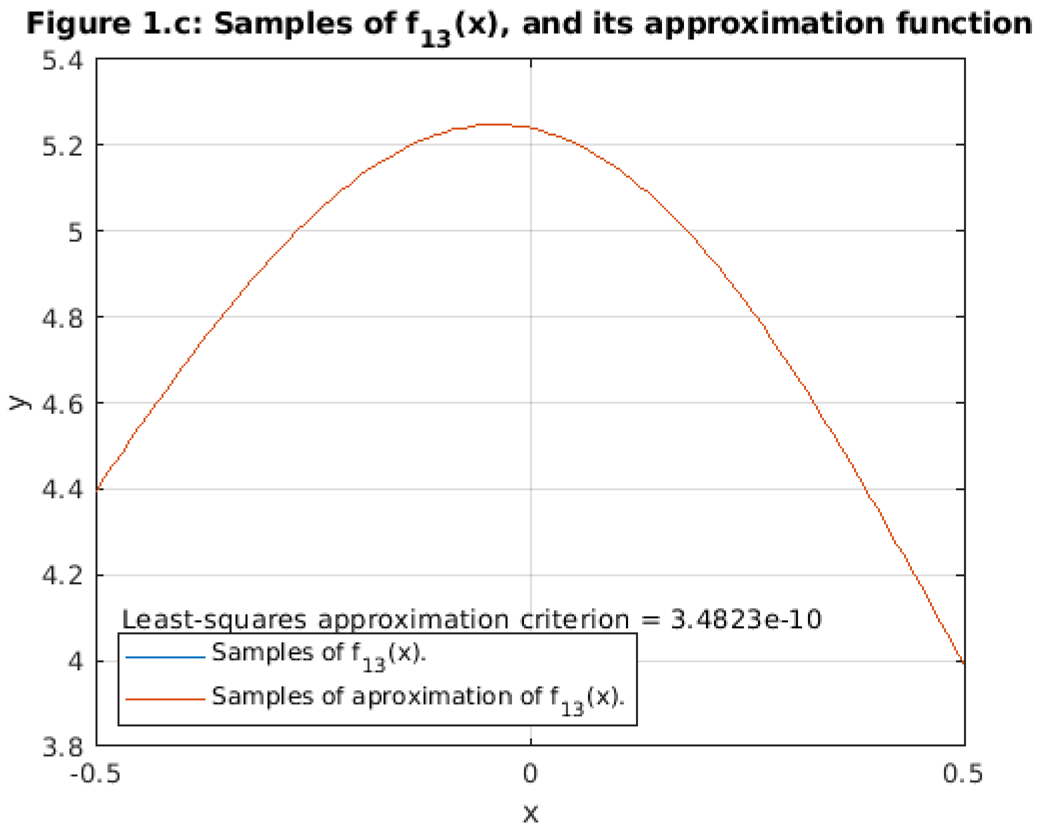

Figure 1.c: Plot of the approximated function $f_{13}(x)$ and its approximation $\hat{f}\ (x; \mathbf{p})$. K is set to 6, x_{max} is set to 0.5 .

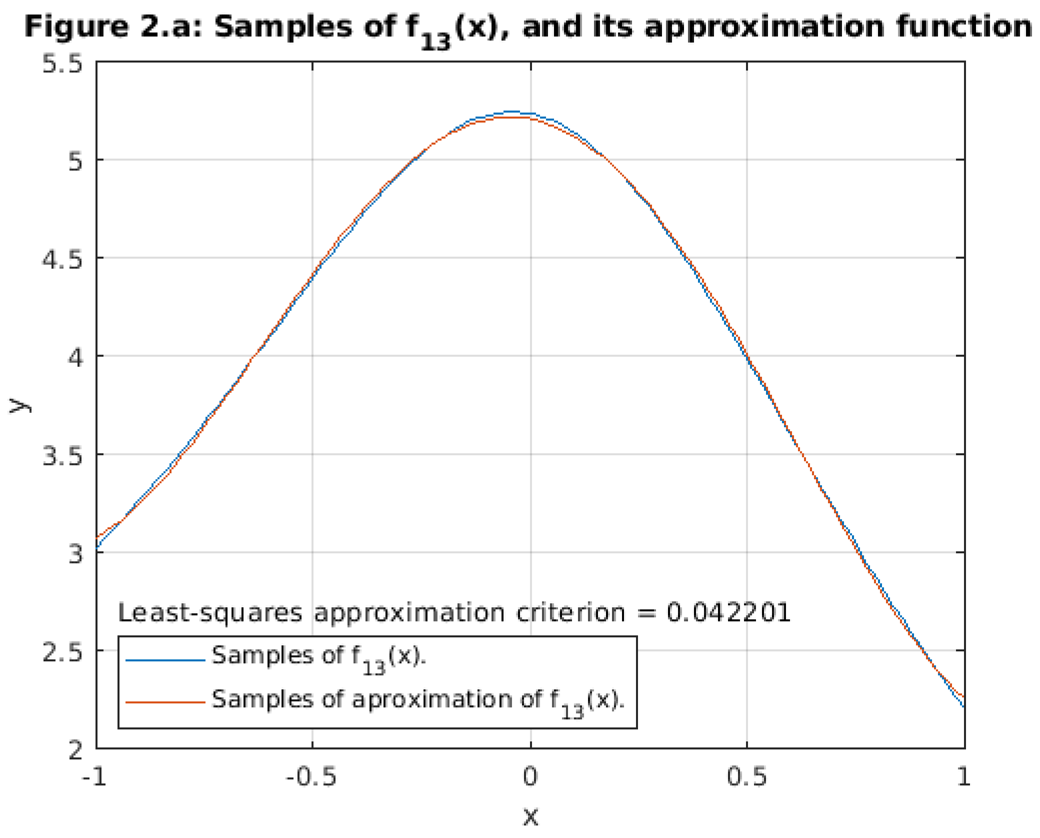

Figure 2.a: Plot of the approximated function $f_{13}(x)$ and its approximation $\hat{f}\ (x; \mathbf{p})$. K is set to 2, x_{max} is set to 1 .

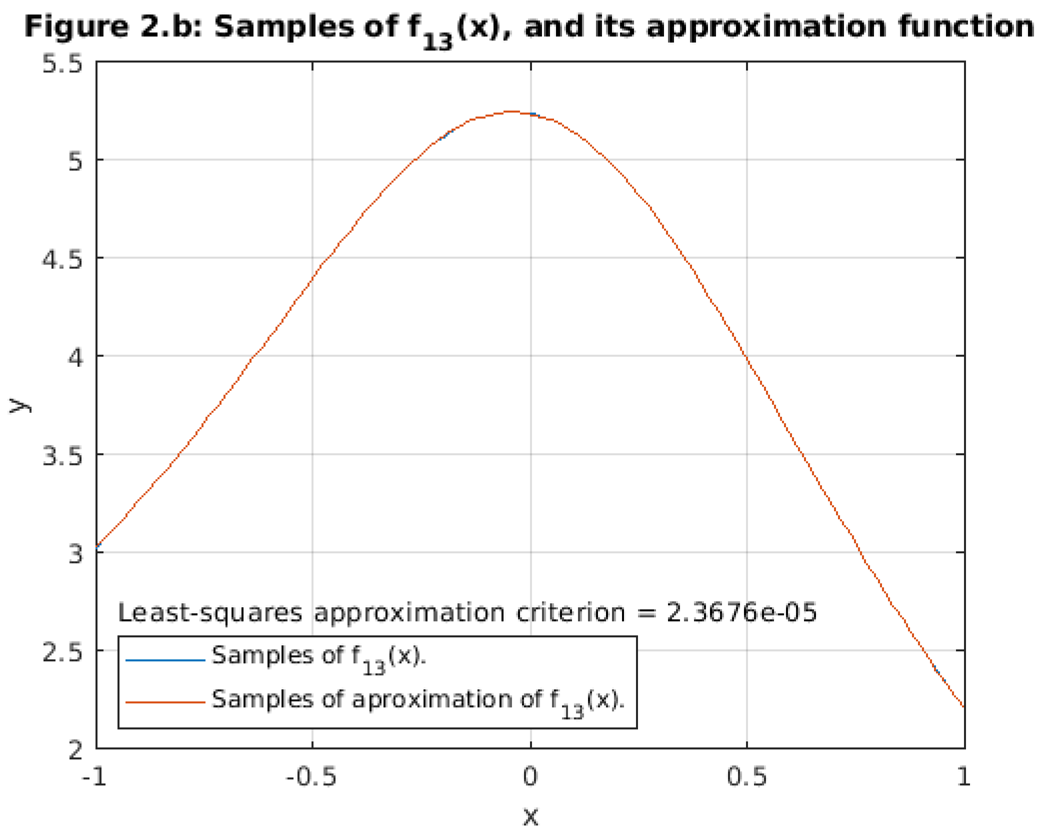

Figure 2.b: Plot of the approximated function $f_{13}(x)$ and its approximation $\hat{f}\,(x;\mathbf{p})$. K is set to 4, x_{max} is set to 1 .

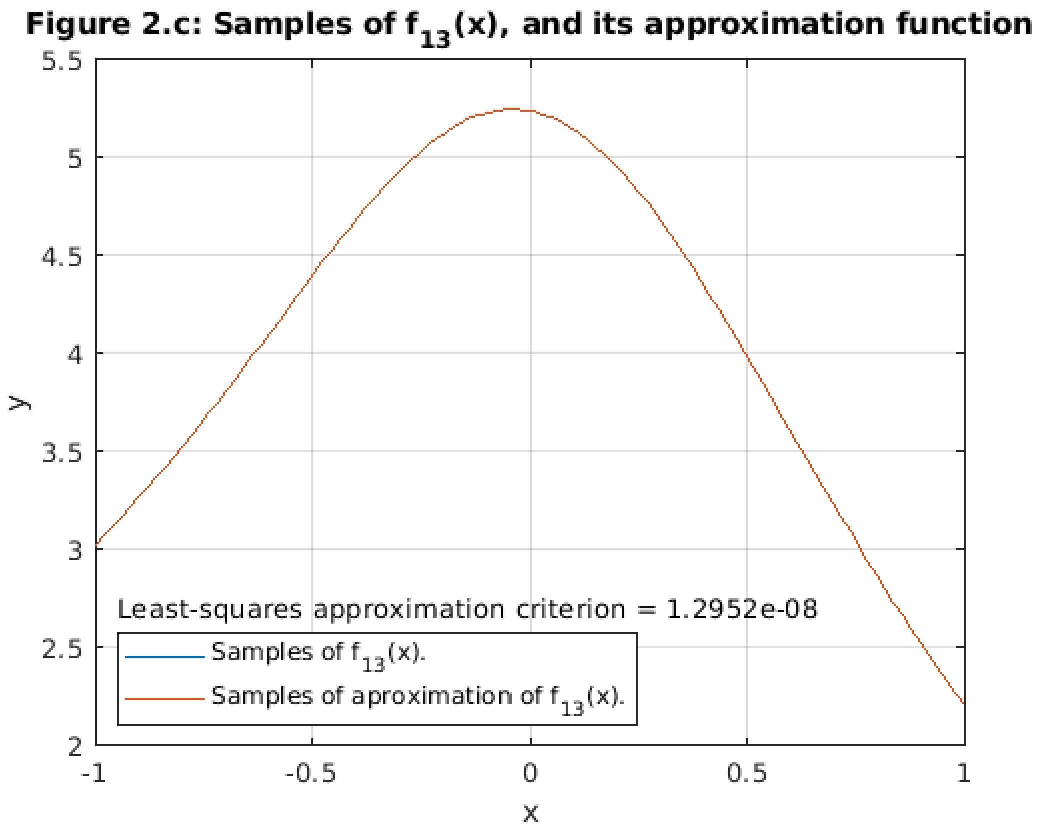

Figure 2.c: Plot of the approximated function $f_{13}(x)$ and its approximation $\hat{f}\,(x;\mathbf{p})$. K is set to 6, x_{max} is set to 1 .

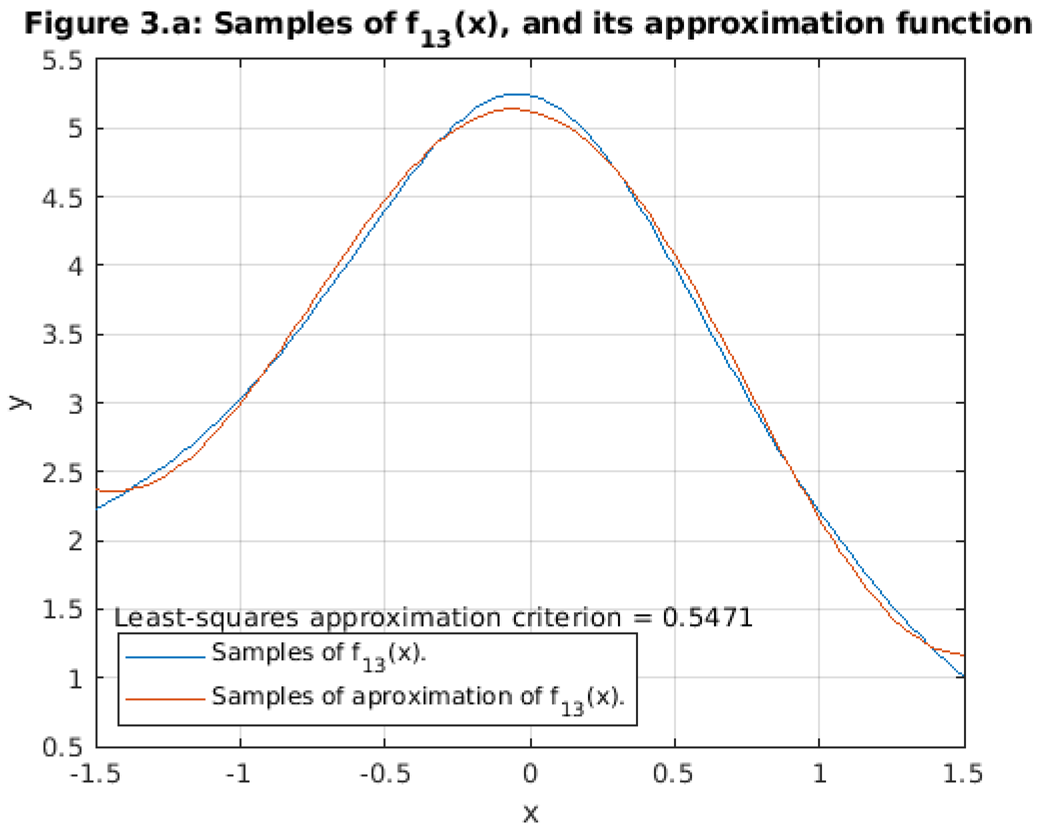

Figure 3.a: Plot of the approximated function $f_{13}(x)$ and its approximation $\widehat{f}\,(x;\mathbf{p})$. K is set to 2, x_{max} is set to 1.5 .

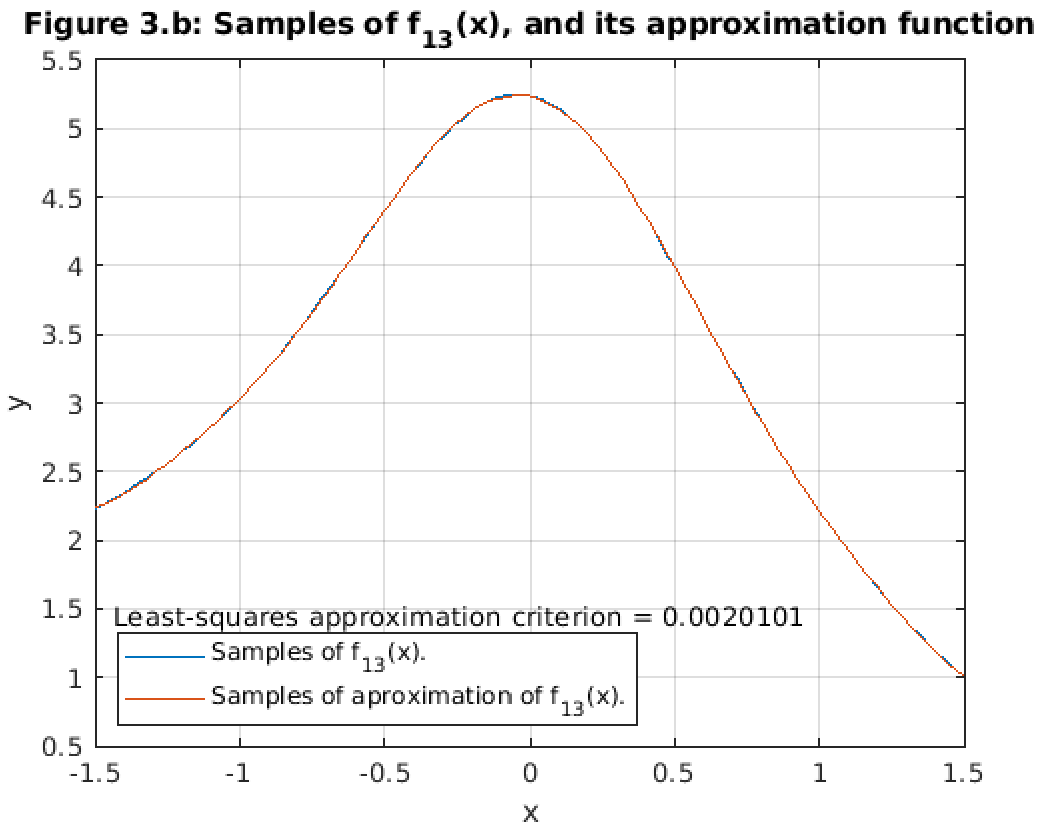

Figure 3.b: Plot of the approximated function $f_{13}(x)$ and its approximation $\widehat{f}\ (x; \mathbf{p})$. K is set to 4, x_{max} is set to 1.5 .

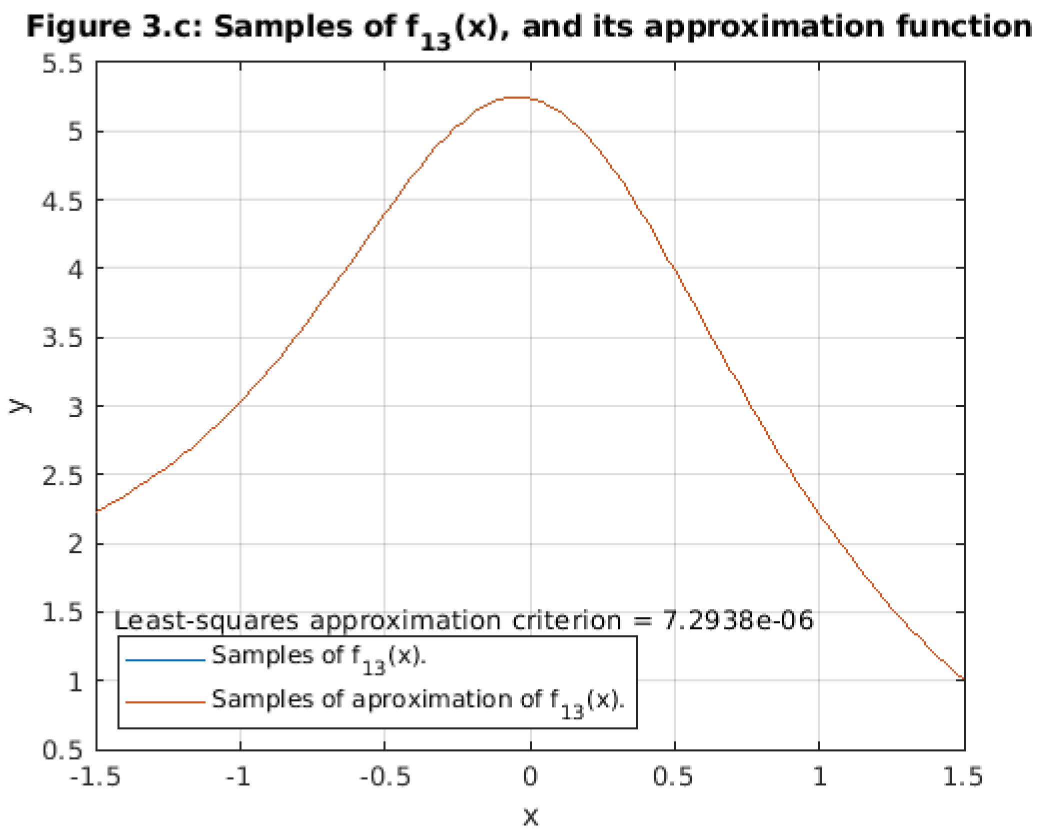

Figure 3.c: Plot of the approximated function $f_{13}(x)$ and its approximation $\widehat{f}\ (x; \mathbf{p})$. K is set to 6, x_{max} is set to 1.5 .

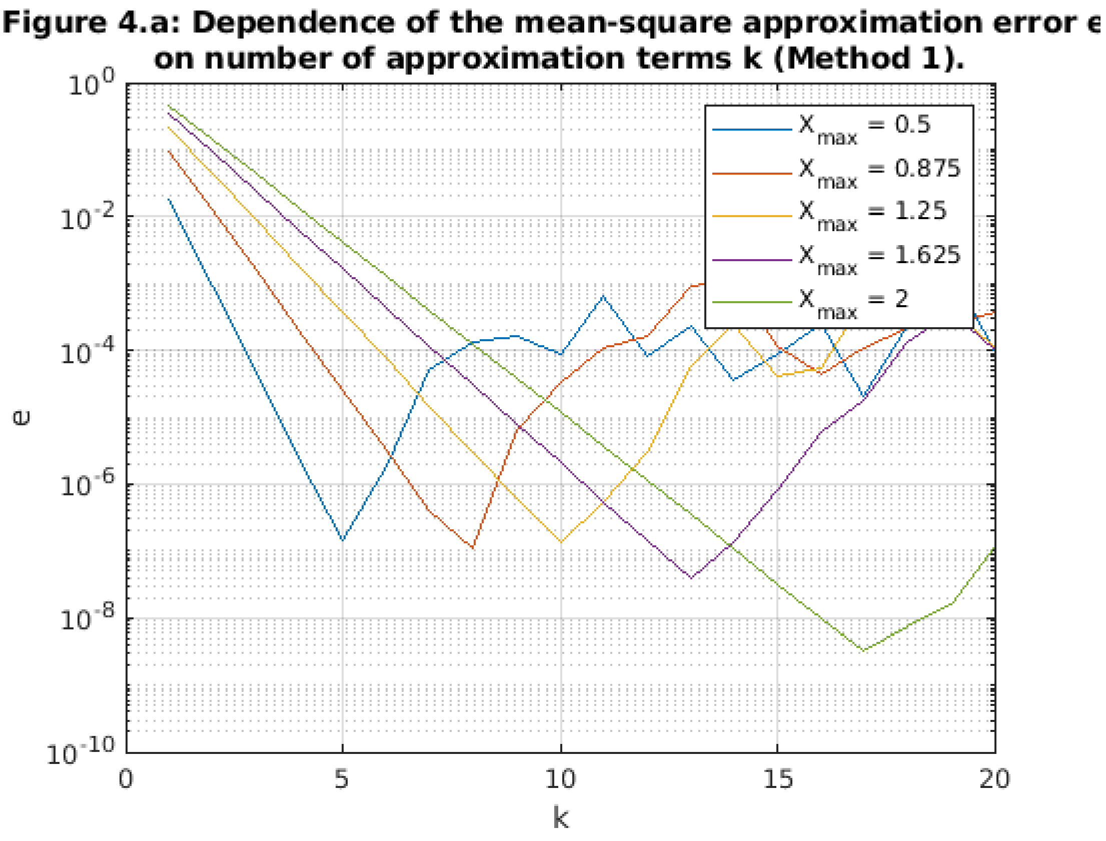

Figure 4.a: Dependence of the mean-square approximation error, Calculated using an approximation function utilizing the set of paramters **p** calculated using method 1.

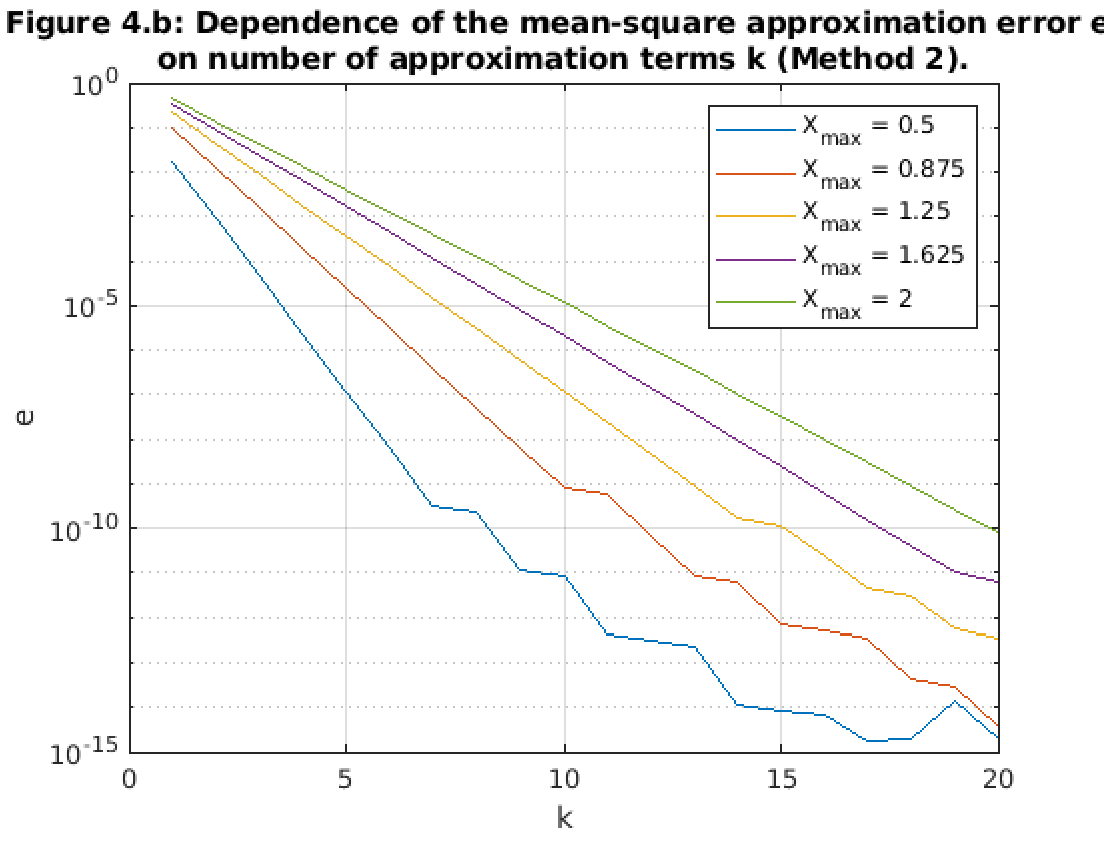

Figure 4.b: Dependence of the mean-square approximation error, Calculated using an approximation function utilizing the set of paramters **p** calculated using method 2.

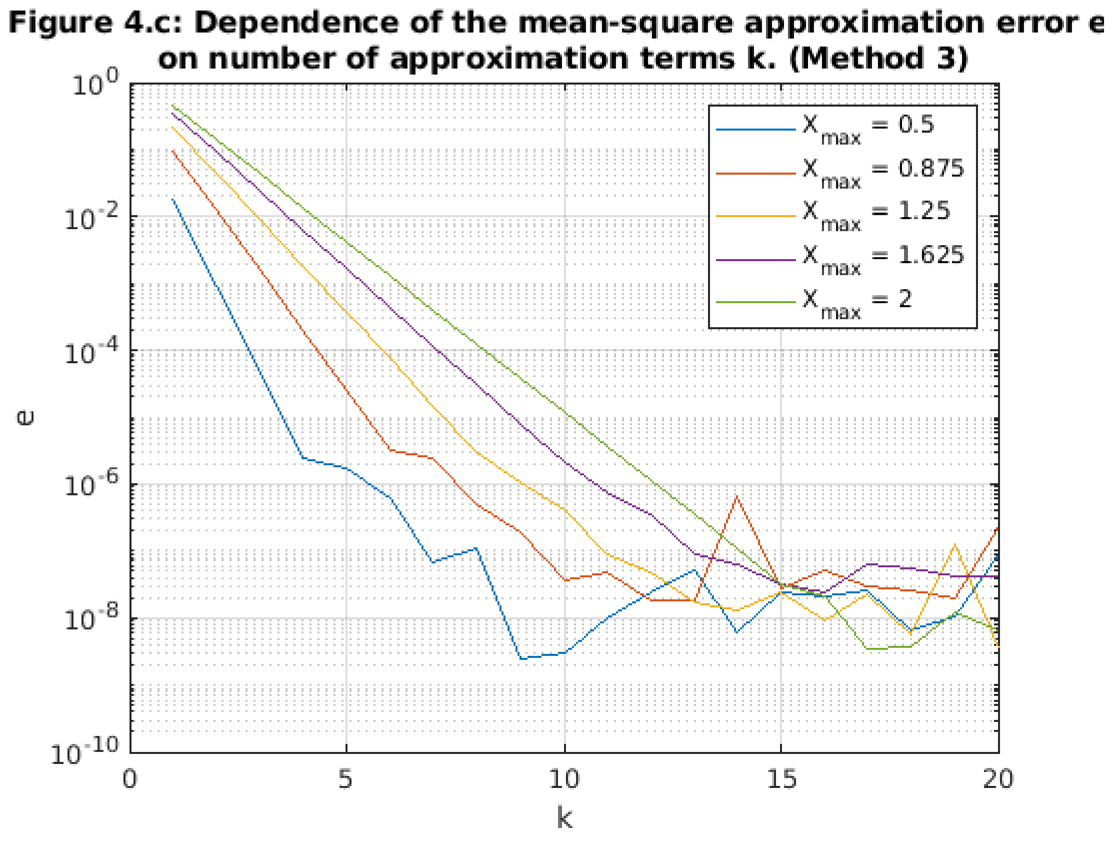

Figure 4.c: Dependence of the mean-square approximation error, Calculated using an approximation function utilizing the set of paramters **p** calculated using method 3.

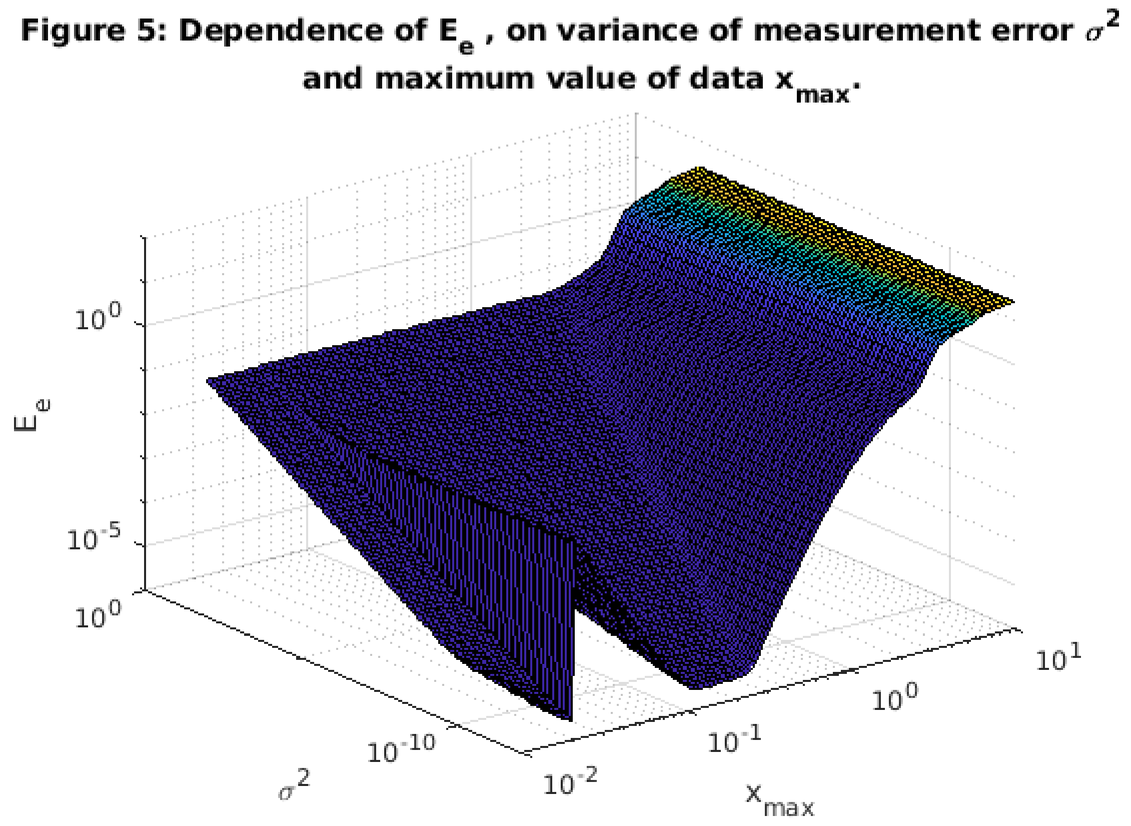

Figure 5: Dependence of the expected value of the mean-square approximation error on the limits of the data vector and the variance of the measurement error.

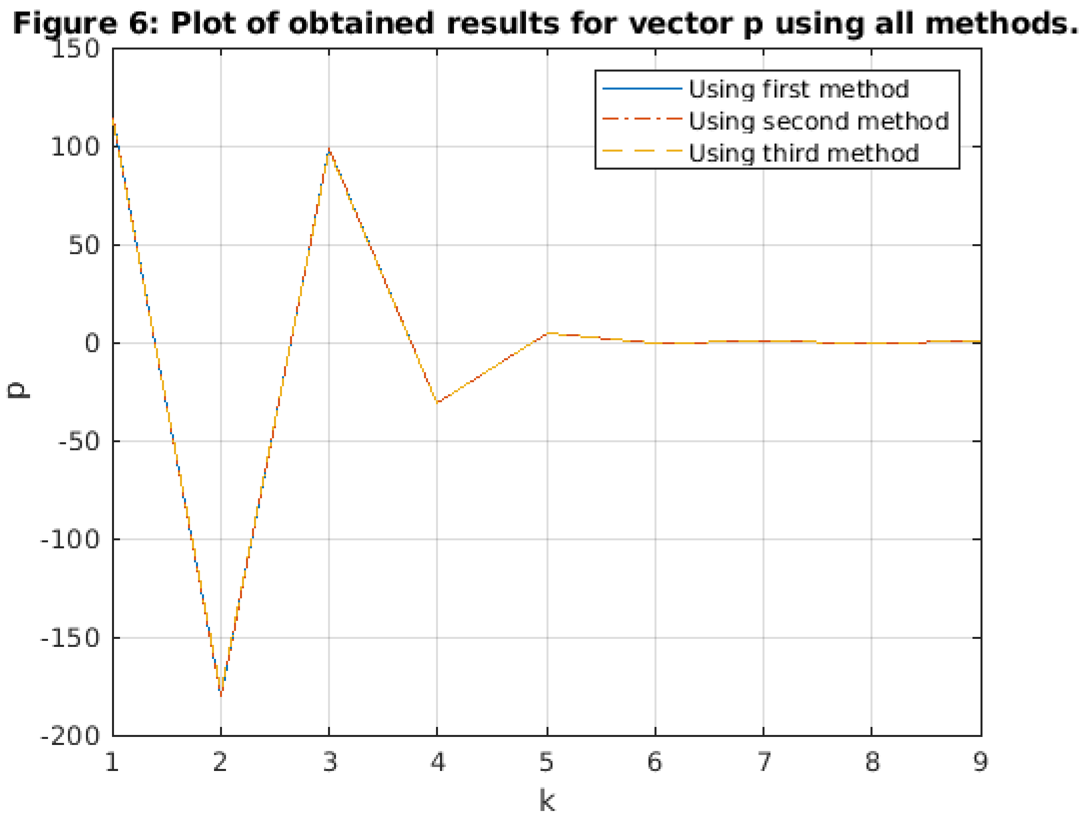

Figure 6: Results of all methods used in the calculation of the vector of approximation function parameters **p**.

3.4.4 Comparison of different methods of obtaining vector of parameters p

Figure 6 shows a comparison of the results of all methods. Figure 7 shows the results of comparing the runtime of 100 trials of calculation of vector **p** using all methods.

4 Discussion

A discussion of obtained results is provided in this section.

- Beginning this discussion with a discussion of obtained results for task 1 (figures 1.a - 1.c, 2.a - 2.b, 3.a - 3.c), it can be seen from the figures that the approximation function utilizing the calculated parameters does indeed closely approximate the function $f_{13}(x)$. The most prominent feature worth

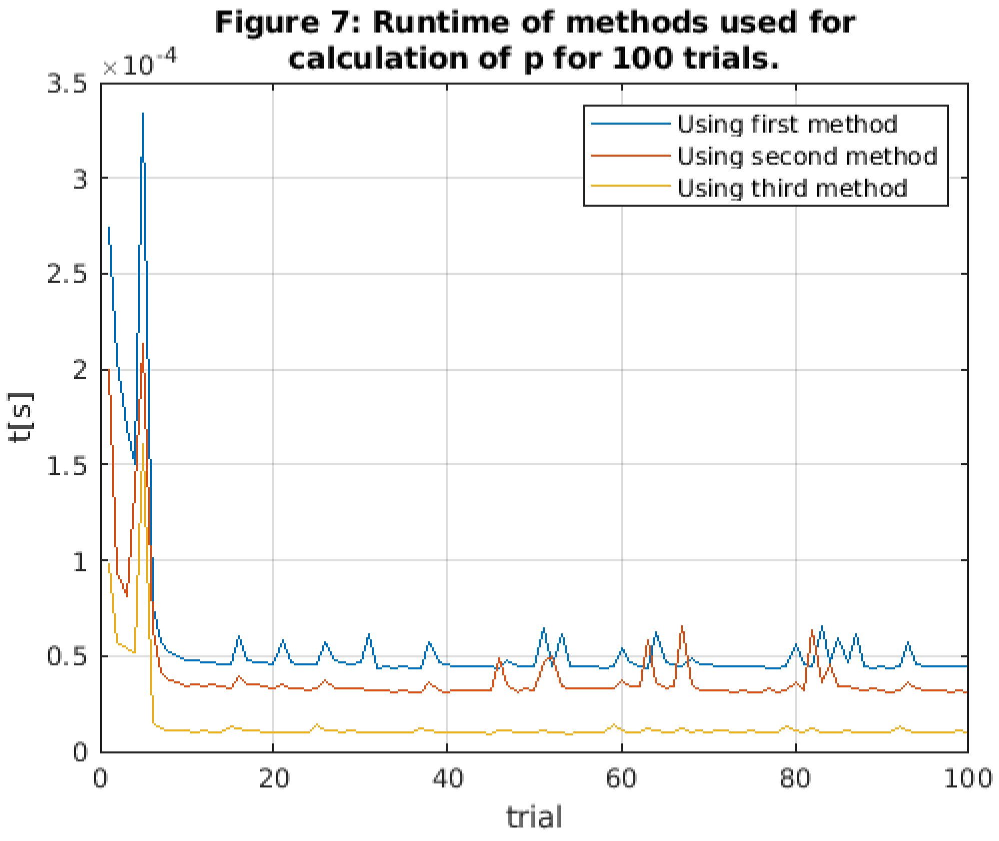

Figure 7: Results of comparison of runtime for all methods using measurements obtained across 100 trials.

addressing is that across the obtained figures, a common pattern is that the error of the approximation (or alternatively, the least-squares criterion) is reduced with increasing K, and increased with increasing x_{max}. This is further confirmed in the later figures such as figures 4.a - 4.c and figure 5 as well. There are a few reasons for this:
First of all, the number K represents the number of terms in the approximation function, which means that for higher values of K, the approximation function becomes more "complex". In other words the approximation function becomes better able to adapt to shape of the approximated function. This is especially true when the approximated function itself is complex or exhibits unpredictable behaviour. Furthermore, this added complexity means the approximation function is better able to approximate regions where the approximated functions changes quickly for a short range of data.
Secondly, from what has already been stated, we can deduce that when x_{max} is increased (and with it the number of the approximated samples, at least in the case of this experiment), more of the approximated function is subject to approximation, which could mean more complex behaviour of the function which makes it more difficult to approximate (this is not necessarily the case, for some functions, such as those with no changes or periodic changes, increasing x_{max} will likely not lead to an increase in error).

- Continuing this discussion with a discussion of the obtained results for task 2 (figures 4.a - 4.c), the same pattern previously mentioned can be seen not only for the least-squares criterion in task 1, but also for the mean square error regardless of the method (although this is true only for a range of values of K, which differs from method to method, and differs depending on x_{max}). However in all figures, the mean square error seems to lose its stability for relatively large values of K, with method 2 providing the most stability for this error, and method 1 the least. To explain this, and other similiar observations, a short overiew of the used methods is needed:
Method 1 uses the Moore-Penrose psuedoinverse matrix, which is implemented through an algorithm which uses the singular decomposition of the input arguement. The psuedoinverse produced using this function does not have all the properties of the inverse, however the properties it does share with the inverse make it suficient for this experiment. The benefit of this matrix is that it always exists, as opposed to the inverse **[2]**.
Methods 2,3 employ the operator \, also known in the *MATLAB* environment as *mldivide* **[3]**. This operator, when used as follows: $x = A\backslash B$ returns the solution to the system of linear equations $A \cdot x = B$, assuming the solution exists. This operator employs a sophisticated algorithm which exploits

symmetries in the problem by dispatching to an appropriate solver.
Further more detailed explanations are provided on the associated *MATLAB* help pages **[2]**,**[3]**.

- Observing the obtained figures, and using the provided overview of the methods, we may directly compare these methods and attempt to explain them. As explained before, there is some discrepancy in the behaviour of the mean-square error between methods for different values of K. A simple explanation for this maybe the difference in the algorithms used. As previously mentioned, the algorithm employed in the operator / is capable of using different solvers depending on the type of the matrix, which method 1 does not employ. Another thing of note is the fact that for methods 2 and 3, the result for the vector $\mathbf{p}$ is computed directly, while for method 1 the psuedoinverse matrix is first returned by the function *pinv(**A**)*, which is then multiplied by $\mathbf{y}$ to produce the vector $\mathbf{p}$. Also, method 2 requires more computation due to the sizes of the matrices used generally being larger than in method 3. These differences can also explain the differences seen in the results of the runtime required for each method. Comparison of the runtime required for each method shows that method 3 is the most efficient, followed by method 2 and finally method 1. It is important to mention that in the numerical implementation of this experiment, for relatively large values of K, method 3 became highly unstable, and returned innacurate results, while the other methods remained stable.

- Finally,a short discussion of the results of task 3 (figure 5). It can be seen from figure 5 that the expected value of the mean-square approximation error can be significantly affected by increasing the limit of the data vector. The expected value of error generally increases when this limit increases. This can be explained using the same observations determined in the first paragraph of this discussion. Furthermore, the variance of the measurement error has a similiar effect on the expected value of the error, however its influence decreases as the limit of the data vector increases. These observations are explained by the fact that increasing both the variance of the limit of the data vector and the variance of the measurement error has the effect of making the measurement error larger relative to the actual value of the approximated function, especially for relatively small values of the measured function, which in turn leads to a higher mean-square error in the approximation.

5 Conclusion

- The least-squares method is a good method for approximating functions which exhibit predictable or simple behaviour.
- In general, the more complicated the behaviour of the approximated function, the more terms requried in the approximation function.
- The following is a summary of the comparison of the methods used in this experiment:
  Method 1: Most stable, but is in general the least efficient.
  Method 2: The stability and runtime of this method is somewhere between that of the other methods, it is a good balance of the advantages of the other methods.
  Method 3: Least stable, but is the most efficient for the values for which it is.
- The mean-square approximation error is inversely proportional to the number of approximation K, and proportional to the limit of the data vector x_{max}.
- The expected value of the mean-square approximation error is proportional to the limit of the data vector x_{max} and the variance of the measurement error, however for relatively large values of the limit of the data vector the influence of the variance is significantly decreased.

6 References

[1] R. Z. Morawski, lecture notes for the course *Numerical Methods*, Warsaw University of Technology, Faculty of Electronics and Information Technology, spring semester 2023/24.

[2] `https://www.mathworks.com/help/matlab/ref/pinv.html#d126e1210505`
[accessed in 14/05/2024]

[3] `https://www.mathworks.com/help/matlab/ref/mldivide.html`
[accessed in 14/05/2024]

7 Appendix

7.1 *MATLAB* implementation of assignment B (Assingment_B.mlx):

```
% Start of MATLAB implementation for assignment B.
% Clear any remaining variables from previous script
    runs.
clear

% TASK 1 *******************************************
disp("Beggining of task 1.")
% Setting initial values for the experiment. The
   values are:
% - N, the number of samples used to represent both
   the approximated functon f13(x),
% and its least-squares approximation using
   trigonemetric polynomials.
% - x_max, the limits of the interval in which all
   the data points used in
% task 1 are.
% - k, the size of the vector of parameters used in
   the approximation
% function. Can also be understood as the number of
   approximation terms
% used in the calculation of each sample of the
   approximation function.
N = 100
x_max = 2
k = 2

% Generating the data vector used in the experiment.
x = linspace(-x_max, x_max, N);

% Generating vectors of the results of the
   approximated function f13(x).
y = f13(x);

% Generating the matrix Phi as described in the
   document, which is used to
```

```matlab
% find the vector of parameters p which minimizes the criterion of the least-squares
% approximation of f13(x). The function used to generate these matrices,
% Phi(x,k), is defined in the file Phi.m .
M = Phi(x,k);

% Generating the vector of parameters which minimizes the criterion of the least-squares
% approximation of f13(x). The function handles P1,P2,P3 are defined for this
% purpose. The function pinv(A) returns the Moore-Penrose inverse of the
% matrix A ([A*A']^(-1)*A').
P1 = @(M,Y) pinv(M)*Y';
P2 = @(M,Y) M \ Y';
P3 = @(M,Y) (M'*M)\(M'*Y');

p1 = P1(M,y);

% Generating the vectors for the results of the approximation function
% f_approx(x,p), which approximates the function f13(x).
y_approx1 = f_approx(x,p1);

% Computing the criterion of the least-squares approximation of f13(x). The
% function handle J(x,p) is defined for this purpose.
J = @(X,P) sum((f_approx(X,P) - f13(X)).^2);
j1 = J(x,p1)

% Code for generation of figures 1.
txt = "Least-squares approximation criterion = " + j1;
figure(1)
plot(x,y);
hold on;grid on;
plot(x,y_approx1);
```

```
title("Figure 1.a: Samples of f_{13}(x), and its
   approximation function")
legend("Samples of f_{13}(x).","Samples of
   aproximation of f_{13}(x).",'Location','southwest
   ')
xlabel("x")
ylabel("y")
text(-0.95,2.7,txt)
hold off;



% TASK 2 *******************************************
disp("Beggining of task 2.")
N =100
k = 4

% Generation of vector of mean-square approximation
   errors. The function
% handle e(x,y,k,N) is defined for this purpose.
e1 = @(X,Y,L,N) sqrt( J(X,P1(Phi(X,L),Y))/N );
e2 = @(X,Y,L,N) sqrt( J(X,P2(Phi(X,L),Y))/N );
e3 = @(X,Y,L,N) sqrt( J(X,P3(Phi(X,L),Y))/N );
K = 1:k;
X_max = linspace(0.5,2,5);
E = zeros(5,size(K,2));

% Code for generation of figure 2.
figure(2)
for j = 1:5
    x = linspace(-X_max(j), X_max(j), N);
    y = f13(x);
    for i = 1:size(K,2)
        E(j,i) = e1(x,y,K(i),N);
    end
    plot(K,E(j,:));
    hold on;grid on;
end
legend("X_{max} = " + X_max(1),"X_{max} = " + X_max
   (2),"X_{max} = " + X_max(3),"X_{max} = " + X_max
   (4),"X_{max} = " + X_max(5))
```

```
yscale("log")
title(["Figure 4.a: Dependence of the mean-square approximation error e,","on number of approximation terms k (Method 1)."])
xlabel("k")
ylabel("e")
hold off;

% Code for generation of figure 3.
figure(3)
for j = 1:5
    x = linspace(-X_max(j), X_max(j), N);
    y = f13(x);
    for i = 1:size(K,2)
        E(j,i) = e2(x,y,K(i),N);
    end
    plot(K,E(j,:));
    hold on;grid on;
end
legend("X_{max} = " + X_max(1),"X_{max} = " + X_max(2),"X_{max} = " + X_max(3),"X_{max} = " + X_max(4),"X_{max} = " + X_max(5))
yscale("log")
title(["Figure 4.b: Dependence of the mean-square approximation error e,","on number of approximation terms k (Method 2)."])
xlabel("k")
ylabel("e")
hold off;

% Code for generation of figure 4.
figure(4)
for j = 1:5
    x = linspace(-X_max(j), X_max(j), N);
    y = f13(x);
    for i = 1:size(K,2)
        E(j,i) = e3(x,y,K(i),N);
    end
    plot(K,E(j,:));
    hold on;grid on;
end
```

```
legend("X_{max} = " + X_max(1),"X_{max} = " + X_max(2),"X_{max} = " + X_max(3),"X_{max} = " + X_max(4),"X_{max} = " + X_max(5))
yscale("log")
title(["Figure 4.c: Dependence of the mean-square approximation error e,","on number of approximation terms k. (Method 3)"])
xlabel("k")
ylabel("e")
hold off;

% TASK 3 ******************************************
disp("Beggining of task 3.")
% R is the number of trials of error generation used in finding the
% expected value of mean-square approximation errors.
N =10
R = 10

% Generation of vector of exemplary values of maximum values of the data
% vector x, X_max (X_max ranges from [10^-2,10], and x is in the range [-x_max,x_max]),
% and vector of exemplary values of the variance
% of the normally distributed measurement error, sig2, from which the standard
% deviation is calculated (The variances are taken from the interval [10^-12,10^-2]).
X_max = logspace(-2,1,N);
sig2 = logspace(-12,-2,N);

% Calculation of the expected values of mean-square approximation errors for
% every combination of variance and maximum data value ([10^-12,10^-2]X[10^-2,10]).
% The results are stored in the NXN matrix T. Each expected value of mean-square
% approximation errors is calculated as the expected value of R many
% individual mean-square approximation errors, each
   calculated for a newly generated vector
% of measurement errors. E is a vector which
   temporarily stores these intermediate
% individual results.
T = zeros(N);
for s = 1:N
    for i = 1:N
        x = linspace(-X_max(i),X_max(i),N);
        y = f13(x);
        E = zeros(1,R);
        for r = 1:R
            y_err = y + sqrt(sig2(s))*randn(1,N);
            E(1,r) = e1(x,y_err,3,N);
        end
        exc = sqrt(sum(E.^2));
        T(s,i) = exc;
    end
end

% Code for generation of figure 5
figure(5)
surf(X_max,sig2,T)
hold on;grid on;
title(["Figure 5: Dependence of E_e , on variance of
    measurement error \sigma^2," ," and maximum
   value of data x_{max}."])
zscale("log");
yscale("log")
xscale("log")
xlabel("x_{max}")
ylabel("\sigma^2")
zlabel("E_e")
hold off;

% TASK 4 ******************************************
disp("Beggining of task 4.")

% t is the number of trails when comparing the
   runtime difference between
% all methods for finding p.
```

```
N = 100
x_max = 0.5
k = 4
t = 100

x = linspace(-x_max, x_max, N);
y = f13(x);

% Generating vectors of the results of the approximated function f13(x).

% Generating the matrix Phi as described in the document, which is used to
% find the vector of parameters p which minimizes the criterion of the least-squares
% approximation of f13(x). The function used to generate these matrices,
% Phi(x,k), is defined in the file Phi.m .
M = Phi(x,k);

% Generation of vector of paramters p using methods described in the report.
% The built-in functions tic,toc are used to measure the rutime of each method.
% This measurement is repeated t many times.
m1 = zeros(1,t);
m2 = zeros(1,t);
m3 = zeros(1,t);
for i = 1:t
    t1 = tic;
    p1 = pinv(M)*y';
    m1(1,i) = toc(t1);

    t2 = tic;
    p2 = M\y';
    m2(1,i) = toc(t2);

    t3 = tic;
    p3 = (M'*M)\(M'*y');
    m3(1,i) = toc(t3);
end
```

```

% Code for generation of figure 6,7.
figure(6)
plot(1:2*k+1, p1)
hold on;grid on;
plot(1:2*k+1, p2,"-.")
plot(1:2*k+1, p3,"--")
title("Figure 6: Plot of obtained results for vector
    p using all methods.")
legend("Using first method","Using second method","
   Using third method")
ylabel("p")
xlabel("k")
hold off;

figure(7)
plot(1:t,m1)
hold on;grid on;
plot(1:t,m2)
plot(1:t,m3)
title(["Figure 7: Runtime of methods used for" ,"
   calculation of p for 100 trials."])
legend("Using first method","Using second method","
   Using third method")
xlabel("trial")
ylabel("t[s]")
hold off;
```

7.2 Function used for generation of matrix Φ (Phi.m):

```
% This function is used to generate the Phi matrix
   as described in the
% report.

function m = Phi(x,K)
% Arguement x is the vector of data for which this
   matrix is calculated.

temp = zeros(size(x,2),(2*K+1));

```

```
    for r = 1:size(x,2)
        for k = 1:K+1
            temp(r,k) = cos((k-1)*x(r));
        end

        for l = K+2:(2*K+1)
            temp(r,l) = sin((l-K-1)*x(r));
        end
    end

m = temp;

end
```

7.3 Function used for generation of samples of the approximation function Φ (f_approx.m):

```
% Function used for least-squares approximation of f
   (x).


function y_approx = f_approx(x,p)
% Arguement x is the vector of data for which the
   function is assessd.
% p is the vector of parameters used in the function
   .

K = (size(p,1)-1)/2;
X = size(x,2);
temp = zeros(1,X);

for i = 1:X
total = p(1);
    for k = 1:K
        total = total + p(k+1)*cos(k*x(i)) + p(k+1+K
           )*sin(k*x(i));
    end
temp(1,i) = total;
end

```

```
y_approx = temp;

end
```