15 May 2019

NUMERICAL METHODS

ASSIGNMENT B
APPROXIMATION OF FUNCTIONS

Michał Kurczak

285665

Under the supervision of Andrzej Miękina, PhD

Page of content

I. Formulation of the problem

The word *approximation* is derived from Latin *approximatus*, from *proximus* meaning *very near.* Approximation is a very powerful, especially tool in engineering. Knowing the discrete form of some signal, we are able to predict its values in between the probes, without the need to perform the measurements with greater frequency. Yet, one should be aware that the problem of approximation does not require the approximated function to pass through the nodes. In my project, the least-squares approximation is to be discussed.

Least squares function approximation applies the principle of least squares to function approximation, by means of a weighted sum of other functions. The best approximation can be defined as that which minimises the difference between the original function and the approximation; for a least-squares approach the quality of the approximation is measured in terms of the squared differences between the two.

In my project, one should perform the non-linear approximation using B-spline functions. The B-spline function is a spline function that has a minimal support with respect to a given degree, smoothness, and domain partition. Any spline function of given degree can be expressed as a linear combination of B-splines of that degree.

II. Methodology

The first problem concerned graphing a function and indicating a sequence of its values which will be next used for approximation.

$$f(x) = -\cos(\pi x) e^{-x-\frac{1}{3}}$$

Formula 2.1 Function upon which the approximation problem will be performed.

$$\left\{ y_n = f(x_n) \middle| n = 1, 2, \ldots, N \right\}, \text{ where } x_n = -1 + 2\frac{n-1}{N-1}$$

Formula 2.2 Indication of a sequence of values, used for the approximation

This exercise is repeated for N = 10, 20, and 30.

The second problem of my assignment was to develop the least-squares approximation of the function given by **formula 1.1** on the basis of the sequence of its values indicated by **formula 1.2**. The operator of pseudoinversion "\" is ought to be used. As the basis of linearly independent functions, the B-spline functions was used.

$$Bs_k(x) = B_S\left(2(x - x'_k) + 2\right) \quad B_S(x) = \begin{cases} x^3 & x \in [0,1) \\ -3(x-1)^3 + 3(x-1)^2 + 3(x-1) + 1 & x \in [1,2) \\ 3(x-2)^3 - 6(x-2)^2 + 4 & x \in [2,3) \\ -(x-3)^3 + 3(x-3)^2 - 3(x-3) + 1 & x \in [3,4] \end{cases}$$

$$\text{where } x'_k = -1 + 2\frac{k-1}{K-1} \quad \text{for } k = 1, 2, \ldots, K$$

Formula 2.3 B-spline functions used as the basis for the approximation

The exercise was to be performed for different pairs of N and K.

A transposed vector of parameters **p**=[p1, …,pk] is sought for the vector that makes the linear combination of linearly independent functions { $\phi_k(x)$ | k = 1,2,…, K}:

$$\hat{f}(x;\mathbf{p}) = \sum_{k=1}^{K} p_k \phi_k(x)$$

Formula 2.4 Approximation of the discrete function

With respect to he least-squares criterion

$$J_2(\mathbf{p}) = \sum_{n=1}^{N} \left[\hat{f}(x_n;\mathbf{p}) - f(x_n) \right]^2$$

Formula 2.5 Least-squares criterion

Then, the necessary condition of the minimum has the form:

$$\frac{\partial J_2(\mathbf{p})}{\partial p_k} = 0 \quad \text{for} \quad k = 1, \ldots, K$$

Formula 2.6 Condition of the minimum

Which boils down the problem to the equation

$$\mathbf{\Phi}^T \cdot \mathbf{\Phi} \cdot \mathbf{p} = \mathbf{\Phi}^T \cdot \mathbf{y}$$

Formula 2.7 Formula used for obtaining the p vector

where:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_K(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_K(x_N) \end{bmatrix}$$

Formula 2.8 Matrix phi obtained with the B-spline functions

$$\mathbf{y} = \left[f(x_1) \;\; f(x_2) \;\; \ldots \; f(x_N) \right]^T$$

Formula 2.9 Vector of y coordinates of the sequence of the points obtained in the first task

Hence

$$\hat{\mathbf{p}} = \left(\mathbf{\Phi}^T \cdot \mathbf{\Phi} \right)^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{y}$$

Formula 2.10 Sought vector p

The final solution of the approximation can be presented as:

$$\tilde{f}(x, \boldsymbol{p}) = \sum_{k=1}^{K} p_k \varphi_k(x)$$

Formula 2.11 The values of the approximated function

The third task was to investigate the dependence of the accuracy of the approximation on the values of N and K. Accuracy indication was to be analysed by the means of the root-mean-square error and the maximum error.

$$\delta_2(K, N) = \frac{\left\| \hat{f}(x; K, N) - f(x) \right\|_2}{\left\| f(x) \right\|_2}$$

Formula 2.12 Root-mean-square error

$$\delta_\infty(K, N) = \frac{\left\| \hat{f}(x; K, N) - f(x) \right\|_\infty}{\left\| f(x) \right\|_\infty}$$

Formula 2.13 Maximum error

Results should be presented for a three-dimensional graph of the functions of the root-mean-square error and the maximum error for N [5;50] and K < N.

The fourth task concerned the investigation of the dependence indicators from the previous task on the standard deviation of random errors the data usef for approximation are corrupted with.

$$\sigma_y \in \left[10^{-5}, 10^{-1}\right]$$

Formula 2.14 Standard deviation, corrupting the data

In order to perform the analysis, one should:

-Generate the error- corrupted data according to the formula:

$$\tilde{y}_n = y_n + \Delta\tilde{y}_n \text{ for } n = 1, \ldots, N$$

Formula 2.15 Error corrupted data

-For each value of the standard deviation, determine the values N and K minimising errors and compute them

$$\delta_{2,MIN}\left(\sigma_y\right) \equiv \delta_2\left(\bar{K}, \bar{N}\right)$$

Formula 2.16 Minimal error

-Approximate the sequence of pairs of standard deviation values and corresponding errors, determined for several dozen values of the standard deviation, by means of the MATLAB operator polyfit

$$\left\langle \sigma_y, \delta_{2,MIN}\left(\sigma_y\right)\right\rangle$$

Formula 2.17 Pairs of standard deviation values and corresponding errors

III. Results

Figures 3.1-3.9 represent continuous function given as a base for generating N points and the approximation of the discrete function. K is the number of B-spline parameters used for the Phi matrix generation.

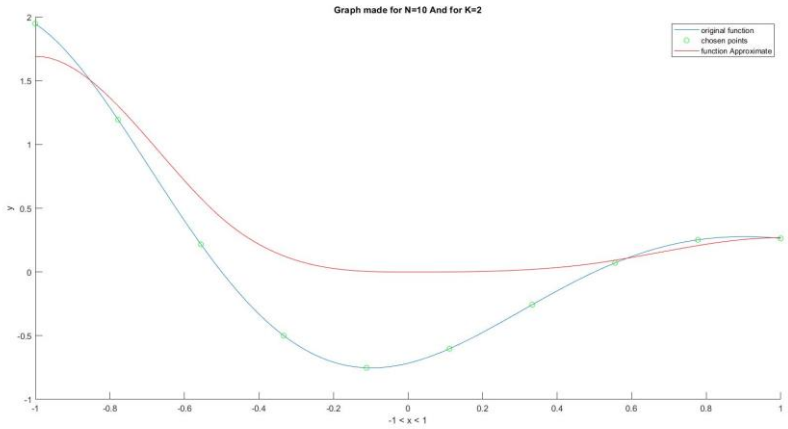

Figure 3.1 Approximation for N= 10 and K=2

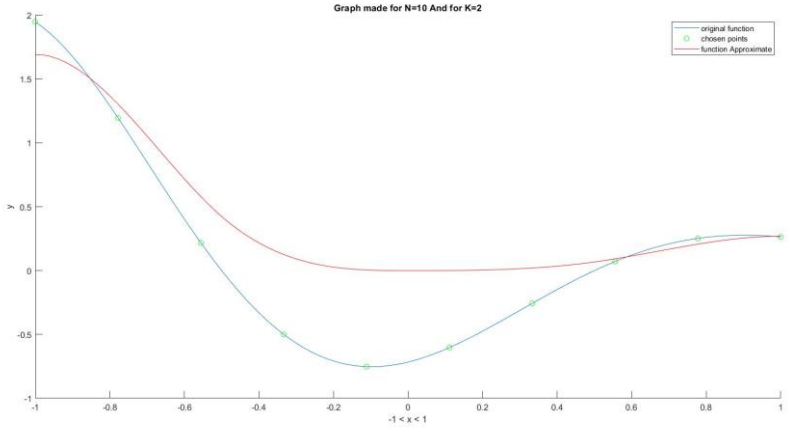

Figure 3.2 Approximation for N= 10 and K=6

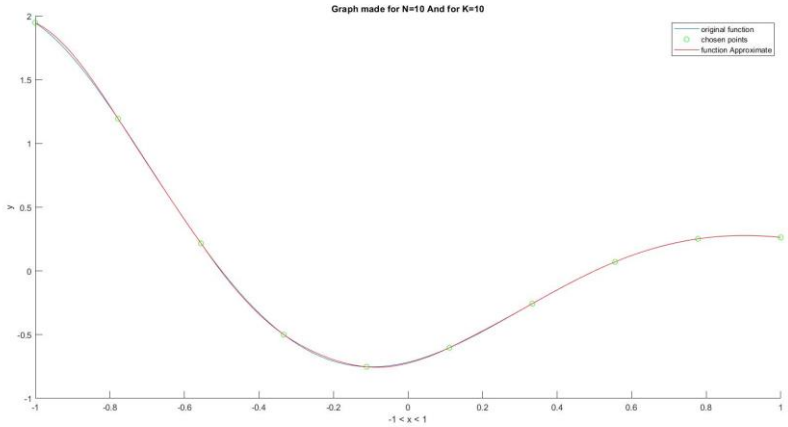

Figure 3.3

Approximation for N= 10 and K=10

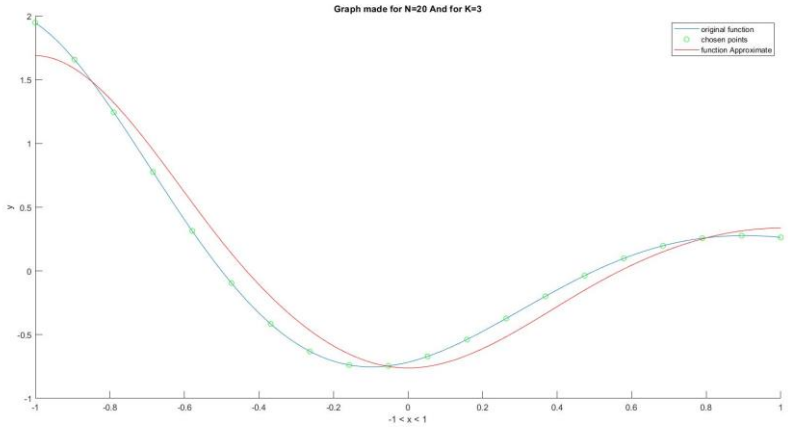

Figure 3.4 Approximation for N= 20 and K=3

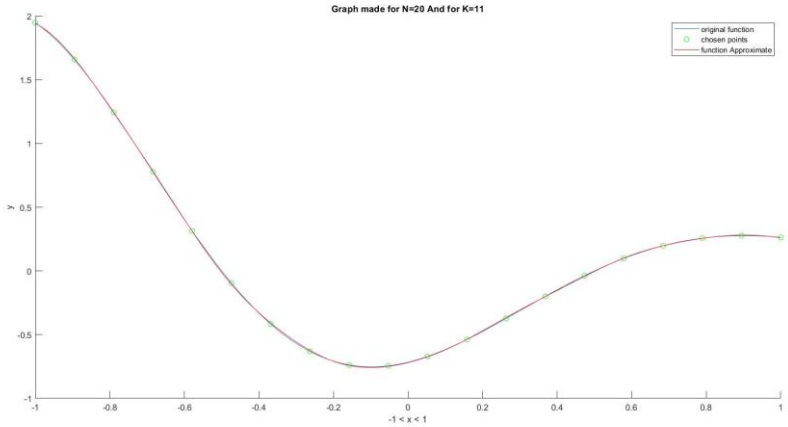

Figure 3.5 Approximation for N= 20 and K=11

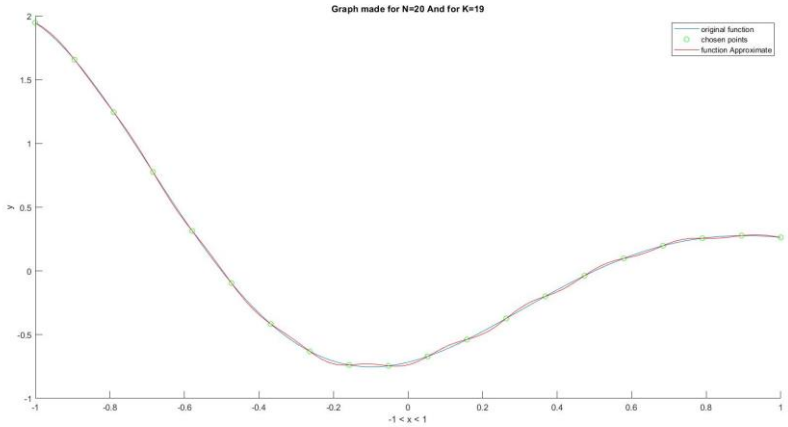

Figure 3.6 Approximation for N= 20 and K=16

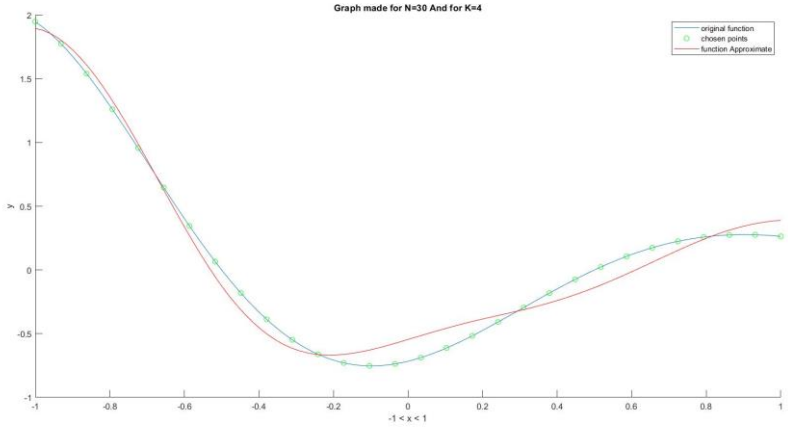

Figure 3.7 Approximation for N= 30 and K=4

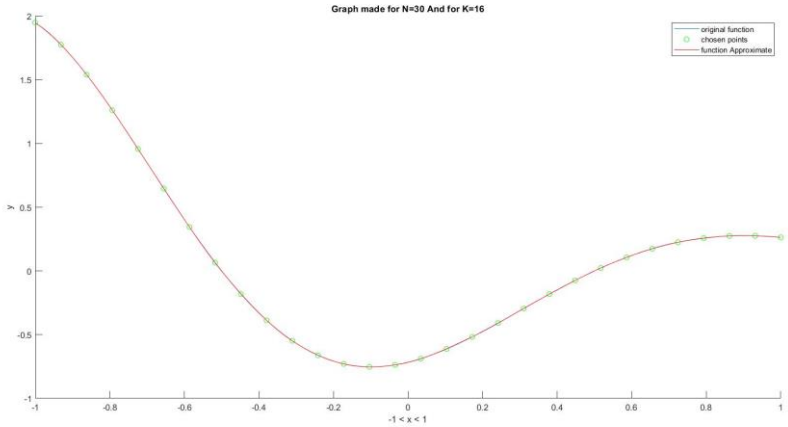

Figure 3.8 Approximation for N= 30 and K=16

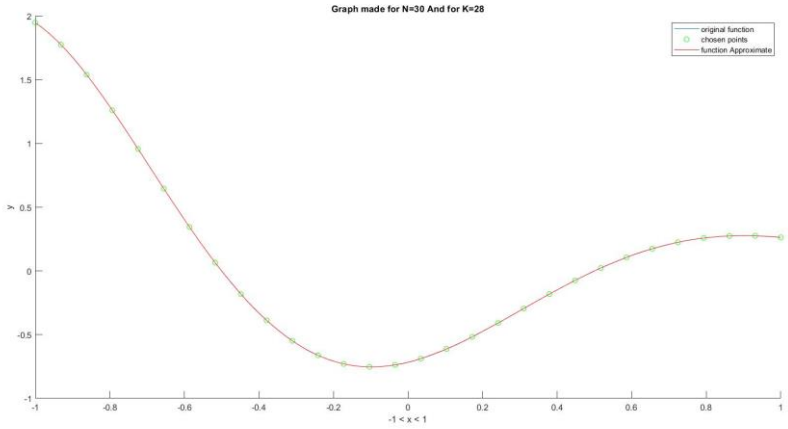

Figure 3.9 Approximation for N= 30 and K=28

As we can see, from figures depicted above, the tendency is that with the growth of values K and N, we obtain better results of the approximation, yet we have to remember that the value K cannot exceed the N.

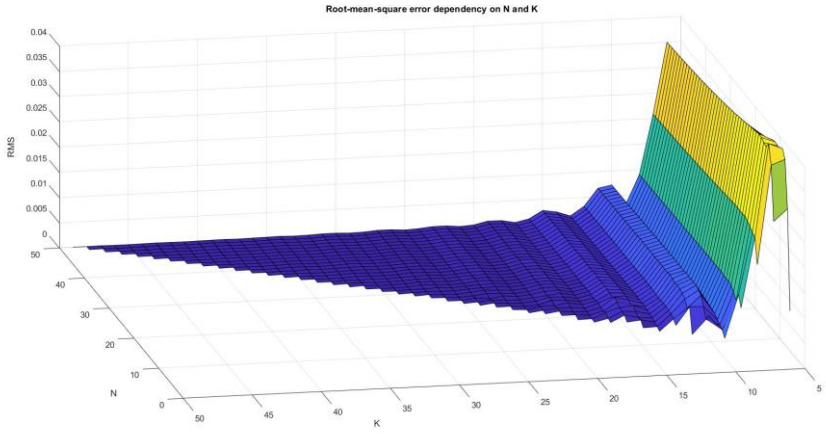

Figure 3.10 Graph of the root-mean-square error dependency on the parameters N and K

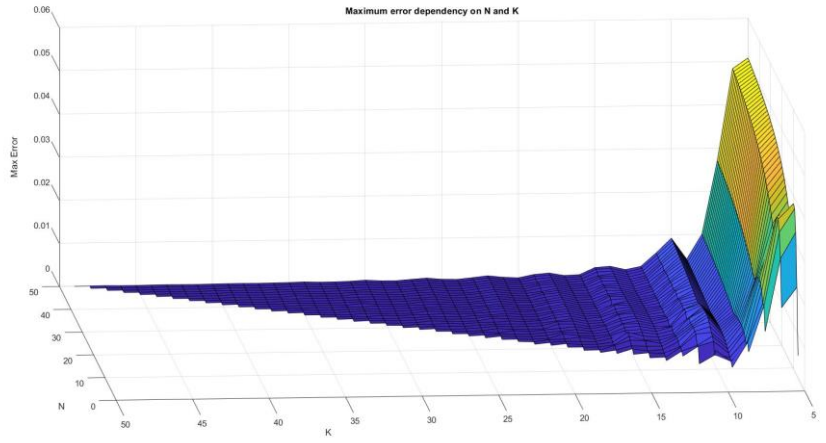

Figure 3.11 Graph of the maximum error dependency on the parameters N and K

The above figures confirm prior suppositions that with the growth of the parameters N and K, the error of approximation is getting smaller.

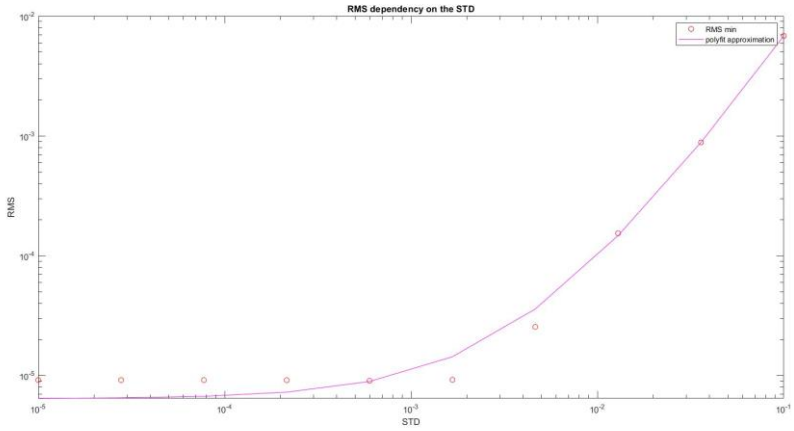

Figure 3.12 Minimal RMS dependency on standard deviation with the approximation obtained by means of the MATLAB operator polyfit

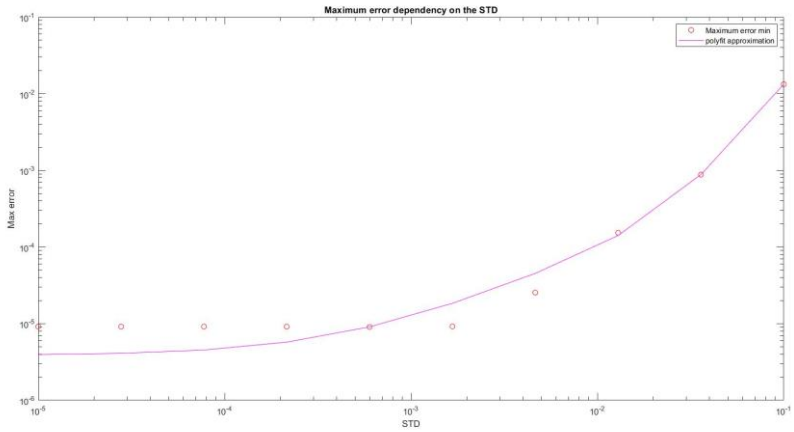

Figure 3.12 Minimal maximum error dependency on standard deviation with the approximation obtained by means of the MATLAB operator polyfit

As we can see, with the growth of data corruption, both the RMS and maximum error grows. MATLAB's operator polyfit curve shows the general trend of the growth of both errors.

IV. Conclusions

Least-squares method is a very powerful tool that has many applications in engineering. Its best advantages are:

- Simplicity – easy to understand and implement
- Infinite resolution – it is possible to predict the value of the data for any given point between nodes
- Good accuracy – we are able to obtain a prediction of a signal, having proper density of the probes

The disadvantages are:

- Dependency on the probes – density of probes strongly determines the accuracy of our measurement
- Dependency on the base function – the accuracy of the approximation strongly depends on the base function we choose to approximate the data with
- Susceptibility to noise – the accuracy of the approximation strongly decreases with the growth of the error corruption of the data.

Concluding, the least-squares method is a very powerful tool, with many practical applications, yet one need to be fully aware of its pros and cons in order to get the most out of its potential.

V. Bibliography

[1] Roman Z. Morawski - ENUME 2019 – Lecture notes

[2] Krystian Król “Wstęp do metod numerycznych”

[3] https://en.wikipedia.org/wiki/Approximation

VI. Appendix

```
%Function calculating root-mean-square error
%On parameters N and K
function y=RMS(N,K)
nominator=zeros(1,N);
denominator=zeros(1,N);
x_n=GenerateXn(N);
for i=1:N
nominator(1,i)=Approximate(x_n(i),N,K)-FirstFunction(x_n(i));
denominator(1,i)=FirstFunction(x_n(i));
end
y=norm(nominator,2)/norm(denominator,2);
end

%Function calculating maximum error
%On parameters N and K
function y=MxError(N,K)
nominator=zeros(1,N);
denominator=zeros(1,N);
x_n=GenerateXn(N);
for i=1:N
nominator(1,i)=Approximate(x_n(i),N,K)-FirstFunction(x_n(i));
denominator(1,i)=FirstFunction(x_n(i));
end
y=norm(nominator,inf)/norm(denominator,inf);
end

%Function generating Matrix of Phi parameters
function y=GeneratePhi(N,K)
Fi=zeros(N,K);
for n=1:N
    x_n=-1+2*(n-1)/(N-1);
    for k=1:K
        Fi(n,k)=Bsk(x_n,k,K);
    end
end
y=Fi;
end

%Generating values of the main function
%For N points
function y=GenerateY(N)
yn=zeros(1,N);
for n=1:N
   x_n=-1+2*(n-1)/(N-1);
```

```
   yn(n)=FirstFunction(x_n);
end
y=yn;
end

%Generating x values of the N points
function y=GenerateXn(N)
x_n=zeros(1,N);
for n=1:N
   x_n(n)=-1+2*(n-1)/(N-1);
end
y=x_n;
end

%Spline Function, used for Phi matrix generation
function y=Bsk(x,k,K)
xk=x_k(k,K);
y=Bs(2*(x-xk)+2);
end

%Function used for approximation the main function
function y=Approximate(x,N,K)
  Fi=GeneratePhi(N,K);
        y=GenerateY(N);
        p=Fi.'*Fi\Fi.'*y.';
y=0;
K=length(p);
for i=1:K
    y=y+p(i)*Bsk(x,i,K);
end
end

%auxilary function used in B-spline function
function x=x_k(k,K)
x=2*((k-1)/(K-1))-1;
end

%The main function to approximate
function y=FirstFunction(x)
y=-cos(pi.*x).*exp(1).^(-1/3-x);
end

%B-sline function
function y=Bs(x)
if (x>=0 && x<1)
    y=x^3;
elseif (x>=1 && x<2)
    y= -3*(x-1)^3+3*(x-1)^2+3*(x-1)+1;
elseif (x>=2 && x<3)
    y= 3*(x-2)^3-6*(x-2)^2+4;
elseif (x>=3 && x<=4)
```

```
    y= -(x-3)^3+3*(x-3)^2-3*(x-3)+1;
else
    y=0;
end
end

%---------Task 4 error corrupted functions-----
function y=CorruptedRMS(N,K,sigma)
nominator=zeros(1,N);
denominator=zeros(1,N);
x_n=GenerateXn(N);
for i=1:N
nominator(1,i)=CorruptedApproximate(x_n(i),N,K,sigma)-
FirstFunction(x_n(i));
denominator(1,i)=FirstFunction(x_n(i));
end
y=norm(nominator,2)/norm(denominator,2);
end

function y=CorruptedMxError(N,K,sigma)
nominator=zeros(1,N);
denominator=zeros(1,N);
x_n=GenerateXn(N);
for i=1:N
nominator(1,i)=CorruptedApproximate(x_n(i),N,K,sigma)-
FirstFunction(x_n(i));
denominator(1,i)=FirstFunction(x_n(i));
end
y=norm(nominator,inf)/norm(denominator,inf);
end


function y=GenerateCorruptedY(x, sigma)
N=length(x);
yn=zeros(1,N);
for n=1:N
   x_n=-1+2*(n-1)/(N-1);
   yn(n)=FirstFunction(x_n)+randn()*sigma^2;
end
y=yn;
end


function y=CorruptedApproximate(x,N,K, sigma)
  Fi=GeneratePhi(N,K);
        x_n=GenerateXn(N);
        y=GenerateCorruptedY(x_n,sigma);
        p=Fi.'*Fi\Fi.'*y.';
y=0;
K=length(p);
for i=1:K
```

```
    y=y+p(i)*Bsk(x,i,K);
end
end
```