Warsaw, 14.V.2019

Numerical Methods (ENUME)

Assignment B: Approximation of functions

Final Report

Name: Jan Radzimiński

Index number: 293052

Supervisor: Andrzej Miękina Ph.D.

Warsaw, 14.V.2019

Table of contents:

Warsaw, 14.V.2019

0. General Remarks

This project was implemented using Matlab environment. The Matlab script containing solutions to all tasks is called ‘’AssB_JR.m”.

After presenting solution to a given task, script stops (by function “pause”) and it waits for any key to be pressed to move on to the next task.

Also, after every task, script calls “clear”, “close all”, “hold off” and “clc” functions, so tasks don’t interfere with each other and each task is more readable.

At the end of the report can be found original page with given assignment and printed Matlab code used to solve given problems.

1. Task 1 – Graph of function

1.1. Introduction

The **first task** was about plotting function:

$$f(x) = -\cos(\pi x)\, e^{-x-\frac{1}{3}} \qquad for\ x \in [-1, 1]$$

that will be later approximated using 10, 20 or 30 single points belonging to this curve, equally distributed on function domain.

1.2. Graph of the function

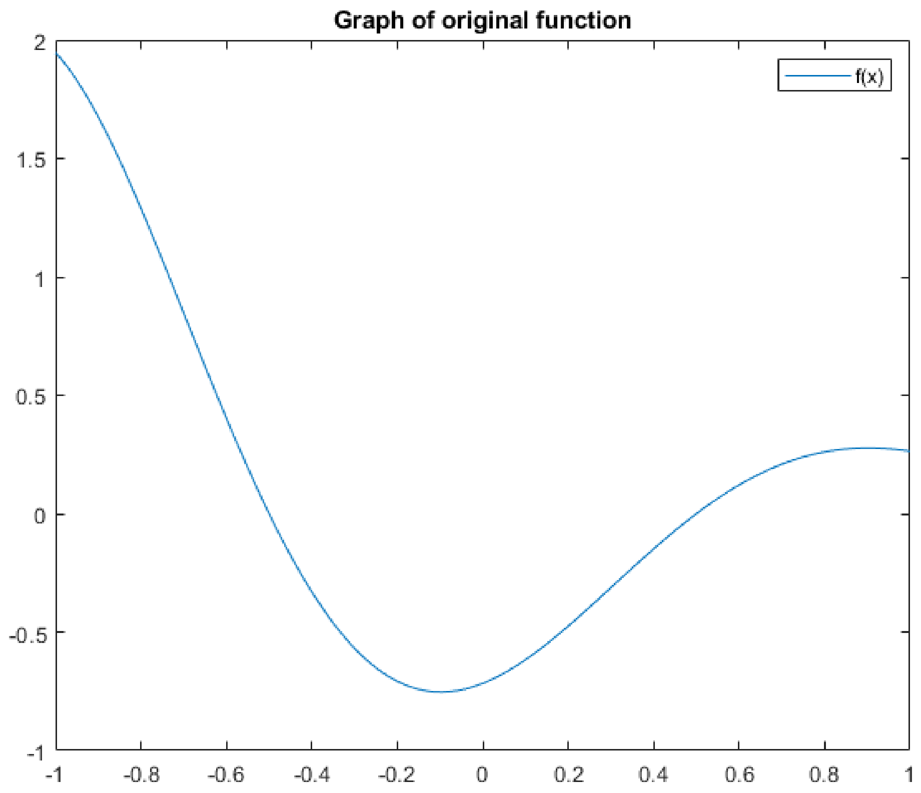

Figure (1.1)

2. Task 2 – Least-squares approximation

2.1. Introduction

In the **second task**, main goal was to develop a program that will make least-squares approximation for three set of points created in task 1.

Least-squares approximation is a typical approach for approximating shape of the given function using only given set of points. This method, first published by Adrien Marie Legendre in 1805, is commonly used in various scientific fields, having its most important application is in data fitting.

Developed program uses built in pseudoinversion operator in Matlab and functions:

$$G_k(x) = \frac{5}{\sqrt{2\pi}} e^{-25(x-x_k)^2} \qquad where \;\; x_k = -1 + 2\frac{k-1}{K-1} \qquad for\; k = 1, \dots, K$$

as a basis of linearly independent functions.

The script creates Φ matrix with Gk functions and N points that it approximates. Then it calculates p vector:

$$\boldsymbol{p} = (\Phi^T \cdot \Phi)^{-1} \cdot \Phi^T \cdot y \qquad \rightarrow \boldsymbol{p} = (\Phi^T \cdot \Phi)/(\Phi^T \cdot y\,)$$

where ‘’/’’ is matlab pseudoinversion operator and y is vector with points used for approximation

Then, it creates another matrix Φ_n, with 1000 points between -1 and 1, and plots p* Φ_n which is a curve of made approximation.

2.2. Graphs of approximations

2.2.1. N=10

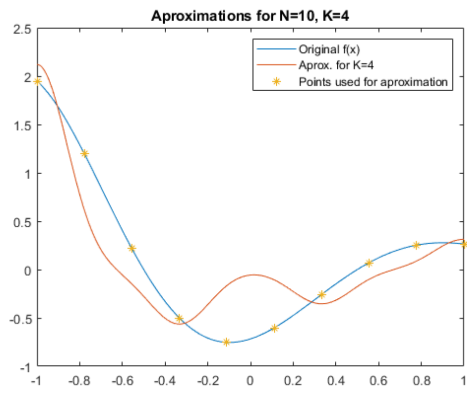

Figure (2.1)

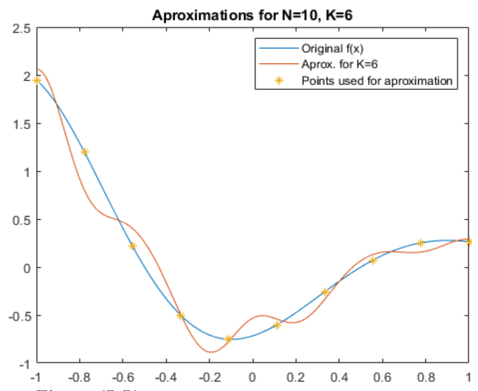

Figure (2.2)

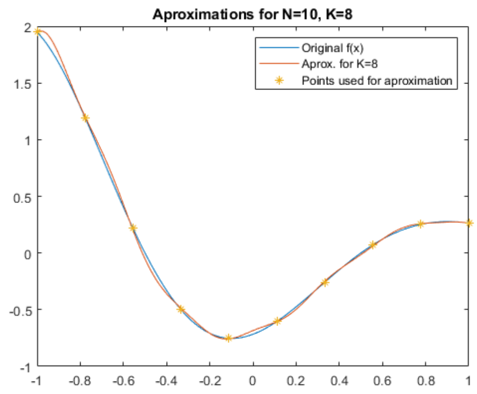

Figure (2.3)

2.2.2. N=20

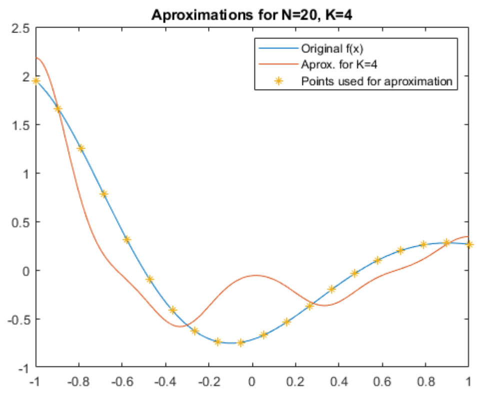

Figure (2.4)

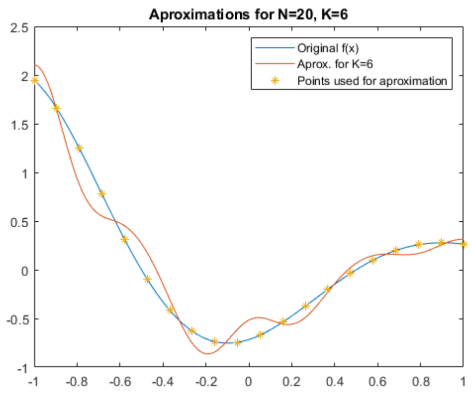

Figure (2.5)

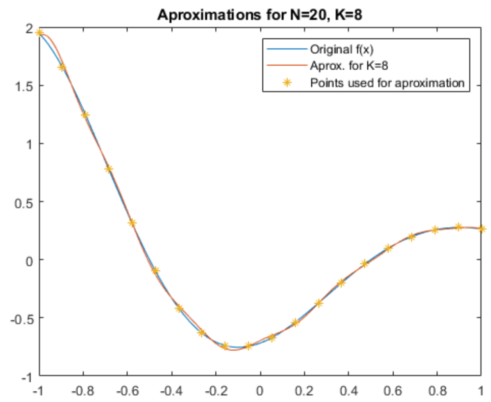

Figure (2.6)

2.2.3. N=30

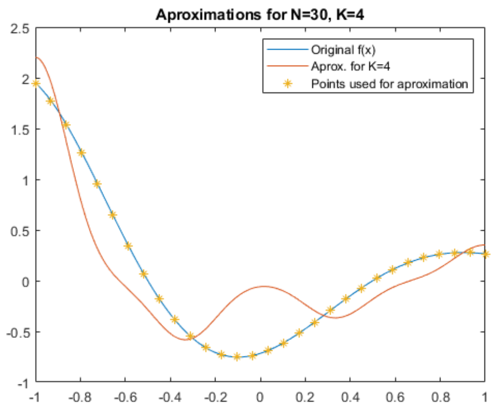

Figure (2.7)

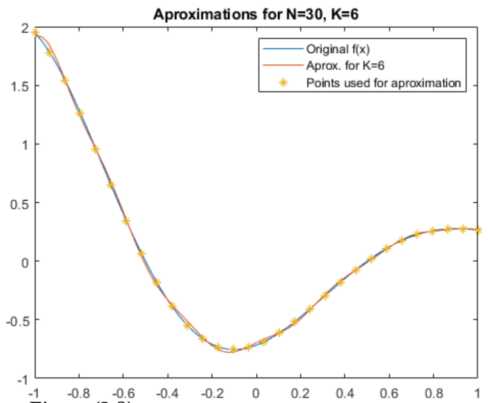

Figure (2.8)

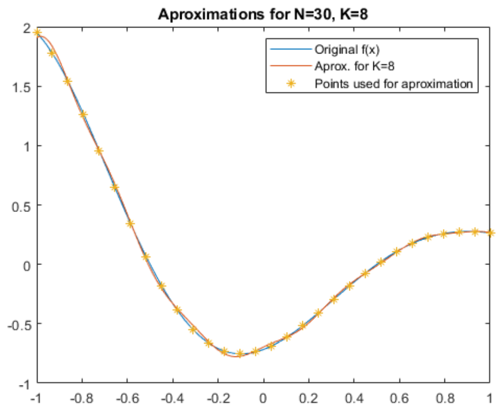

Figure (2.9)

2.3. Conclusions

As can be seen on obtained graphs of approximations, the approximated curve fits better or worse, depending on N and K used in approximation.

Since N is the number of points used in method, it can be clearly seen that N does not influence made approximations in a major way like K, but generally the more points are used for approximations the better the result – one may observe this especially in K=6 case – with 10 points curve its rather bad, with 20 its closer to original function and with 30 points it almost covers the original curve, similar as in K=8.

Finally, it is clear that for bigger value of K the approximations are better (although K should be smaller than N) – in all cases the best result was achieved with biggest K=8 (with higher K values while they are smaller than N results was even better). For K=4 in all cases approximations are not good when compared to the others, the curve starts looking better for K=6 and for K=8 in all cases shape of approximated function is very similar to the original one.

3. Task 3 – Dependence of the accuracy of approximation on N and K

3.1. Introduction

The aim of the third task was to carry out a systematic investigation of the dependence of the accuracy of approximation on values N and K. For this purpose, two indicators was used:

$$\delta_2(K,N) = \frac{||f(x;K,N) - f(x)||_2}{||f(x)||_2} \qquad \rightarrow \textit{the root mean suared error}$$

$$\delta_\infty(K,N) = \frac{||f(x;K,N) - f(x)||_\infty}{||f(x)||_\infty} \qquad \rightarrow \textit{the maximum error}$$

For more clear and readable results, from both calculated errors there was taken $\log_{10}$ (N, and K axes are normal).

3.2. Graphs

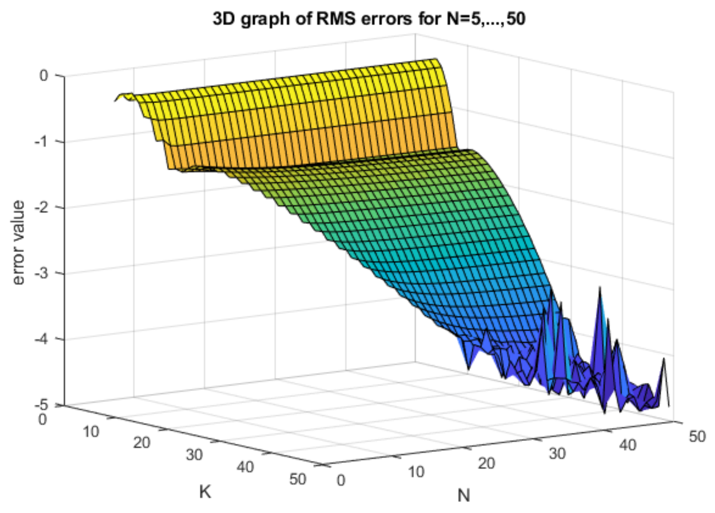

Figure (3.1)

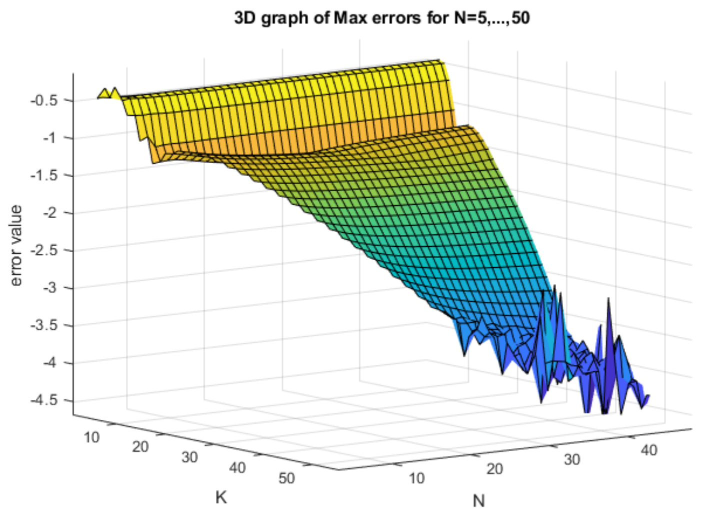

Figure (3.2)

3.3. Conclusions

From these graphs we can see, that biggest errors occur for small amount of K (between 3 and 6), for every N. At K=6 errors start to drop to deepen around K=8 – in previous task it was shown that for this value in all cases curve fit best to original one.

Then for larger K's values of errors smoothly drop down, and for K's bigger that about 35 there appears ''noise'' – values start oscillating from big ones to small ones, probably, because number of points used for approximation is much larger and points close to each other can influence much shape of a curve, which creates these single peaks and bumps.

Generally such graphs can be useful for finding smallest K for which approximation may be acceptable – in this case is K=8 – on both graphs there is a visible dip on graph surface at that exact value.

Also, we can see that errors does not depend much on N (directly, because of course for bigger N we may use bigger K which may result in better results), for all of them at the same K values are more or less the same (on max error graph *(figure 2.3)* there can be seen small deep in surface at n around 25).

4. Investigation of the dependence of the error indicators on the standard deviation

4.1. Introduction

In the last task, the goal was to crate error-corrupted data with given σ, where corrupted data would be points belonging to original function + σ where σ belongs to [-5, -1]. Then, the aim was to measure influence of changing σ on minimum of the RMS and Max error value of the approximations functions made in previous tasks. Then for given points fitting curve was find with usage of ‘’polyfit’’ matlab function.

4.2. Graph

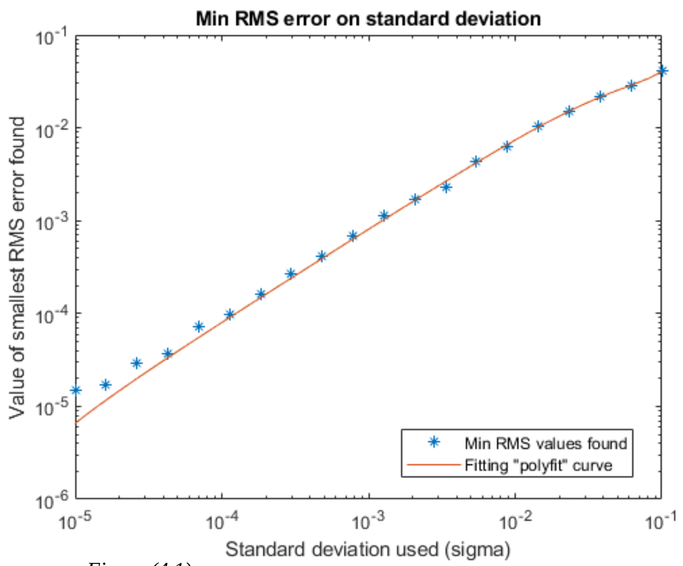

Figure (4.1)

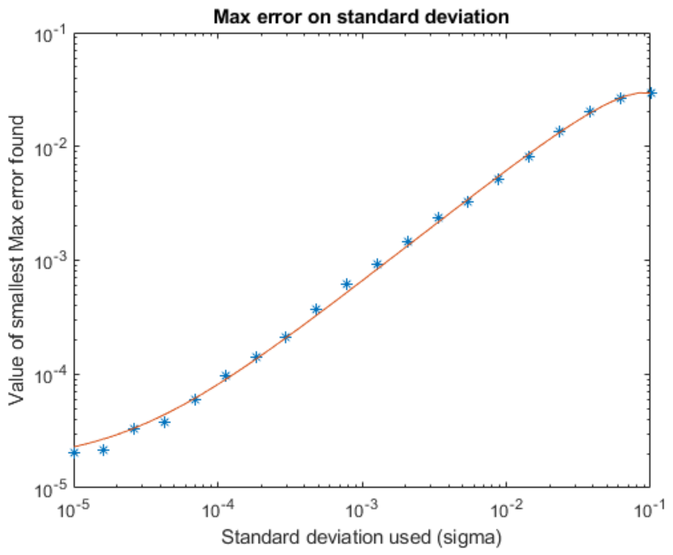

Figure (4.2)

4.3. Conclusions

As we can see on the graphs, for smaller σ the errors are smaller, and for larger σ they increase. That is intuitive since for smaller values of σ the points used for approximation are less corrupted, therefore the errors when compared to original function will be smaller as well. In this case these values increase linearly, however, the values increase slightly ‘slower’ at values closer to ends of interval.

5. Script functions description

- `function [Y] = AproxFx(N, K, Xarg)`
  - Function that creates vector of approximated values for given N,K and 1000 points between -1 and 1 in Xarg
- `function [Y] = AproxFx2(Yn, K, Xarg)`
  - Similar as previous one, but used in task 4 (separate function since error corrupted data was needed)
- `function [Y] = fx(X)`
  - Function that takes vector X and creates results vector Y according to assignment function formula
- `function [Y] = Gk(N, K)`
  - basis of linearly independent function used for crating fi matrices
- `function [Y] = S2(N, K, Xarg)`
  - function calculating RMS error for approximation with given N, K and points between -1 and 1
- `function [Y] = Sinf(N, K, Xarg)`
  - same as previous but for Max Error
- `function [X, Y] = Corrupted(N, sig_pow)`
  - function that creates corrupted data for given N and given sigma

```
% JAN RADZIMINSKI (293052)
% ==== ENUME - ASSIGNMENT B PROJECT ====
hold off;
close all;
clear;
clc;

% ========== TASK 1 =============
% Original function
Xarg = linspace(-1, 1, 1000);
% this vector will be later used for making
graphs in all cases
Y = fx(Xarg);

figure(1)
plot(Xarg, Y);
hold on;
title('Graph of original function');
legend('f(x)');

pause;
hold off;
close all;

%Generating points that will be used in
aprox.
%N = 10
Xn1=linspace(-1, 1, 10);
Yn1=fx(Xn1);

%N = 20
Xn2=linspace(-1, 1, 20);
Yn2=fx(Xn2);

%N = 30
Xn3=linspace(-1, 1, 30);
Yn3=fx(Xn3);

% ========== TASK 2 =============

% Ploting aproximated funcitons

%N = 10
figure(1);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(10, 4, Xarg))
hold on;
plot(Xn1, Yn1, '*')
hold on;
title('Aproximations for N=10, K=4');
legend('Original f(x)', 'Aprox. for K=4',
'Points used for aproximation');

figure(2);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(10, 8, Xarg))
hold on;
plot(Xn1, Yn1, '*')
hold on;
title('Aproximations for N=10, K=8');
legend('Original f(x)', 'Aprox. for K=8',
'Points used for aproximation');

%N = 20
figure(4);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(20, 4, Xarg))
hold on;
plot(Xn2, Yn2, '*')
hold on;
title('Aproximations for N=20, K=4');
figure(5);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(20, 8, Xarg))
hold on;
plot(Xn2, Yn2, '*')
hold on;
title('Aproximations for N=20, K=8');
legend('Original f(x)', 'Aprox. for K=8',
'Points used for aproximation');

%N = 30

figure(7);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(30, 4, Xarg))
hold on;
plot(Xn3, Yn3, '*')
hold on;
title('Aproximations for N=30, K=4');
legend('Original f(x)', 'Aprox. for K=4',
'Points used for aproximation');

figure(8);
plot(Xarg , Y)
hold on;
plot(Xarg , AproxFx(30, 8, Xarg))
hold on;
plot(Xn3, Yn3, '*')
hold on;
title('Aproximations for N=30, K=8');
legend('Original f(x)', 'Aprox. for K=8',
'Points used for aproximation');

pause;

% ========== TASK 3 =============
hold off;
close all;
clc;

% Calculating RMS and Max error matrices
N = linspace(5, 50, 10);
for i=5:50
    for j=3:(i-1)
        S2tab(i, j) = S2(i, j, Xarg);
        SItab(i, j) = Sinf(i, j, Xarg);
    end
end

% Ploting 3D Graphs
figure(1)
surf(log10(S2tab));
title('3D graph of RMS errors for
N={5,...,50}')
ylabel('N')
xlabel('K')
zlabel('RMS error value')

figure(2)
surf(log10(SItab));
title('3D graph of Max errors for
N={5,...,50}')
ylabel('N')
xlabel('K')
zlabel('Max error value')

pause;
% ========== TASK 4 =============
hold off;
close all;
clc;
```

```
sig = linspace(-5, -1, 20);
k=1;
for p=sig
    RMSmin(k)=1000;
    Mmin(k)=1000;
for i=5:50
    [Xc1, Yc1] = Corrupted(i, p);
    for j=3:(i-1)
        S2ctab(i, j) = norm(AproxFx2(Yc1, 
j, Xarg) - fx(Xarg))/norm(fx(Xarg));
        SIctab(i, j) = norm(AproxFx2(Yc1, 
j, Xarg) - fx(Xarg), 'inf')/norm(fx(Xarg), 
'inf');

%         finding smallest value:
        if(S2ctab(i, j)>0)
            if(S2ctab(i, j)<RMSmin(k))
RMSmin(k)=S2ctab(i, j);
            end
        end

        if(SIctab(i, j)>0)
            if(SIctab(i, j)<Mmin(k))
Mmin(k)=SIctab(i, j);
            end
        end
    end

end
k=k+1;
end

% Finding best fit to the points
sig=10.^sig;
space=logspace(-5, -1);
p = polyfit(sig, RMSmin, 3);
pm = polyval(p, space);

figure(1)
loglog(sig, RMSmin, '*');
hold on;
loglog(space, pm);
hold on;
xlabel('Standard deviation used (sigma)');
ylabel('Value of smallest RMS error 
found');

p = polyfit(sig, Mmin, 3);
pm = polyval(p, space);

figure(2)
loglog(sig, Mmin, '*');
hold on;
loglog(space, pm);
hold on;
xlabel('Standard deviation used (sigma)');
ylabel('Value of smallest Max error 
found');


% ================================
% Used functions:

function [Y] = AproxFx(N, K, Xarg)

Xn = linspace(-1, 1, N);
Yn(:) = fx(Xn(:));

Fi(:, :) = Gk(Xn, K);
A = ((Fi*Fi.')\(Fi*Yn.')).';

G(:, :) = Gk(Xarg, K);
Y=A*G;
end
```

```
function [Y] = AproxFx2(Yn, K, Xarg)

Xn = linspace(-1, 1, size(Yn,2));
Fi(:, :) = Gk(Xn, K);
A = ((Fi*Fi.')\(Fi*Yn.')).';
G(:, :) = Gk(Xarg, K);
Y=A*G;
end

function [Y] = fx(X)
Y(:)=-cos(pi*X(:)).*exp(-X(:)-1/3);
end

function [Y] = Gk(N, K)
xk=linspace(-1, 1, K);
Y(:, :) = 5/(2*pi).*exp(-25*(N(1, :)-
xk(:)).^2);
end

function [Y] = S2(N, K, Xarg)
Y=norm(AproxFx(N, K, Xarg) - 
fx(Xarg))/norm(fx(Xarg));
end

function [Y] = Sinf(N, K, Xarg)
Y=norm(AproxFx(N, K, Xarg) - fx(Xarg), 
'inf')/norm(fx(Xarg), 'inf');
end

function [X, Y] = Corrupted(N, sig_pow)
X=linspace(-1, 1, N);
Y = fx(X);
R=randn(1, N)*(10^sig_pow);
Y=Y+R;
end
```