ASSIGNMENT B: APPROXIMATION OF FUNCTION

Author: Minh Hieu Do (ID: 288414)

Course: Numerical Methods (ENUME)

Advisor: Andrzej Miekina, Ph.D., Assistant Professor

Warsaw, 22/05/2018

Contents

1. The concise description of numerical algorithms

a. *Least-squares approximation of a function*

A set of *n* discrete data points $\{x_n, f(x_n)\}, n = 1, 2, \dots, N$ is given.

Consider the function

$$\hat{f}(x; \alpha) = \sum_{k=1}^{K} \alpha_k \, \Phi_k(x)$$

where $\Phi_k(x)$ is a linearly independent function (base function)

The coefficients $\alpha = [\alpha_1 \dots \alpha_K]^T$ are given by the solution to the equation

$$\boldsymbol{\Phi}^{\boldsymbol{T}} \cdot \boldsymbol{\Phi} \cdot \boldsymbol{\alpha} = \boldsymbol{\Phi}^{\boldsymbol{T}} \cdot \boldsymbol{y}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} \Phi_1(x_1) & \Phi_2(x_1) & \cdots & \Phi_K(x_1) \\ \Phi_1(x_2) & \Phi_2(x_2) & \cdots & \Phi_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(x_N) & \Phi_2(x_N) & \cdots & \Phi_K(x_N) \end{bmatrix}$$

and

$$\boldsymbol{y} = [f(x_1)\; f(x_2) \dots\; f(x_n)]^T$$

b. *Legendre polynomials*

The set of *Legendre polynomials* is orthogonal on [-1, 1] with *w(x) = 1*

$P_k(x)\; for\; k = 1, 2 \dots K$:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_k(x) = \frac{2k-1}{k} x P_{k-1}(x) - \frac{k-1}{k} P_{k-2}(x) \quad for\; k > 1$$

c. *Accuracy indictors*

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

2. The methodology for testing numerical algorithms

- Make the graphs of the function $f(x) = -\sin(\pi x)\, e^{-x}\ with - 1 \leq x \leq 1$
- Design a MATLAB procedure using the method of least square and the *Legendre polynomials* as the base function to approximate the function $f(x)$ on the basis of the data $\{(x_n, y_n)|n = 1, \dots, N\}$.
- Plot the graphs to compare the approximation to the exact data for several pairs of the values of the parameters N and K.
- Make the matrixes of $\delta_2(K, N)$ and $\delta_\infty(K, N)$ for K = 4, …, 40; N = k+2, …, 42.
- Plot 3D graphs of the dependence of $\delta_2(K, N)$ and $\delta_\infty(K, N)$ on K and N.
- Repeat the above study with the pseudorandom additive errors $\{\Delta\tilde{y}_n|n = 1, \dots, N\}$ following the normal distribution with the zero mean and variance σ_y^2:

  $$\{(x_n, \tilde{y}_n)|n = 1, \dots, N\}, where\ \tilde{y}_n = y_n + \Delta\tilde{y}_n$$

  by using the MATLAB function ***randn*** to generate the errors.

3. The results of testing numerical algorithms

a. Problem 1

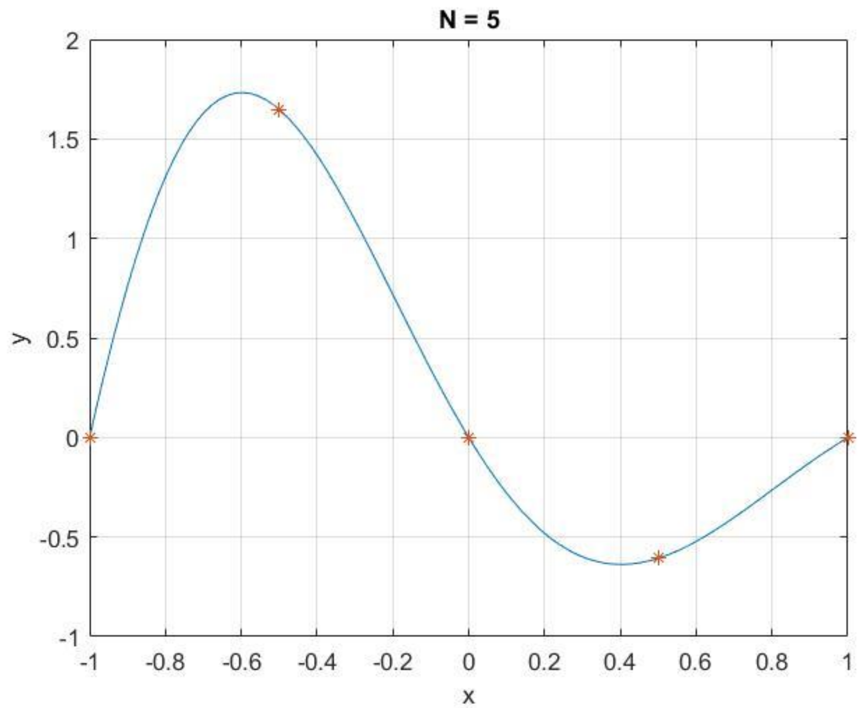

Figure 1. Exact data for N = 5

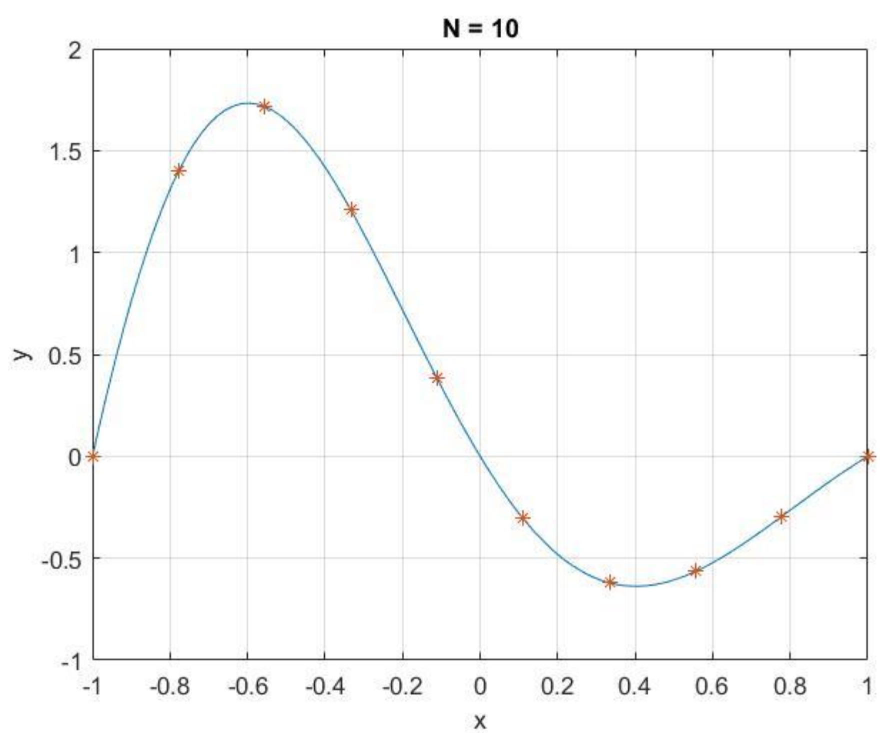

Figure 2. Exact data for N = 10

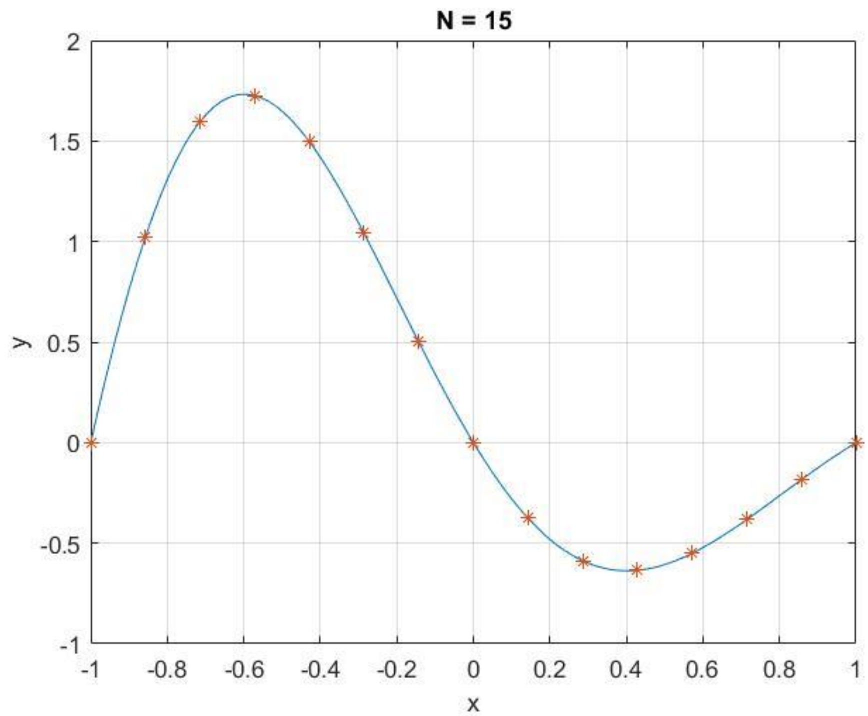

Figure 3. Exact data for N = 15

b. Problem 2

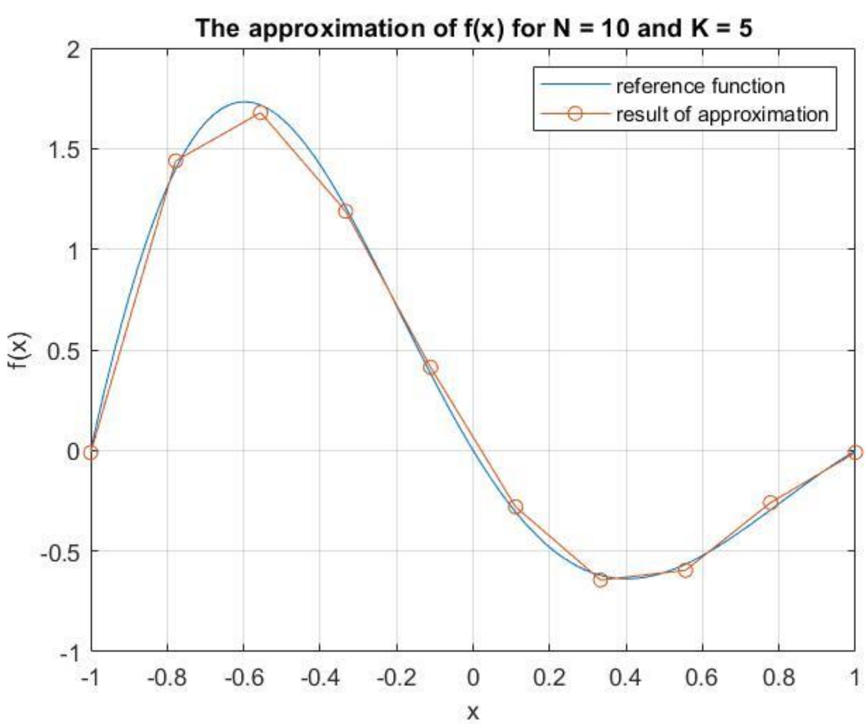

Figure 4. The approximation of f(x) for N = 10 and K = 5

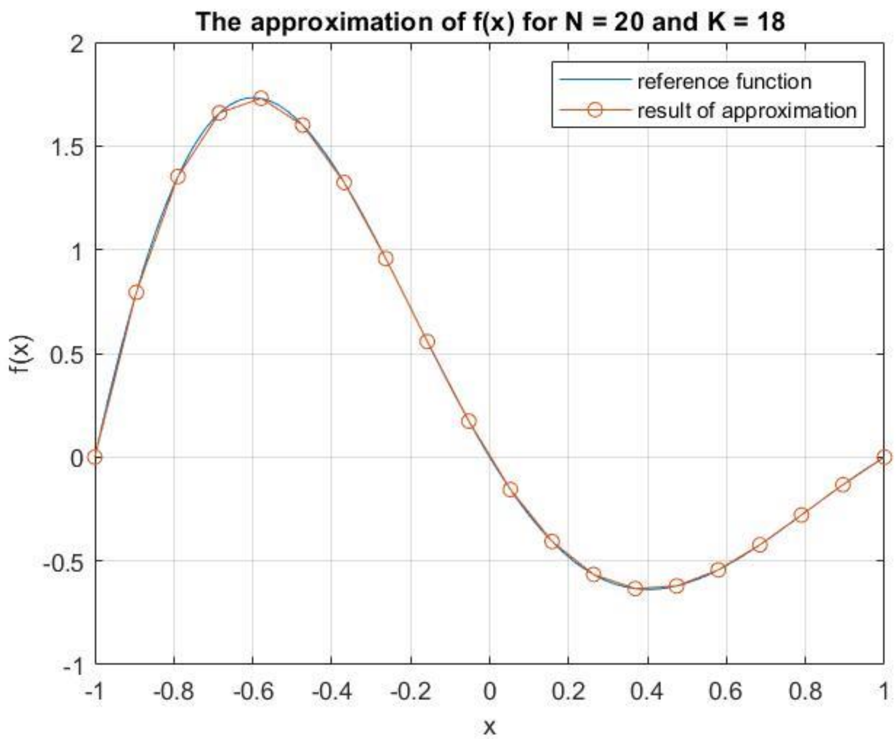

Figure 5. The approximation of f(x) for N = 20 and K = 15

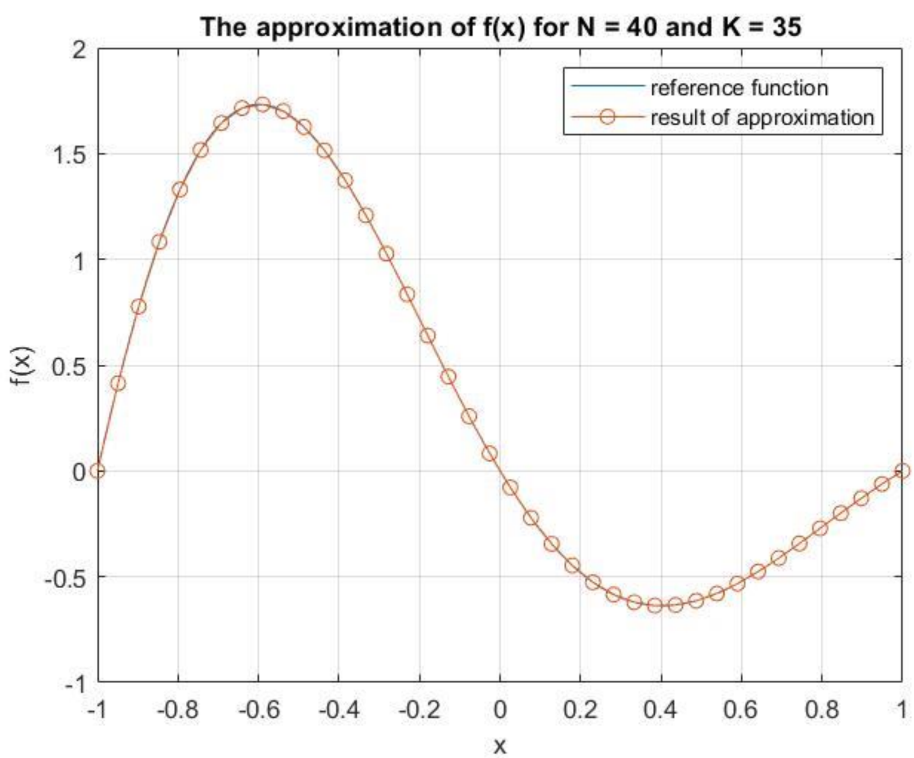

Figure 6. The approximation of f(x) for N = 40 and K = 35

c. Problem 3

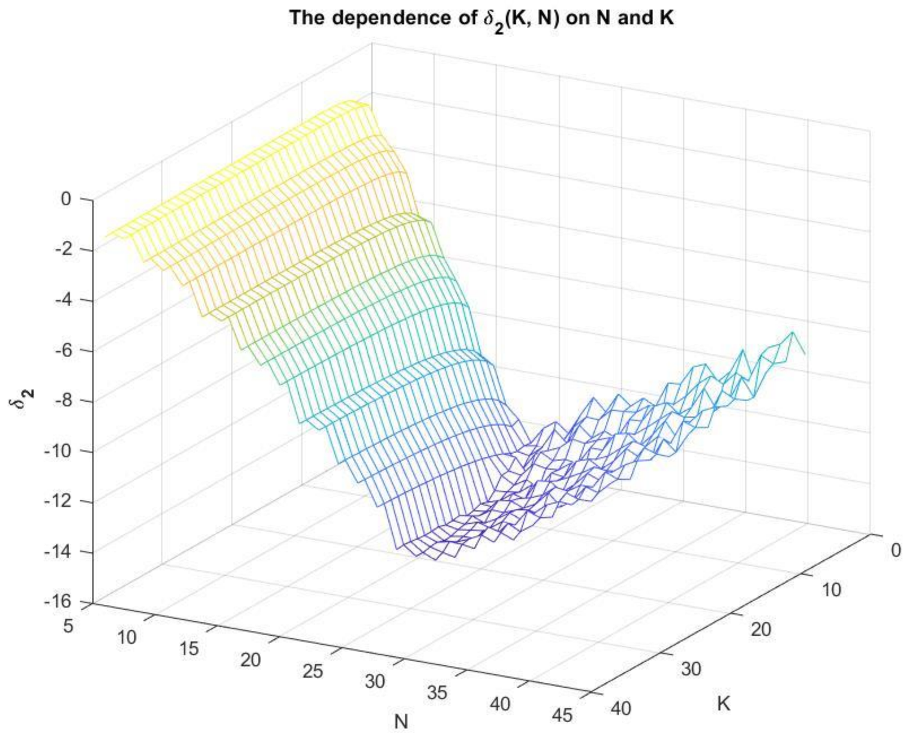

Figure 7. The dependence of $\delta_2(K, N)$ on N and K

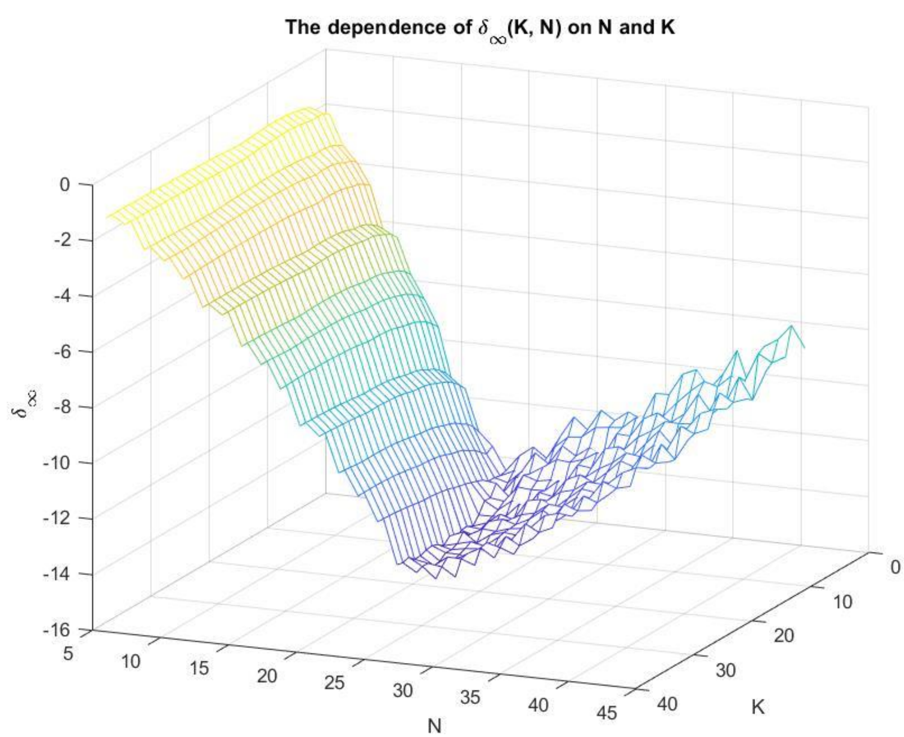

Figure 8. The dependence of $\delta_\infty(K, N)$ on N and K

d. Problem 4

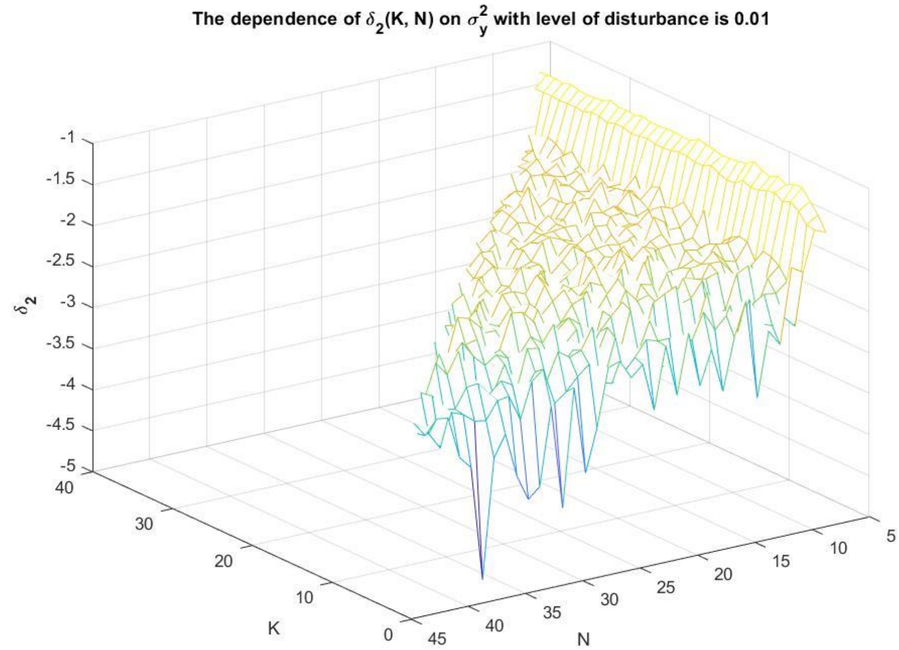

Figure 9. The dependence of $\delta_2(K,N)$ on $\sigma_y^2$with the level of disturbance is 0.01

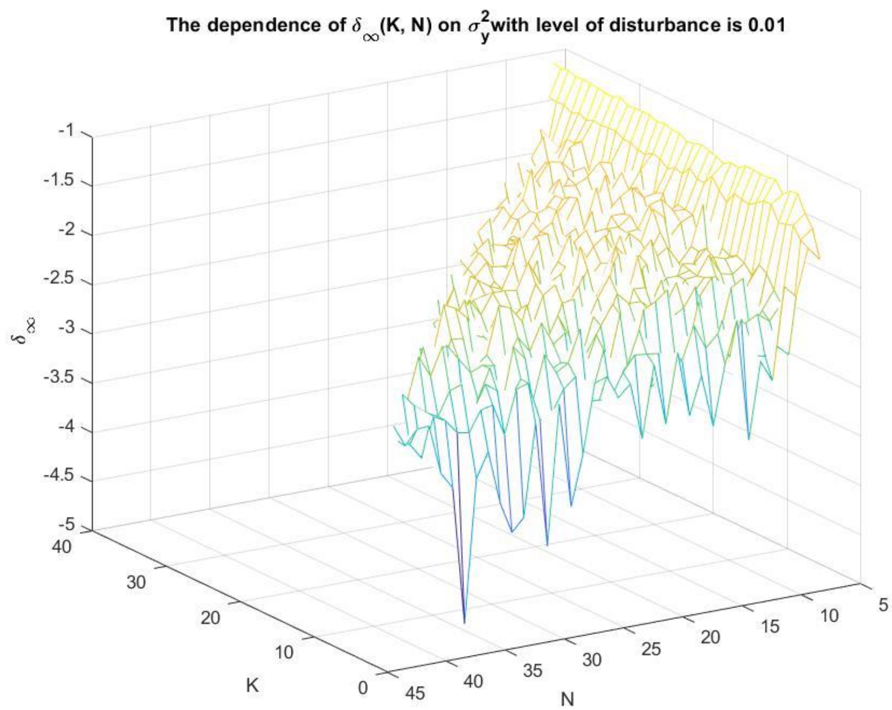

Figure 100. The dependence of $\delta_\infty(K,N)$ on $\sigma_y^2$with the level of disturbance is 0.01

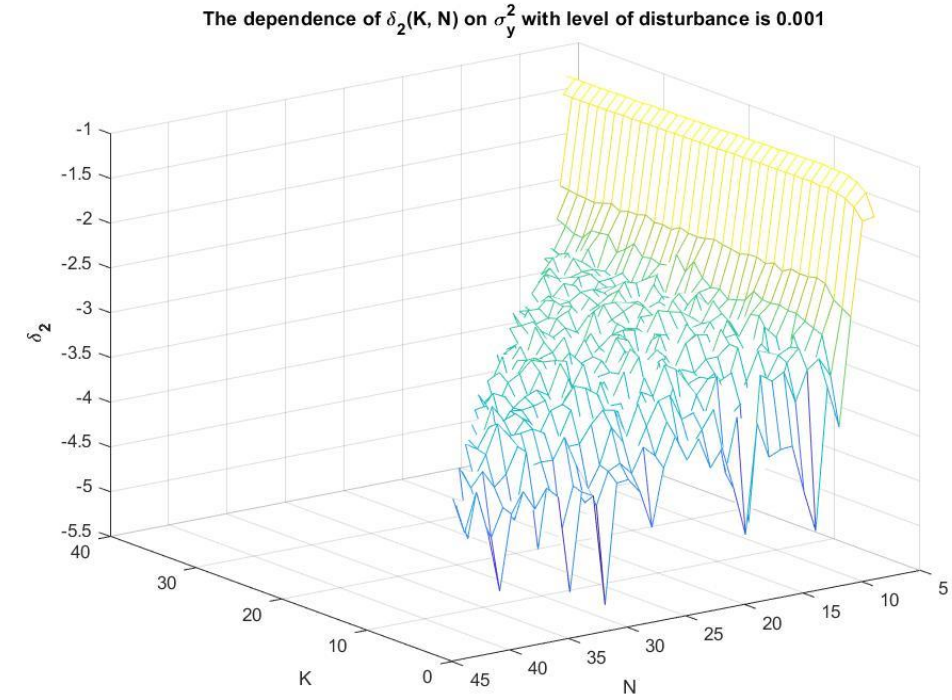

Figure 11. The dependence of $\delta_2(K,N)$ on $\sigma_y^2$with the level of disturbance is 0.001

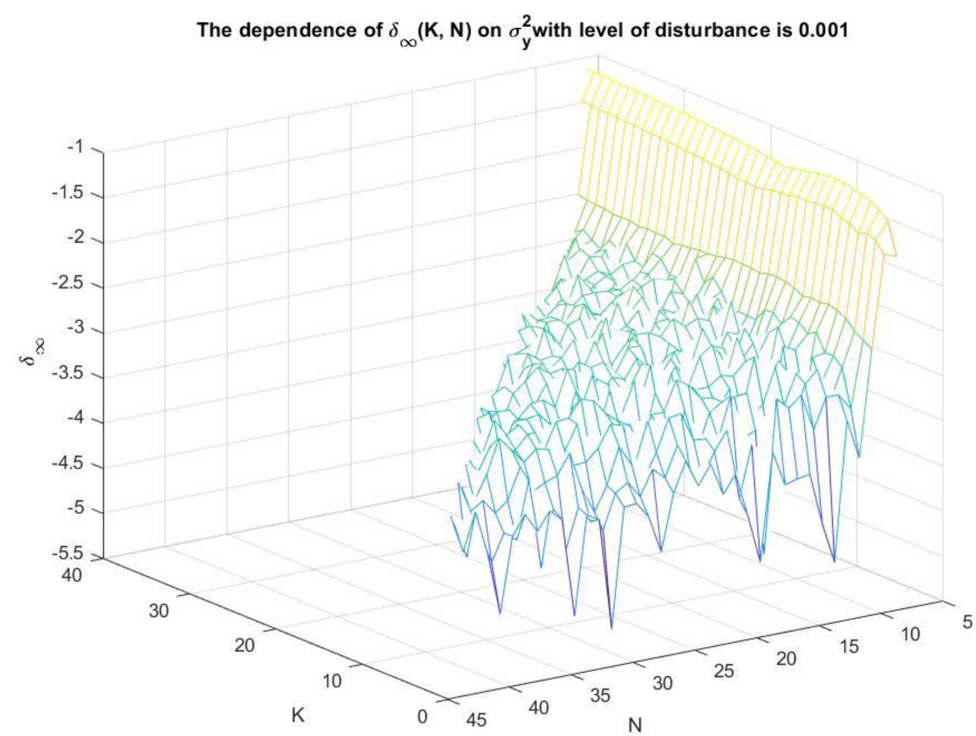

Figure 112. The dependence of $\delta_\infty(K,N)$ on $\sigma_y^2$with the level of disturbance is 0.001

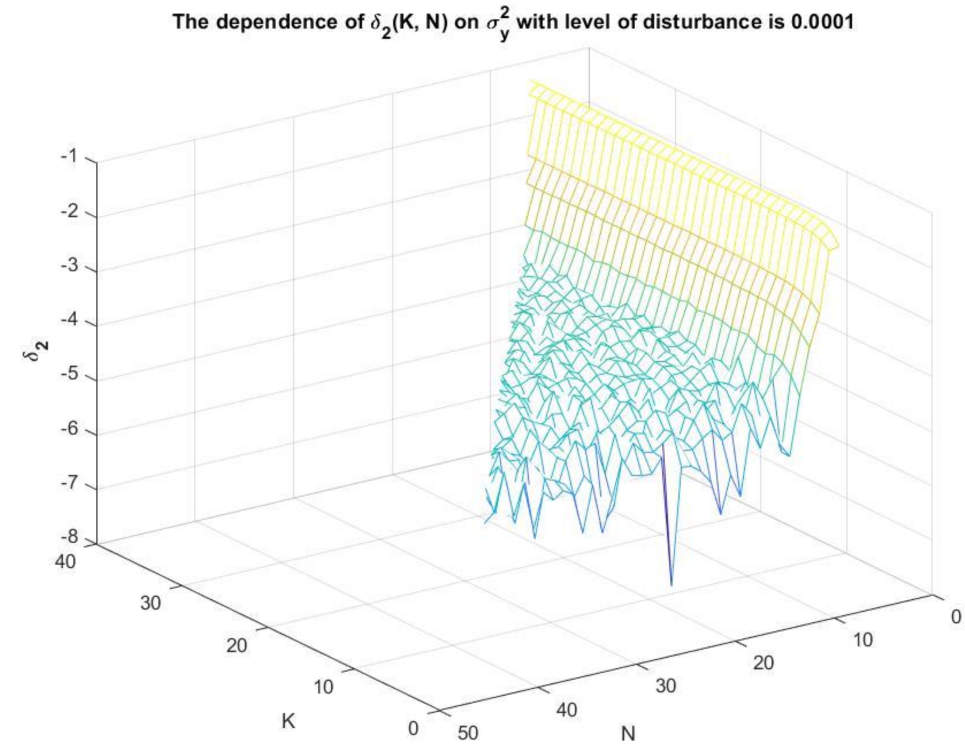

Figure 123. The dependence of $\delta_2(K, N)$ on $\sigma_y^2$with the level of disturbance is 0.0001

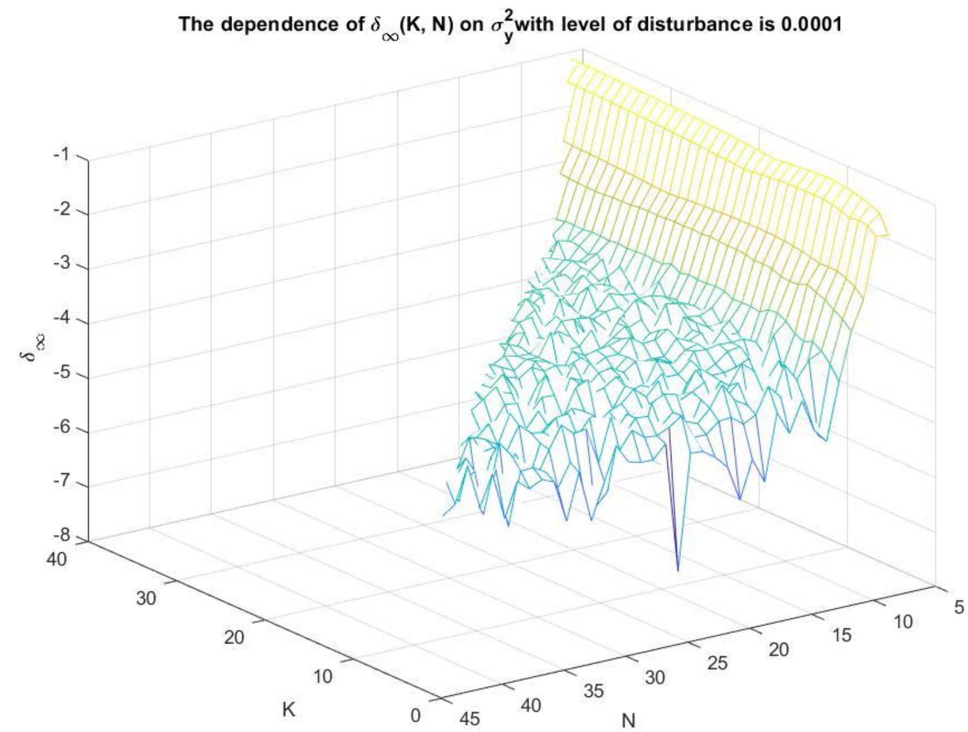

Figure 134. The dependence of $\delta_\infty(K, N)$ on $\sigma_y^2$with the level of disturbance is 0.0001

4. Conclusion

After using the method of least squares to approximate the function based on the discrete data, two numerical methods of solving the system of normal equations: Cholesky-Banachiewicz and MATLAB/Simulink built-in method, were used to solve and analyze the solutions of approximation. The obtained results are very close to each other. From three pairs of the values of the parameters N and K, it can be concluded that the higher N and K are, the more accurate the approximation is.

For the given function, with $x \in [-1, 1]$, both accuracy indicators $\delta_2(K, N)$ and $\delta_\infty(K, N)$ bottom at N around 25.

By analyzing the graphs of the dependence of $\delta_2(K, N)$ and $\delta_\infty(K, N)$ on σ_y^2, it can be seen that the lower the level of disturbance is, the more precise the approximation is.

List of references

R. Z. Morawski, Lecture notes for ENUME students

A. Miękina, ENUME MatLab Intro 2018

MathWorks, MATLAB Documentation, https://www.mathworks.com/help/index.html

MATLAB Code

```
clear all
close all
clc

f = @(x) -sin(pi*x).*exp(-x);
Ns = [5, 10, 15];
Ks = [2, 5, 8];
x1 = linspace(-1, 1, 100);

%%Problem 1
for i = 1 : 3
    N = Ns(i);
    y = f(x1);

    figure (i)
    plot(x1, y);
    hold on
    grid on
    [x, y] = createXY(N);
    plot(x, y, '*');
    xlabel('x');
    ylabel('y');
    title(['N = ', num2str(N)]);
end


Ns = [10, 20, 40];
Ks = [5, 18, 35];

%%Problem 2
for i = 1 : 3
    N = Ns(i);
    K = Ks(i);
    [x, y, fxLS, fxLSChol] = LSsolve(K, N);
    figure(i+3)
    plot(x1, f(x1), x, fxLS, '-o')
    legend('reference function', 'result of approximation')
    grid on
    xlabel('x')
    ylabel('f(x)')
    title(['The approximation of f(x) for N = ',
num2str(N), ' and K = ', num2str(K)]);
end
```

```
%Problem 3
%Dependence of indicator on N and K
K = [4:40];
N = [6:42];
accuracy = zeros(length(K), length(N));
accuracyInf = zeros(length(K), length(N));

for k = 4 : 40
    for n = k+2 : 42
        [x, y, fxLS, ] = LSsolve(k, n);
        accuracy(k-3, n-k-1) = norm(fxLS - y) / norm(y);
        accuracyInf(k-3, n-k-1) = norm(fxLS - y, Inf) /
norm(y, Inf);
    end
end

figure(7)
mesh(K, N, log10(accuracy))
xlabel('K');
ylabel('N');
zlabel('\bf \delta_{2}');
title('The dependence of \delta_{2}(K, N) on N and K');
grid on

figure(8)
mesh(K, N, log10(accuracyInf))
xlabel('K');
ylabel('N');
zlabel('\bf \delta_{\infty}');
title('The dependence of \delta_{\infty}(K, N) on N and
K');
grid on


%Problem 4
err = [0.01, 0.001, 0.0001];
accuracy = zeros(length(K), length(N));
accuracyInf = zeros(length(K), length(N));
variance = zeros(length(K), length(N));
for i = 1 : 3
    lvDisturbance = err(i);
    for k = 4 : 40
        for n = k+2 : 42
            [x, y, fxLS, ] = LSsolveErr(k, n,
lvDisturbance);
```

```
            accuracy(k-3, n-k-1) = norm(fxLS - y) /
norm(y);
            accuracyInf(k-3, n-k-1) = norm(fxLS - y, Inf) /
norm(y, Inf);
            variance(k-3, n-k-1) = var(y);
        end
    end

    figure(2*i + 7)
    mesh(K, N, log10(accuracy))
    xlabel('K');
    ylabel('N');
    zlabel('\bf \delta_{2}');
    title(['The dependence of \delta_{2}(K, N) on
\sigma_{y}^{2} with level of disturbance is ',
num2str(lvDisturbance)]);
    grid on

    figure(2*i + 8)
    mesh(K, N, log10(accuracyInf))
    xlabel('K');
    ylabel('N');
    zlabel('\bf \delta_{\infty}');
    title(['The dependence of \delta_{\infty}(K, N) on
\sigma_{y}^{2}with level of disturbance is ',
num2str(lvDisturbance)]);
    grid on
end

%%Function to solve approximation
function[x, y, fxLS, fxLSChol] = LSsolve(K, N)
    [x, y] = createXY(N);
    y = y';
    P = createBase(K, N, x);
    res = (P' * P) \ (P' * y);
    resCB = solveCB(P' * P, P' * y);
    fxLS = P * res;
    fxLSChol = P * resCB;
end


%%Function to solve approximation with error
function[x, y, fxLS, fxLSChol] = LSsolveErr(K, N, err)
    [x, y] = createXY(N);
    y = y';
    yErr = randn(N,1) * err;
```

```
    y = y .* (1 + yErr);
    P = createBase(K, N, x);
    res = (P' * P) \ (P' * y);
    resCB = solveCB(P' * P, P' * y);
    fxLS = P * res;
    fxLSChol = P * resCB;
end


%%Function to create data {(xn, yn)|n = 1, ..., N}---------
----
function [X, Y] = createXY(N)
    f = @(x) -sin(pi*x).*exp(-x);
    for n = 1 : N
        X(n) = -1 + 2*(n-1)/(N-1);
    end
    Y = f(X);
end


%%Function to creaatebase----------------------------------
-----
function P = createBase(K, N, x)
    for n = 1 : N
        P(n, 1) = 1;
        P(n, 2) = x(n);
        for j = 3 : K+1
            k = j-1;
            P(n, j) = (2*k-1)/k * x(n) * P(n, j-1) - (k-
1)/k * P(n, j-2);
        end
    end
end


%%Cholesky function----------------------------------------
------
function [L] = Cholesky(A)
    N = length(A);
    L = A-A;
    for i = 1  : N
        L(i, i) = sqrt( A(i, i) - L(i, :)*L(i, :)' );

        for j = (i + 1) : N
            L(j, i) = ( A(j, i) - L(i, :)*L(j, :)' )/L(i,
i);
```

```
        end
    end
end


%%Function to sovle linear system---------------------------
--------------
function [X] = solveCB(A, b)
    L = Cholesky(A);
    Lt = L';
    [n , ~] = size(A);
    y = zeros(n, 1);
    X = zeros(n, 1);

    y(1) = b(1)/L(1, 1);
    for i = 2 : n
        y(i) = (b(i) - L(i, :)*y)/L(i, i);
    end

    X(n) = y(n)/Lt(n, n);
    for i = n-1 : -1 : 1
        X(i) = (y(i) - Lt(i, :)*X)/L(i, i);
    end
end
```