Numerical Algorithms
Laboratory Report 3

Ill-posed least-squares problems

Submitted in partial fulfillment of
the requirements for the award of the degree of

Master of Science in
Advanced Applied Electronics

Submitted by

Roll No.	Name of Student
241194	Kamil Mężyński
226336	Konrad Dudziak

on 19th April 2021

Under supervision of
Rafał Zdunek, PhD

Wrocław University
of Science and Technology

Faculty of Advanced Applied Electronics

Summer Semester 2021

Contents

1 Laboratory tasks

1.1 Task 1

Find the solution that best approximates the system of inconsistent linear equations:

$$\text{(a)} \begin{cases} 3x_1 - x_2 = 4 \\ x_1 + 2x_2 = 0 \\ 2x_1 + x_2 = 1 \end{cases} \quad \text{(b)} \begin{cases} 3x_1 + x_2 + x_3 = 6 \\ 2x_1 + 3x_2 - x_3 = 1 \\ 2x_1 - x_2 + x_3 = 0 \\ 3x_1 - 3x_2 + 3x_3 = 8 \end{cases} \quad \text{(c)} \begin{cases} x_1 + x_2 - x_3 = 5 \\ 2x_1 - x_2 + 6x_3 = 1 \\ -x_1 + 4x_2 + x_3 = 0 \\ 3x_1 + 2x_2 - x_3 = 6 \end{cases}.$$

1.1.1 Analytical solution

a)

$$\begin{bmatrix} 3 & -1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 3 & 1 & 2 \\ -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 1 \\ 1 & 6 \end{bmatrix}$$

$$(A^T A)^{-1} = \left[\begin{array}{cc|cc} 14 & 1 & 1 & 0 \\ 1 & 6 & 0 & 1 \end{array}\right] \xrightarrow{R_1/14} \left[\begin{array}{cc|cc} 1 & \frac{1}{14} & \frac{1}{14} & 0 \\ 1 & 6 & 0 & 1 \end{array}\right] \xrightarrow{R_2 - R_1} \left[\begin{array}{cc|cc} 1 & \frac{1}{14} & \frac{1}{14} & 0 \\ 0 & \frac{83}{14} & -\frac{1}{14} & 1 \end{array}\right]$$
$$\xrightarrow{R_2/\frac{83}{14}} \left[\begin{array}{cc|cc} 1 & \frac{1}{14} & \frac{1}{14} & 0 \\ 0 & 1 & -\frac{1}{83} & \frac{14}{83} \end{array}\right] \xrightarrow{R_1 - \frac{R_2}{14}} \left[\begin{array}{cc|cc} 1 & 0 & \frac{6}{83} & -\frac{1}{83} \\ 0 & 1 & -\frac{1}{83} & \frac{14}{83} \end{array}\right]$$

$$(A^T A)^{-1} A^T = \begin{bmatrix} \frac{6}{83} & -\frac{1}{83} \\ -\frac{1}{83} & \frac{14}{83} \end{bmatrix} \begin{bmatrix} 3 & 1 & 2 \\ -1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} \frac{19}{83} & \frac{4}{83} & \frac{11}{83} \\ -\frac{17}{83} & \frac{27}{83} & \frac{12}{83} \end{bmatrix}$$

$$(A^T A)^{-1} A^T b = \begin{bmatrix} \frac{19}{83} & \frac{4}{83} & \frac{11}{83} \\ -\frac{17}{83} & \frac{27}{83} & \frac{12}{83} \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{87}{83} \\ -\frac{56}{83} \end{bmatrix}$$

The final solution:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{87}{83} \\ -\frac{56}{83} \end{bmatrix} \approx \begin{bmatrix} 1.0482 \\ -0.6747 \end{bmatrix}$$

b)

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & -1 \\ 2 & -1 & 1 \\ 3 & -3 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 1 \\ 0 \\ 8 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 3 & 2 & 2 & 3 \\ 1 & 3 & -1 & -3 \\ 1 & -1 & 1 & 3 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ 2 & 3 & -1 \\ 2 & -1 & 1 \\ 3 & -3 & 3 \end{bmatrix} = \begin{bmatrix} 26 & -2 & 12 \\ -2 & 20 & -12 \\ 12 & -12 & 12 \end{bmatrix}$$

$$(A^T A)^{-1} = \left[\begin{array}{ccc|ccc} 26 & -2 & 12 & 1 & 0 & 0 \\ -2 & 20 & -12 & 0 & 1 & 0 \\ 12 & -12 & 12 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_1/26, R_2/2, R_3/12} \left[\begin{array}{ccc|ccc} 1 & -\frac{1}{13} & \frac{6}{13} & \frac{1}{26} & 0 & 0 \\ -1 & 10 & -6 & 0 & \frac{1}{2} & 0 \\ 1 & -1 & 1 & 0 & 0 & \frac{1}{12} \end{array}\right]$$

$$\xrightarrow[R_3 - R_1]{R_2 + R_1} \left[\begin{array}{ccc|ccc} 1 & -\frac{1}{13} & \frac{6}{13} & \frac{1}{26} & 0 & 0 \\ 0 & \frac{129}{13} & -\frac{72}{13} & \frac{1}{26} & \frac{1}{2} & 0 \\ 0 & -\frac{12}{13} & \frac{7}{13} & -\frac{1}{26} & 0 & \frac{1}{12} \end{array}\right] \xrightarrow{R_2 * \frac{13}{129}} \left[\begin{array}{ccc|ccc} 1 & -\frac{1}{13} & \frac{6}{13} & \frac{1}{26} & 0 & 0 \\ 0 & 1 & -\frac{24}{43} & \frac{1}{258} & \frac{13}{258} & 0 \\ 0 & -\frac{12}{13} & \frac{7}{13} & -\frac{1}{26} & 0 & \frac{1}{12} \end{array}\right]$$

$$\xrightarrow[R_3 + \frac{12}{13} R_2]{R_1 + \frac{1}{13} R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & \frac{18}{43} & \frac{5}{129} & \frac{1}{258} & 0 \\ 0 & 1 & -\frac{24}{43} & \frac{1}{258} & \frac{13}{258} & 0 \\ 0 & 0 & \frac{1}{43} & -\frac{3}{86} & \frac{2}{43} & \frac{1}{12} \end{array}\right] \xrightarrow{R_3 * 43} \left[\begin{array}{ccc|ccc} 1 & 0 & \frac{18}{43} & \frac{5}{129} & \frac{1}{258} & 0 \\ 0 & 1 & -\frac{24}{43} & \frac{1}{258} & \frac{13}{258} & 0 \\ 0 & 0 & 1 & -\frac{3}{2} & 2 & \frac{43}{12} \end{array}\right]$$

$$\xrightarrow[R_2 + \frac{24}{43} R_3]{R_1 - \frac{18}{43} R_3} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{2}{3} & -\frac{5}{6} & -\frac{3}{2} \\ 0 & 1 & 0 & -\frac{5}{6} & \frac{7}{6} & 2 \\ 0 & 0 & 1 & -\frac{3}{2} & 2 & \frac{43}{12} \end{array}\right]$$

$$(A^T A)^{-1} A^T = \begin{bmatrix} \frac{2}{3} & -\frac{5}{6} & -\frac{3}{2} \\ -\frac{5}{6} & \frac{7}{6} & 2 \\ -\frac{3}{2} & 2 & \frac{43}{12} \end{bmatrix} \begin{bmatrix} 3 & 2 & 2 & 3 \\ 1 & 3 & -1 & -3 \\ 1 & -1 & 1 & 3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{6} & -\frac{5}{6} & 0 \\ \frac{13}{12} & -\frac{7}{12} & -\frac{17}{12} & \frac{1}{4} \end{bmatrix}$$

$$(A^T A)^{-1} A^T b = \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{6} & -\frac{5}{6} & 0 \\ \frac{13}{12} & -\frac{7}{12} & -\frac{17}{12} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} 6 \\ 1 \\ 0 \\ 8 \end{bmatrix} = \begin{bmatrix} -\frac{5}{3} \\ \frac{23}{6} \\ \frac{95}{12} \end{bmatrix}$$

The final solution:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{5}{3} \\ \frac{23}{6} \\ \frac{95}{12} \end{bmatrix} \approx \begin{bmatrix} -1.6667 \\ 3.8333 \\ 7.9167 \end{bmatrix}$$

c)

$$\begin{bmatrix} 1 & 1 & -1 \\ 2 & -1 & 6 \\ -1 & 4 & 1 \\ 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 0 \\ 6 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 2 & -1 & 3 \\ 1 & -1 & 4 & 2 \\ -1 & 6 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 2 & -1 & 6 \\ -1 & 4 & 1 \\ 3 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 15 & 1 & 7 \\ 1 & 22 & -5 \\ 7 & -5 & 39 \end{bmatrix}$$

$$(A^T A)^{-1} = \left[\begin{array}{ccc|ccc} 15 & 1 & 7 & 1 & 0 & 0 \\ 1 & 22 & -5 & 0 & 1 & 0 \\ 7 & -5 & 39 & 0 & 0 & 1 \end{array}\right] \xrightarrow{R_1/15} \left[\begin{array}{ccc|ccc} 1 & \frac{1}{15} & \frac{7}{15} & \frac{1}{15} & 0 & 0 \\ 1 & 22 & -5 & 0 & 1 & 0 \\ 7 & -5 & 39 & 0 & 0 & 1 \end{array}\right]$$

$$\xrightarrow[R_3-7R_1]{R_2-R_1} \left[\begin{array}{ccc|ccc} 1 & \frac{1}{15} & \frac{7}{15} & \frac{1}{15} & 0 & 0 \\ 0 & \frac{329}{15} & -\frac{82}{15} & -\frac{1}{15} & 1 & 0 \\ 0 & -\frac{82}{15} & \frac{536}{15} & -\frac{7}{15} & 0 & 1 \end{array}\right] \xrightarrow{R_2 * \frac{15}{329}} \left[\begin{array}{ccc|ccc} 1 & \frac{1}{15} & \frac{7}{15} & \frac{1}{15} & 0 & 0 \\ 0 & 1 & -\frac{89}{329} & -\frac{1}{15} & 1 & 0 \\ 0 & -\frac{82}{15} & \frac{536}{15} & -\frac{7}{15} & 0 & 1 \end{array}\right]$$

$$\xrightarrow[R_3+\frac{82}{15}R_2]{R_1-\frac{1}{15}R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & \frac{2392}{4935} & \frac{22}{329} & -\frac{1}{329} & 0 \\ 0 & 1 & -\frac{82}{15} & -\frac{1}{15} & 1 & 0 \\ 0 & 0 & \frac{169406}{4935} & -\frac{159}{329} & \frac{82}{329} & 1 \end{array}\right] \xrightarrow{R_3 * \frac{4935}{169406}} \left[\begin{array}{ccc|ccc} 1 & 0 & \frac{2392}{4935} & \frac{22}{329} & -\frac{1}{329} & 0 \\ 0 & 1 & -\frac{82}{15} & -\frac{1}{15} & 1 & 0 \\ 0 & 0 & 1 & -\frac{2385}{169046} & \frac{615}{84523} & \frac{4935}{169046} \end{array}\right]$$

$$\xrightarrow[R2+\frac{89}{329}R_3]{R_1-\frac{2392}{4935}R_3} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{6230}{84523} & -\frac{555}{84523} & -\frac{1196}{84523} \\ 0 & 1 & 0 & -\frac{1156}{169046} & \frac{4020}{84523} & \frac{1335}{169046} \\ 0 & 0 & 1 & -\frac{2385}{169046} & \frac{615}{84523} & \frac{4935}{169046} \end{array}\right]$$

$$(A^T A)^{-1} A^T = \begin{bmatrix} \frac{6230}{84523} & -\frac{555}{84523} & -\frac{1196}{84523} \\ -\frac{1156}{169046} & \frac{4020}{84523} & \frac{1335}{169046} \\ -\frac{2385}{169046} & \frac{615}{84523} & \frac{4935}{169046} \end{bmatrix} \begin{bmatrix} 1 & 2 & -2 & 3 \\ 1 & -1 & 4 & 2 \\ -1 & 6 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{6871}{84523} & \frac{5839}{84523} & -\frac{9646}{84523} & \frac{18776}{84523} \\ \frac{5549}{169046} & -\frac{1171}{84523} & \frac{34651}{169046} & \frac{11277}{169046} \\ -\frac{3045}{84523} & \frac{11805}{84523} & \frac{6120}{84523} & -\frac{4815}{84523} \end{bmatrix}$$

$$(A^T A)^{-1} A^T b = \begin{bmatrix} \frac{6871}{84523} & \frac{5839}{84523} & -\frac{9646}{84523} & \frac{18776}{84523} \\ \frac{5549}{169046} & -\frac{1171}{84523} & \frac{34651}{169046} & \frac{11277}{169046} \\ -\frac{3045}{84523} & \frac{11805}{84523} & \frac{6120}{84523} & -\frac{4815}{84523} \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 0 \\ 6 \end{bmatrix} = \begin{bmatrix} \frac{152850}{84523} \\ \frac{93065}{169046} \\ -\frac{32310}{84523} \end{bmatrix}$$

The final solution:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{152850}{84523} \\ \frac{93065}{169046} \\ -\frac{32310}{84523} \end{bmatrix} \approx \begin{bmatrix} 1.8084 \\ 0.5505 \\ -0.3823 \end{bmatrix}$$

1.1.2 MATLAB solution

```
% (a)
A = [3 -1; 1 2; 2 1];                    b = [4; 0; 1];
% (b)
% A = [3 1 1; 2 3 -1; 2 -1 1; 3 -3 3];    b = [6; 1; 0; 8];
% (c)
% A = [1 1 -1; 2 -1 6; -1 4 1; 3 2 -1];   b = [5; 1; 0; 6];

A \ b
inv(A'*A)*A'*b
```

Listing 1: main.m - task 1 source code

1.2 Task 2

Find the least squares approximating function of the form $a_0 + a_1x^2 + a_2\sin(\frac{\pi x}{2})$ for each of the following sets of data pairs:

$$a)(0,3),(1,0),(1,-1),(-1,2) \quad b)(-1,0.5),(0,1),(2,5),(3,9)$$

$$A = \begin{bmatrix} 1 & x_1^2 & \sin(\frac{\pi x_1}{2}) \\ 1 & x_2^2 & \sin(\frac{\pi x_2}{2}) \\ 1 & x_3^2 & \sin(\frac{\pi x_3}{2}) \\ 1 & x_4^2 & \sin(\frac{\pi x_4}{2}) \end{bmatrix}$$

$$A \cdot a = b$$

Data set a) transformed to matrix form:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}, x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}, a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 0 \\ -1 \\ 2 \end{bmatrix}$$

Data set b) transformed to matrix form:

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 4 & 0 \\ 1 & 9 & -1 \end{bmatrix}, x = \begin{bmatrix} -1 \\ 0 \\ 2 \\ 3 \end{bmatrix}, a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, b = \begin{bmatrix} 0.5 \\ 1 \\ 5 \\ 9 \end{bmatrix}$$

1.2.1 MATLAB solution

```
%(a)
x = [0 1 1 -1];
y = [3 0 -1 2];
%(b)
% x = [-1 0 2 3];
% y = [0.5 1 5 9];

A = zeros(length(y), 3);
for i = 1:length(y)
    A(i,:) = [ 1 x(i)^2 sin(x(i)*pi/2) ];
end
b = y';

res_bi = A \ b
% classical LS fitting
res_LSfit = inv(A'*A)*A'*b

figure(1);
plot(x, y, 'ro');
hold on;

x_plot = min(x):0.01:max(x);
y_plot = res_LSfit(1) + res_LSfit(2)*x_plot.^2 + res_LSfit(3)*sin(x_plot*pi/2);
plot(x_plot, y_plot, 'k');
```

Listing 2: main.m - task 2 source code

In both cases the matlab built-in function A\b, classical LS fitting, Iterative Tikhonov Regularization, SVD and TSVD algorithms were used. Each algorithm returned the same result.

$$a = \begin{bmatrix} 0.9000 \\ 1.0500 \\ 1.4000 \end{bmatrix}$$

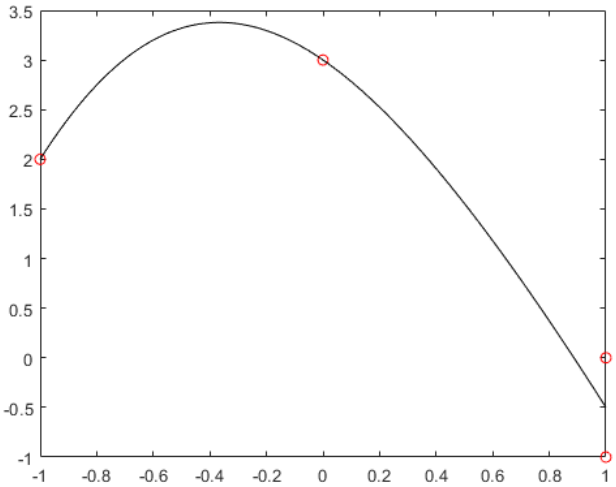

Figure 1: Approximate function - task (a)

$$a = \begin{bmatrix} 3.0000 \\ -2.2500 \\ -1.2500 \end{bmatrix}$$

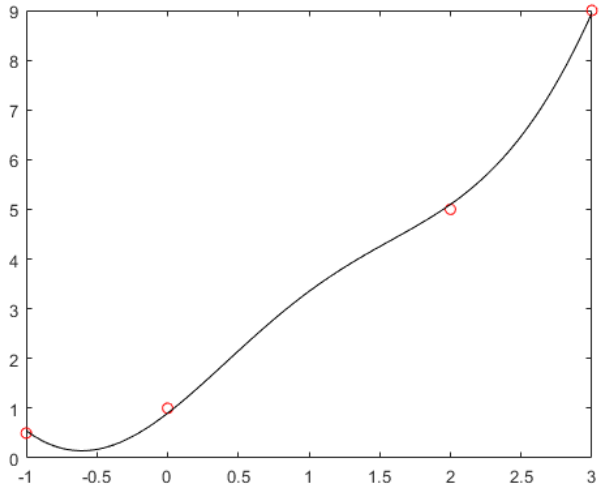

Figure 2: Approximate function - task (b)

1.3 Task 3

The yield y of wheat in quintals per hectare appears to be a linear function of the number of days x_1 of sunshine, the number of centimeters x_2 of rainfall, and the number of kilograms x_3 of fertilizer per hectare. Find the best fit to the data in the table with an equation in the form: $y = a_0 + a_1x_1 + a_2x_2 + a_3x_3$.

y	x_1	x_2	x_3
28	50	18	10
30	40	20	16
21	35	14	10
23	40	12	12
23	30	16	14

1.3.1 MATLAB solution

```
46  x1 = [50 40 35 40 30];
47  x2 = [18 20 14 12 16];
48  x3 = [10 16 10 12 14];
49  y = [28 30 21 23 23];
50
51  A = zeros(length(y), 4);
52  for i = 1:length(y)
53      A(i,:) = [ 1 x1(i) x2(i) x3(i) ];
54  end
55  b = y';
56
57  res_bi = A \ b
58  % classical LS fitting
59  res_LSfit = inv(A'*A)*A'*b
```

Listing 3: main.m - task 3 source code

The problem was transformed to form of matrix equation:

$$\begin{bmatrix} 1 & 50 & 18 & 10 \\ 1 & 40 & 20 & 16 \\ 1 & 35 & 14 & 10 \\ 1 & 40 & 12 & 12 \\ 1 & 30 & 16 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 28 \\ 30 \\ 21 \\ 23 \\ 23 \end{bmatrix}$$

$$A \cdot a = b$$

The matlab built-in function A\b, classical LS fitting, Iterative Tikhonov Regularization, SVD and TSVD algorithms were used. Each algorithm returned the same result:

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -5.1890 \\ 0.3384 \\ 0.5128 \\ 0.7085 \end{bmatrix}$$

1.4 Task 4

Using the least square find the "best" straight-line fit and the error estimates for the slope and intercept of that line for the following set of data:

x_i	1	2	3	4	5	6	7	8
y_i	1.5	2.0	2.8	4.1	4.9	6.3	5.0	11.5

1.4.1 MATLAB solution

```
x = [1 2 3 4 5 6 7 8];
y = [1.5 2 2.8 4.1 4.9 6.3 5.0 11.5];

A = zeros(length(y), 2);
for i = 1:length(y)
    A(i,:) = [ 1 x(i) ];
end
b = y';

res_bi = A \ b
% classical LS fitting
res_LSfit = inv(A'*A)*A'*b

figure(1);
plot(x, y, 'ro');
hold on;

x_plot = 0:0.01:max(x);
y_plot = res_LSfit(1) + res_LSfit(2)*x_plot;
error = sqrt(sum((y - mean(y)).^2) / (length(y)-2)) / sqrt(sum((x-mean(x)).^2))
plot(x_plot, y_plot, 'k');
```

Listing 4: main.m - task 4 source code

To compute the coefficients the LS method with interactive Tikhinov regularization was used. The algorithm is using following formula:

$$x_{LS} = (A^T A + \alpha^2 L^T L)^{-1} A^T b$$

The data in matrix form:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 2.0 \\ 2.8 \\ 4.1 \\ 4.9 \\ 6.3 \\ 5.0 \\ 11.5 \end{bmatrix}$$

$$A \cdot a = b$$

The algorithms returned the same results:

$$a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} -0.3964 \\ 1.1464 \end{bmatrix}$$

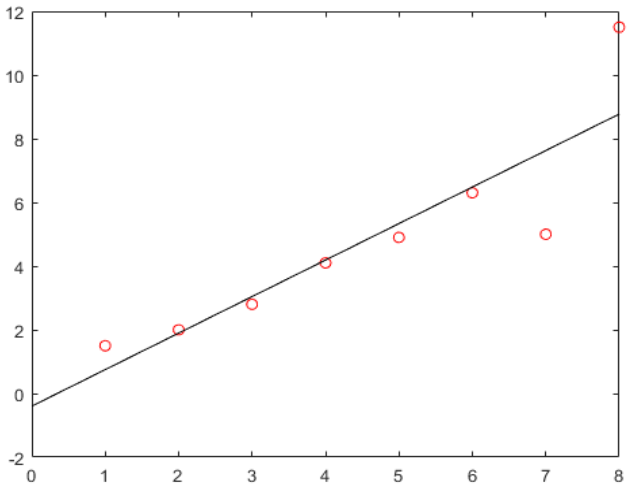

Figure 3: Approximate function

1.5 Task 5

A missile is fired from enemy territory, and its position in flight is observed by radar tracking devices at the following positions:

x_i[km]	0	250	500	750	1000
y_i[km]	0	8	15	19	20

Suppose our intelligence sources indicate that enemy missiles are programmed to follow a parabolic flight path. Predict how far down the range the missile will land.

1.5.1 MATLAB solution

```
x = [0 250 500 750 1000];
y = [0 8 15 19 20];

A = zeros(length(y), 3);
for i = 1:length(y)
    A(i,:) = [ 1 x(i) x(i).^2 ];
end
b = y';

res_bi = A \ b
% classical LS fitting
res_LSfit = inv(A'*A)*A'*b

figure(1);
plot(x, y, 'ro');
hold on;

x_plot = min(x):0.01:(max(x)+1050);
y_plot = res_LSfit(1) + res_LSfit(2)*x_plot + res_LSfit(3)*x_plot.^2;
plot(x_plot, y_plot, 'k');
```

Listing 5: main.m - task 5 source code

The data was transformed to the matrix form, next the 2nd order polynomial was fitted into data set. Than the 2nd root was found.

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 250 & 62500 \\ 1 & 500 & 250000 \\ 1 & 750 & 562500 \\ 1 & 1000 & 1000000 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \\ 15 \\ 19 \\ 20 \end{bmatrix}$$

The predicted missile range is 2044.2 km.

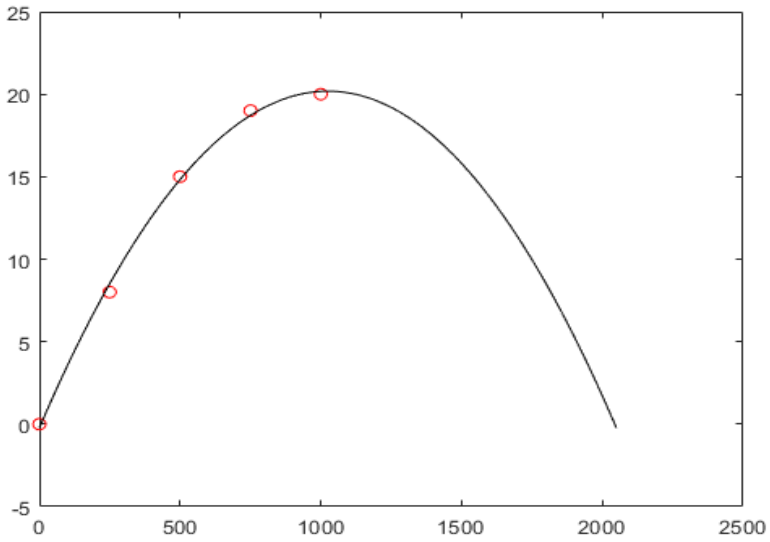

Figure 4: Approximate function

1.6 Task 6

Using least squares techniques, fit the following data:

x	-5	-4	-3	-2	-1	0	1	2	3	4	5
y	2	7	9	12	13	14	14	13	10	8	4

with a line $y = a_0 + a_1 x$ and than fit the data with a quadratic $y = a_0 + a_1 x + a_2 x^2$. Determine which of these two curves best fits the data by computing the l_2 norm of the errors in each case.

1.6.1 Linear equation

$$A \cdot a = b$$

$$\begin{bmatrix} 1 & -5 \\ 1 & -4 \\ 1 & -3 \\ 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 9 \\ 12 \\ 13 \\ 14 \\ 14 \\ 13 \\ 10 \\ 8 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 9.6364 \\ 0.1818 \end{bmatrix}$$

The l_2 norm error was estimated using following formula:

$$err = \sqrt{\sum_{i=1}^{n} (y_i - f(x_i))^2} \approx 12.7636$$

1.6.2 Quadratic equation

$$A \cdot a = b$$

$$\begin{bmatrix} 1 & -5 & 25 \\ 1 & -4 & 16 \\ 1 & -3 & 9 \\ 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 9 \\ 12 \\ 13 \\ 14 \\ 14 \\ 13 \\ 10 \\ 8 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 13.9720 \\ 0.1818 \\ -0.4336 \end{bmatrix}$$

The l_2 norm error was estimated using following formula:

$$err = \sqrt{\sum_{i=1}^{n} (y_i - f(x_i))^2} \approx 1.2737$$

1.6.3 MATLAB solution

```
117 x = [-5 -4 -3 -2 -1 0 1 2 3 4 5];
118 y = [2 7 9 12 13 14 14 13 10 8 4];
119
120 A = zeros(length(y), 2);
121 A2 = zeros(length(y), 3);
122 for i = 1:length(y)
123     A(i,:) = [ 1 x(i) ];
124     A2(i,:) = [ 1 x(i) x(i).^2 ];
125 end
126 b = y';
127
128 res_bi = A \ b
129 res_bi2 = A2 \ b
130 % classical LS fitting
131 res_LSfit = inv(A'*A)*A'*b
132 res_LSfit2 = inv(A2'*A2)*A2'*b
133
134 figure(1);
135 plot(x, y, 'ro');
136 hold on;
137
138 x_plot = min(x):0.01:max(x);
139 y_plot = res_LSfit(1) + res_LSfit(2)*x_plot;
140 error = zeros(length(y), 1);
141
142 y_plot2 = res_LSfit2(1) + res_LSfit2(2)*x_plot + res_LSfit2(3)*x_plot.^2;
143 error2 = zeros(length(y), 1);
144 for i = 1:length(y)
145     error(i) = y(i) - (res_LSfit(1) + res_LSfit(2)*x(i));
146     error2(i) = y(i) - (res_LSfit2(1) + res_LSfit2(2)*x(i) + res_LSfit2(3)*x(i)^2);
147 end
148 error = norm(error)
149 error2 = norm(error2)
150 plot(x_plot, y_plot, 'k', x_plot, y_plot2, 'b');
```

Listing 6: main.m - task 6 source code

1.6.4 Conclusions

The approximation is much better using quadratic function. It can be observed not only by comparing the errors but also by looking on the plot.

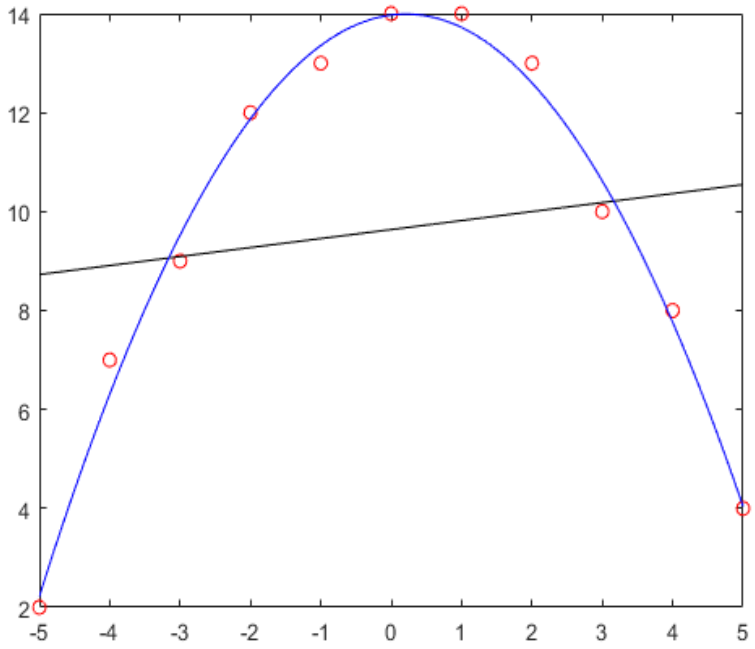

Figure 5: Approximate functions

1.7 Task 7

Fit the cosine polynomial $g(x) = \sum_{j=0}^{n} c_j cosjx$ to the function $f(x) = \pi^2 - x^2$ to minimize the error $\|f(x) - g(x)\|_2$ in the range $[0, \pi]$. Estimate the error for n = 1, 2, 3, ... , 10.

1.7.1 MATLAB solution

```
x = 0:0.01:pi;
n = 10;
f = pi^2 - x.^2;

for i = 1:n
    A = [ cos(0*x') cos((1:i).*x') ];
    c = inv(A'*A)*A'*f';
    g = A*c;

    error(i) = norm(f' - g);

    figure(i);
    plot(x, f, 'k');
    hold on;
    plot(x, g, 'r');
    title(['n=', num2str(i)]);
    legend('f(x)=\pi^2-x^2','g(x)=\Sigma c_i cos(ix)');
end
error
```

Listing 7: main.m - task 7 source code

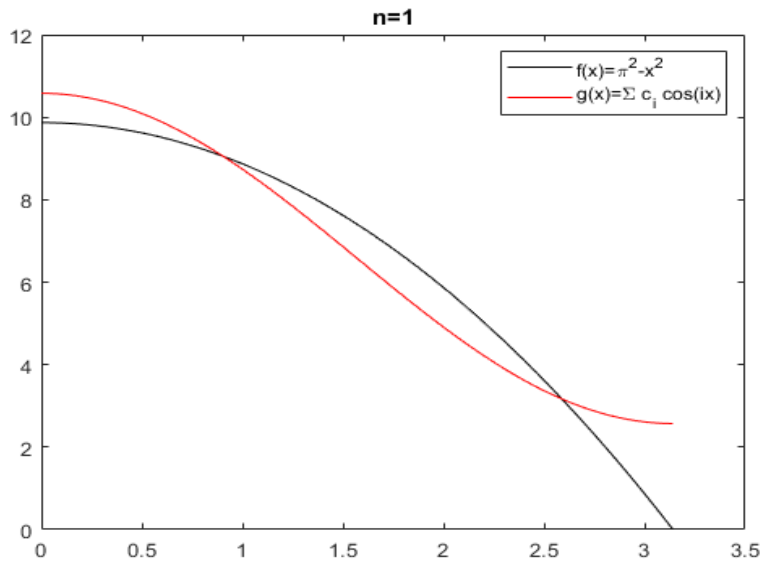

Figure 6: Function and its approximation for n=1

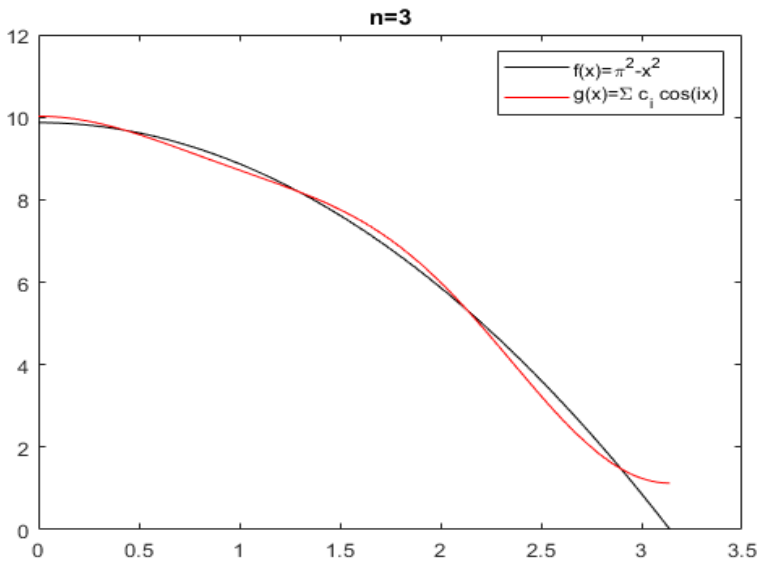

Figure 7: Function and its approximation for n=3

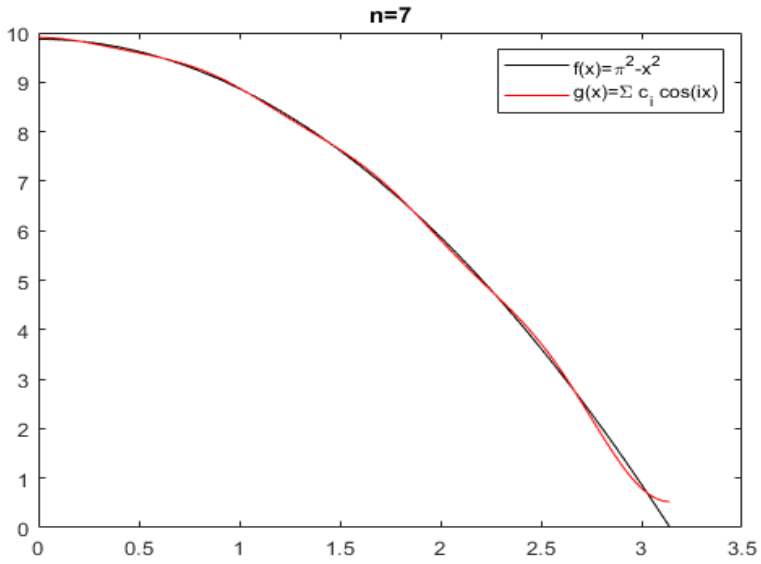

Figure 8: Function and its approximation for n=7

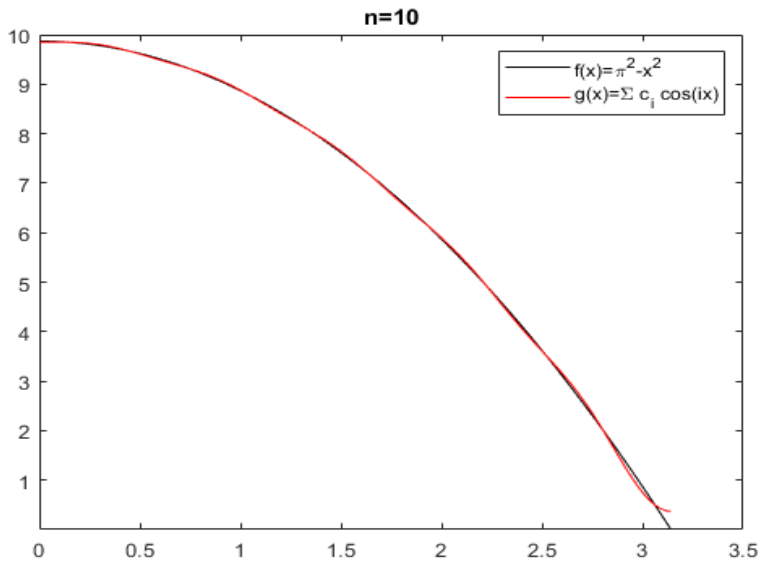

Figure 9: Function and its approximation for n=10

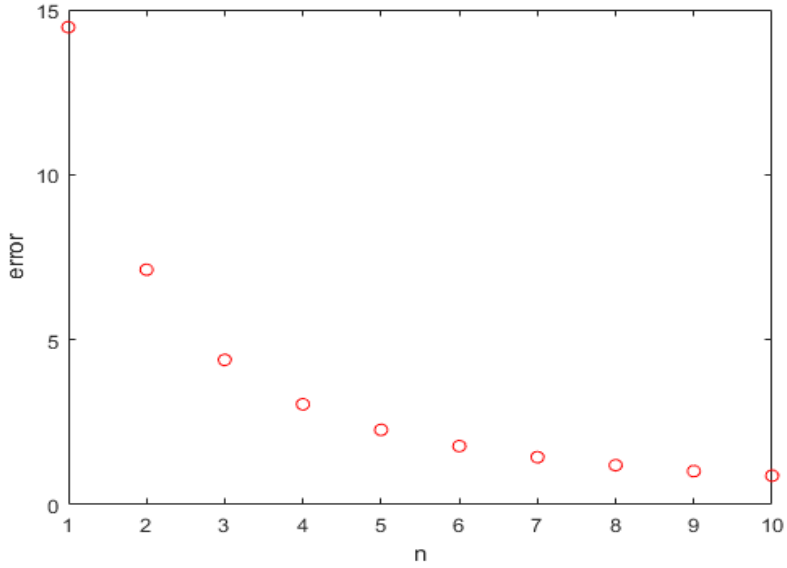

Figure 10: The error vs n

1.7.2 Conclusions

The greater the n the smaller the error was.

1.8 Task 8

Let $f(x) = \sin(\pi x)$ on $[0,1]$. The object of this problem is to determine the coefficients α_i of the cubic polynomial $p(x) = \sum i = 1^3 a_i x^i$ that is as close to $f(x)$ as possible in the sense that $r = \int_0^1 |f(x) - p(x)|^2 dx$ is as small as possible.

1.8.1 MATLAB solution

```
178 x = 0:0.01:1;
179 f = sin(pi*x);
180
181 A = [x' (x.^2)' (x.^3)'];
182 a = inv(A'*A)*A'*f';
183 p = A*a;
184
185 % estimate error
186 error = trapz(abs((f'-p).^2));
187
188 figure(1);
189 plot(x, f, 'k');
190 hold on;
191 plot(x, p, 'r');
192 legend('f(x)=sin(\pi x)', 'p(x)=\Sigma a_i x^i');
```

Listing 8: main.m - task 8 source code

To obtain the solutions were obtained by utilizing following code. Than SVD algorithm were used to check the correctness of the calculations. The obatined results:

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 3.7533 \\ -3.4025 \\ -0.4093 \end{bmatrix}$$

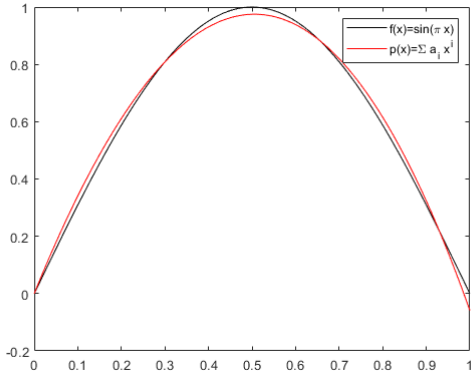

Figure 11: The function and its approximation

Error calculation:

$$error = \int_0^1 |f(x) - p(x)|^2 dx = 0.0460$$

1.9 Task 9

For the matrix $A = \begin{bmatrix} -4 & -2 & -4 & -2 \\ 2 & -2 & 2 & 1 \\ -4 & 1 & -4 & -2 \end{bmatrix}$ and $b = \begin{bmatrix} -12 & 3 & -9 \end{bmatrix}^T$,

- find the solution of $Ax = b$ that has the minimum Euclidean norm,
- estimate rank(A), cond(A), A^+,
- compute orthogonal projectors onto each of the four fundamental subspaces associated with A,
- find the point in $N(A)^\perp$ that is closest to b.

1.9.1 Analytical solution

The following formulas were used:
Pseudoinverse:

$$A^+ = V \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^H$$

Orthogonal projectors:

$$P_{R(A)} = AA^+ = U_1 U_1^H$$
$$P_{N(A^H)} = I_M - AA^+ = U_2 U_2^H$$
$$P_{R(A^H)} = A^+A = V_1 V_1^H$$
$$P_{N(A)} = I_N - A^+A = V_2 V_2^H$$

1.9.2 MATLAB solution

Using Tikhonov Regularization:

$$x = \begin{bmatrix} 1.1111 \\ 1.0000 \\ 1.1111 \\ 0.5556 \end{bmatrix}$$

rank(A) = 2
$\text{cond}(A) = \frac{\sigma_{MAX}}{\sigma_{MIN}} = \frac{\sigma_{MAX}}{0} = \infty$

$$A^+ = \begin{bmatrix} 0.0048 & -0.0240 & 0.0480 \\ -0.2222 & -0.2222 & 0.1111 \\ -0.2046 & 0.1023 & -0.2046 \\ 0.0911 & -0.0455 & 0.0911 \end{bmatrix}$$

$$P_{R(A)} = \begin{bmatrix} 0.8889 & 0.22220.2222 & \\ 0.2222 & 0.5556 & -0.4444 \\ 0.2222 & -0.4444 & 0.5556 \end{bmatrix}$$

$$P_{R(A^H)} = \begin{bmatrix} 0.4444 & 0 & 0.4444 & 0.2222 \\ 0 & 1.0000 & 0 & 0 \\ 0.4444 & 0 & 0.4444 & 0.2222 \\ 0.2222 & 0 & 0.2222 & 0.1111 \end{bmatrix}$$

$$P_{N(A^H)} = \begin{bmatrix} 0.1979 & -0.0989 & 0.1979 \\ -0.0989 & 0.0495 & -0.0989 \\ 0.1979 & 0.0989 & 0.1979 \end{bmatrix} \cdot 10^{-15}$$

$$P_{N(A)} = \begin{bmatrix} 0.5556 & 0 - 0.4444 & -0.2222 & \\ 0 & 0 & 0 & 0 \\ -0.4444 & 0 & 0.5556 & -0.2222 \\ -0.2222 & 0 & -0.2222 & 0.8889 \end{bmatrix}$$

```
A = [-4 -2 -4 -2; 2 -2 2 1; -4 1 -4 -2];
b = [-12; 3; -9];
cond(A)
null(A)

format short
[U, sigma, V] = svd(A)

% LS solution using SVD
res_LSSVD = singValDecomp(A, b)
% LS solution using TSVD
res_LSTSVD = truncSingValDecomp(A, b)

% Iterative Tikhonov Regularization
[x_LS, error] = iterTikhReg(A, b)

% four subspaces
r = rank(A);
[m, n] = size(A);
u1 = U(:, 1:r);     % columnspace
u2 = U(:, r+1:m);   % nullspace for c-space
v1 = V(:, 1:r);     % rowspace
v2 = V(:, r+1:n);   % nullspace for r-space

% orthagonal projectors
P_r = u1*u1'
P_rh = v1*v1'
P_nh = u2*u2'
P_n = v2*v2'
```

Listing 9: main.m - task 9 source code

1.10 Task 10

For the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$ and the exact solution $x_* = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$,

- determine the data vector for the model Ax = b, than estimate the minimal norm LS solution with the selected regularization methods,
- estimate rank(A), cond(A), A^+,
- find the N(A), and the missing components that went onto N(A),
- estimate the solution error $||x - x_*||_2$ and the residual error $||b - Ax||_2$.

$$b = Ax = \begin{bmatrix} 14 \\ 32 \\ 50 \\ 68 \\ 86 \end{bmatrix}$$

1.10.1 MATLAB solution

Using Tikhonov Regularization with regularization parameter set to 0.0000001:

$$x = \begin{bmatrix} 1.000000440515578 \\ 1.999998299404979 \\ 3.000000332947820 \end{bmatrix}$$

rank(A) = 2 cond(A) = $\frac{\sigma_{MAX}}{\sigma_{MIN}} = \frac{\sigma_{MAX}}{0} = \infty$

$$A^+ = \begin{bmatrix} -1.6329 & -0.9835 & -0.3956 & 0.2415 & 0.8294 \\ 2.4658 & 1.4669 & 0.5912 & -0.3830 & -1.2587 \\ -0.8996 & -0.5168 & -0.1956 & 0.1748 & 0.4960 \end{bmatrix}$$

$$N(A) = \begin{bmatrix} 0.7454 & 0.0000 \\ -0.0000 & -0.0000 \\ -0.5963 & -0.4472 \\ -0.2981 & 0.8944 \end{bmatrix}$$

$||x - x_*|| = 1.7880e - 06$
$||b - Ax||_2 = 1.8986e - 05$

```
A = [ 1 2 3; 4 5 6; 7 8 9; 10 11 12; 13 14 15];
xeq = [1; 2; 3];

% LS fitting with tikhonov regularization
b = A*xeq
[m,n] = size(A);
format long
if(m>n && rank(A) == n)
    x = inv(A'*A)*A'*b;
else
    alpha = 0.0000001;
    display("Thikonov")
    x = inv(A'*A + alpha*eye(n)'*eye(n))*A'*b;
end

rank(A)
cond(A)
inv(A'*A) * A'
A' * inv(A*A')
norm(x-xeq)
norm(b-A*x)
norm(A)

% Iterative Tikhonov Regularization
[x_LS, error] = iterTikhReg(A, b)

% LS solution using SVD
res_LSSVD = singValDecomp(A, b)

format short
```

Listing 10: main.m - task 10 source code

1.11 Task 11

Generate the values of the polynomial $y = 40+10x+5x^2+3x^3+2x^4+x^5+x^6$ for x = 1,2, ... , 14. First fit the polynomial $y = a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5+a_6x^6$ to the generated data in the LS sense and compare the estimated coefficients $a_0, a_1, ..., a_6$ to the true ones. Then perturb the observed data with an additive Gaussian noise $N(0, \sigma^2)$,and illustrate the fitting error (the Euclidean norm) versus the standard deviation σ. Estimate the maximal value of σ for which the fitting error exceeds 10^{-6}, assuming double precision arithmetic operations.

Obtained results:

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \approx \begin{bmatrix} 39.999959622102324 \\ 10.000030882627470 \\ 4.999995587932062 \\ 3.000002984546882 \\ 1.999999183327873 \\ 1.000000097157852 \\ 1.000000001077982 \end{bmatrix} \tag{1}$$

1.11.1 MATLAB solution

```
format long
x = 1:14;
a = [40; 10; 5; 3; 2; 1; 1];
b = (40 + 10*x + 5*x.^2 + 3*x.^3 + 2*x.^4 + x.^5 + x.^6)';

A = zeros(length(x), 7);
for i = 1:length(x)
    A(i,:) = [ 1 x(i) x(i)^2 x(i)^3 x(i)^4 x(i)^5 x(i)^6 ];
end

res_LSfit = inv(A'*A)*A'*b

sig = zeros(1, 200);        %creating array of standard deviation
euc_norm = zeros(1, 200);   %creating array of standard euclidean norm
sig(1) = 0.00000006;
b2 = b + randn(length(b), 1)*sig(1);
euc_norm(1) = norm(b - b2);
for i=1:199
    sig(i+1) = sig(i) + 0.000000001;
    b2 = b + randn(length(b), 1)*sig(i+1);  %perturbing b with the noise
    euc_norm(i+1) = norm(b - b2);   %calculating the fitting error
end
figure(1);
plot(sig, euc_norm, 'k');        %plotting standard deviation vs the error
hold on;
yline(10^(-6), 'r');
```

Listing 11: main.m - task 11 source code

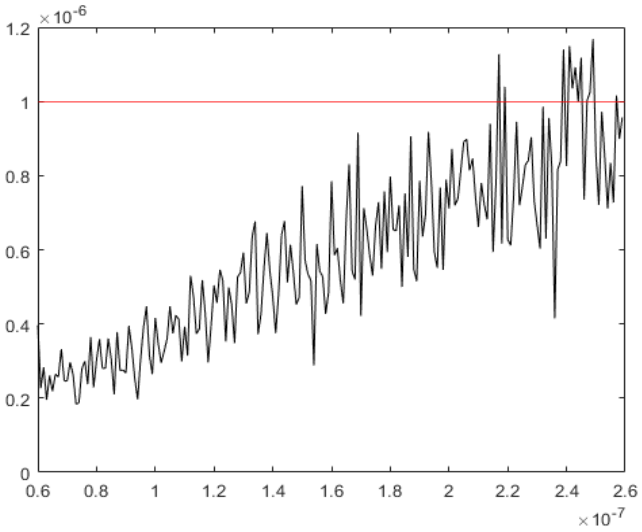

Figure 12: Euclidean norm versus standard deviation

While performing the tests, the chart's shape was varying. We can however assume that to keep the fitting error not exceeding 10^{-6} the value of σ should not exceed $2 \cdot 10^{-7}$.

2 MATLAB functions

2.1 Linear regression

```
function [a, b] = linReg(X, Y)
    b = (Y*X' - size(Y, 2) * mean(Y) * mean(X)) / (X*X' - size(Y, 2) * mean(X)^2);
    a = mean(Y) - b * mean(X);
```

Listing 12: linReg.m

2.2 Classical LS fitting

```
function [LS] = LSFit(A, b)
    LS = inv(A'*A) * A' * b;
```

Listing 13: LSFit.m

2.3 LS solution using SVD

```
function [SVD] = singValDecomp(A)
    [V, L] = eig(A'*A);

    % evaluate eigenpairs of matrix A
    L = sort(diag(sqrt(L)), 'descend'); % order values for Sigma
    S = zeros(length(L));
    % create and fill Sigma
    for i = 1:length(L)
        S(i, i) = L(i);
    end

    % evalute Sigma matrix
    [Sm, Sn] = size(S);
    tmp = zeros(Sm, Sn);
    for i = 1:min(Sm, Sn)
        tmp(i, i) = 1./S(i,i);
    end

    % evalute V factor
    V = fliplr(V);
    % evalute U factor
    U = A * V * tmp';

    SVD = V * inv(S) * U' * b;
```

Listing 14: singValDecomp.m

2.4 LS solution using TSVD

```
function [TSVD] = truncSingValDecomp(A)
    % esential threshold
    tau = 0.1;
    [V, L] = eig(A'*A);

    % evaluate eigenpairs of matrix A
    L = sort(diag(sqrt(L)), 'descend'); % order values for Sigma
    S = zeros(length(L));
    % create and fill Sigma
    for i = 1:length(L)
        S(i, i) = L(i);
    end

    % evalute Sigma matrix
    [Sm, Sn] = size(S);
    tmp = zeros(Sm, Sn);
    for i = 1:min(Sm, Sn)
        tmp(i, i) = 1./S(i, i);
    end

    % evalute V factor
    V = fliplr(V);
    % evalute U factor
    U = A * V * tmp';

    TSVD = zeros(length(diag(S)), 1);
    for i = 1:length(diag(S))
        if abs(S(i, i)) > tau
            TSVD = TSVD + (U(:, i)'*f'*V(:, i))/S(i, i);
        end
    end
```

Listing 15: truncSingValDecomp.m

2.5 Iterative Tikhonov Regularization

```
function [error] = iterTikhReg(A)
    format long;
    [~, n] = size(A);
    x_LS = zeros(n, 1);
    L = eye(n);
    alpha = 1;
    error = 1;

    % evalute an error
    while(error > 0.00000000001)
        x0 = x_LS;
        x_LS = x_LS + inv(A'*A + alpha^2*L'*L) * A' * (b - A*x_LS);
        error = norm(b - A*x0) - norm(b - A*x_LS);
    end
```

Listing 16: iterTikhReg.m

Literature

[GVL96] G. H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[GVL13] G.H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, fourth edition, 2013.