

# Assignment in Numerical Analysis LaTeX

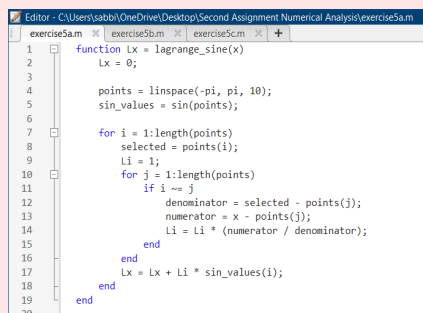
Full Name: Savvidis Theocharis 4555

January 2024

## 1 Fifth Exercise

### 1.1 Polynomial approximation

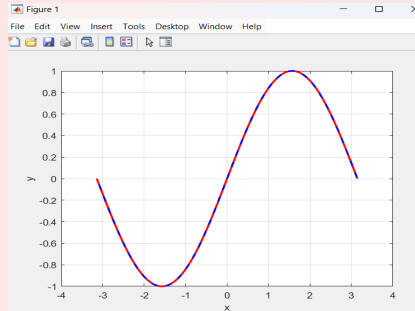
For the Polynomial approximation the Lagrange polynomial was used.

A screenshot of a MATLAB script editor window titled 'Editor - C:\Users\sabbi\OneDrive\Desktop\Second Assignment Numerical Analysis\exercice5a.m'. The script defines a function 'Lx = lagrange\_sine(x)' and implements the Lagrange polynomial approximation for the sine function. The code includes variable declarations, point generation, and nested loops for calculating the polynomial coefficients.

```
1 function Lx = lagrange_sine(x)
2     Lx = 0;
3
4     points = linspace(-pi, pi, 10);
5     sin_values = sin(points);
6
7     for i = 1:length(points)
8         selected = points(i);
9         Li = 1;
10        for j = 1:length(points)
11            if i ~= j
12                denominator = selected - points(j);
13                numerator = x - points(j);
14                Li = Li * (numerator / denominator);
15            end
16        end
17        Lx = Lx + Li * sin_values(i);
18    end
19 end
20
```

*Note: The code was implemented without the assistance of a language model.*

Testing 200 different values in the interval  $[-\pi, \pi]$ , **7 digits of precision** are obtained.



*Note: The plot displaying the curve created by 200 values using the Polynomial approximation alongside the plot of sine.*

## 1.2 Splines

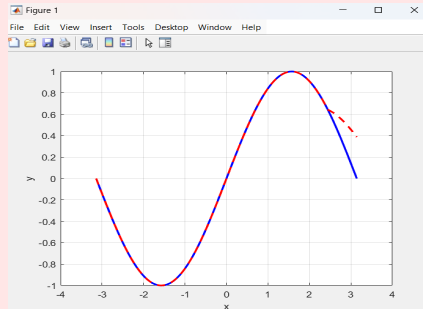
```

Editor - C:\Users\abhi\OneDrive\Desktop\Second Assignment Numerical Analysis\exercice5.m
exercice5.m | exercice5.m | +
1 function spline_value = splines_sine(x)
2 points = linspace(pi, -pi, 100);
3 sin_values = sin(points);
4
5 h = diff(points);
6 n = length(points);
7 r = zeros(1, n-2);
8 for i = 2:n-1
9     r(i-1) = 3 * ((sin_values(i+1) - sin_values(i)) / h(i) - (sin_values(i) - sin_values(i-1)) / h(i-1));
10 end
11
12 A = zeros(n-2, n-2);
13 for i = 2:n-2
14     if i == 1
15         A(i, i) = 2 * (h(i) + h(i+1));
16         A(i, i+1) = h(i+1);
17     elseif i == n-2
18         A(i, i-1) = h(i);
19         A(i, i) = 2 * (h(i) + h(i+1));
20     else
21         A(i, i-1) = h(i);
22         A(i, i) = 2 * (h(i) + h(i+1));
23         A(i, i+1) = h(i+1);
24     end
25 end
26
27 c = A \ r';
28 c = [0; c; 0];
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49 spline_value = NaN;
50 for i = 1:n-1
51     if x >= points(i) && x <= points(i+1)
52         dx = x - points(i);
53         spline_value = a(i) + b(i) * dx + c(i) * dx^2 + d(i) * dx^3;
54     end
55 end
56
57

```

*Note: The code was implemented without the assistance of a language model.*

Testing 200 different values in the interval  $[-\pi, 2.4]$  (approximation), **4 digits of precision** are obtained, while in the interval  $[2.4, \pi]$  the error is increasing as shown at the plot, losing precision digits continually.



*Note: The plot displaying the curve created by 200 values using Splines alongside the plot of sine.*

### 1.3 Least Squares

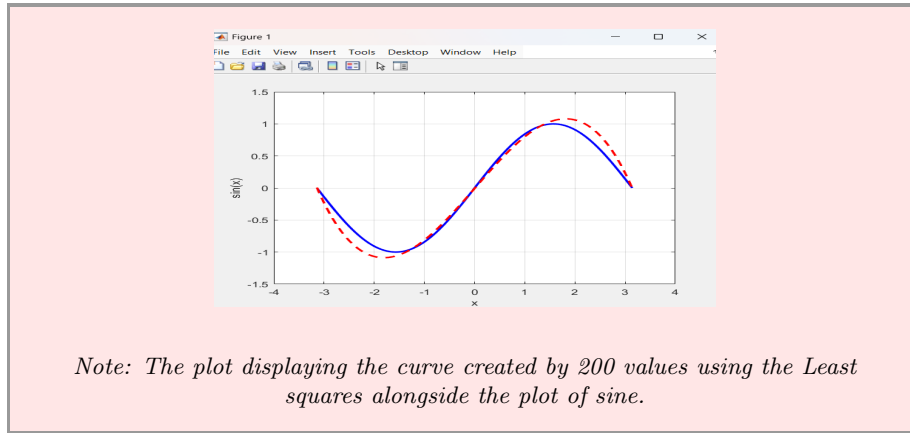
```

1 function a = least_squares_sine()
2     points = [-pi, -pi/2, -pi/3, -pi/4, -pi/6, 0, pi/6, pi/3, pi/2, pi];
3     sin_values = sin(points);
4     n = length(points);
5
6     A = zeros(n, 4);
7     for i = 1:n
8         A(i, :) = [1, points(i), points(i)^2, points(i)^3];
9     end
10
11     b = sin_values(:);
12     a = (A' * A) \ (A' * b);
13 end
14
15 a = least_squares_sine();

```

*Note: For the implementation of the Least Squares method , a polynomial of degree 3 was used.*

Testing 200 different values in the interval  $[-\pi, \pi]$  , **1 digit of precision** is obtained.



## 1.4 Comparison

Based on the specific implementations and the choices made (Lagrange polynomial, 3rd degree polynomial used in Least Squares) **most precision is achieved using the Polynomial approximation** (the first method used) as 7 digits of precision are obtained in comparison with the 4 and 1 precision digits obtained by the following methods of Splines and Least squares correspondingly.

## 2 Sixth Exercise

### 2.1 Simpson

The implementation of the Simpson method for the sine function using 11 points between the interval  $[0, \pi]$  leads to the **result 1.000003**, thus the numerical **error** is  $1.000003 - 1 = 0.000003$ , considering that

$$\int_0^{\frac{\pi}{2}} \sin(x) dx = 1$$

The **theoretical error** as calculated by the code below is **0.000008**.

```

1 %Μεθοδος Simpson
2 syms x;
3 f = sin(x);
4 first_derivative = diff(f,x);
5 second_derivative = diff(first_derivative, x);
6 a = 0;
7 b = pi/2;
8 n = 11;
9 points = linspace(a, b, n);
10 f_numeric = matlabFunction(f);
11
12 sum_odd = 0;
13 sum_even = 0;
14 for i=2:n-1
15     if mod(i, 2) == 1
16         sum_even = sum_even + f_numeric(points(i));
17     else
18         sum_odd = sum_odd + f_numeric(points(i));
19     end
20 end
21 result = (b-a) / (3*(n-1)) * (f_numeric(points(1)) + f_numeric(points(n)) + 2*sum_even + 4*sum_odd);
22 fprintf('Result: %f\n', result);
23

```

*Note: Code implementing the Simpson method.*

```

23
24 second_derivative_numeric = matlabFunction(second_derivative);
25 M = fminbnd(@(x) -abs(second_derivative_numeric(x)), a, b);
26 error_simpson = ((b - a)^5 / (180 * (n - 1)^4)) * M;
27 fprintf('Result: %f\n', result);
28 fprintf('Error %f\n', error_simpson);
29

```

*Note: Code calculating the theoretical error of the Simpson method.*

## 2.2 Trapezoid Rule

The implementation of the Trapezoid Rule for the sine function using 11 points between the interval  $[0, \pi]$  leads to the **result 0.997943**, thus the numerical error is  $1 - 0.997943 = 0.002057$ , considering that

$$\int_0^{\frac{\pi}{2}} \sin(x) dx = 1$$

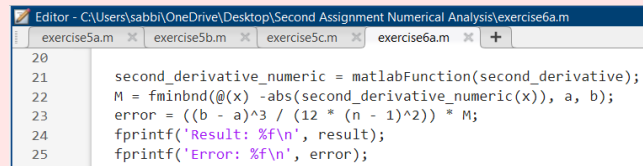
The **theoretical error** as calculated by the code below is **0.005073**.

```

1 %Μεθοδος Τραπεζιου
2 syms x;
3 f = sin(x);
4 first_derivative = diff(f,x);
5 second_derivative = diff(first_derivative, x);
6 a = 0;
7 b = pi/2;
8 n = 11;
9 points = linspace(a, b, n);
10 f_numeric = matlabFunction(f);
11 fvalues(1)=f_numeric(points(1));
12 fvalues(n)=f_numeric(points(n));
13
14 sum = 0;
15 for i=2:n-1
16     sum = sum + f_numeric(points(i));
17     fvalues(i) = f_numeric(points(i));
18 end
19 result = (b-a) / (2*(n-1)) * (f_numeric(points(1)) + f_numeric(points(n)) + 2*sum);

```

*Note: Code implementing the Trapezoid Rule.*



The image shows a MATLAB Editor window with the title bar "Editor - C:\Users\sabbil\OneDrive\Desktop\Second Assignment Numerical Analysis\exercise6a.m". The window contains four tabs: "exercise5a.m", "exercise5b.m", "exercise5c.m", and "exercise6a.m". The code in the editor is as follows:

```
20
21     second_derivative_numeric = matlabFunction(second_derivative);
22     M = fminbnd(@(x) -abs(second_derivative_numeric(x)), a, b);
23     error = ((b - a)^3 / (12 * (n - 1)^2)) * M;
24     fprintf('Result: %f\n', result);
25     fprintf('Error: %f\n', error);
```

*Note: Code for calculating the theoretical error of the Trapezoid Rule.*

### 3 Seventh Exercise

- **Conclusions and Quality Comparison**

The two stocks used from the Athens Stock Market were **LAMDA** and **KPI-KPI**. As a base **date** the **5/7/2024** was used and predictions were made for the next six days that the stock market was open (8/7/2024 - 15/7/2024).

It is evident that the **predictions** for the dates after the known prices are more accurate using the lower degree polynomial (the second degree) as it is the less volatile than the other two (third and fourth degree polynomials). **As the distance from the known prices increases, predictions made using a higher degree polynomials are becoming more and more extreme**, thus usually diverge significantly from the actual values as they adapt more efficiently to the known points-prices.

As for the **approximations for the known prices** using both the second, third and fourth degree polynomials are close to the real values. It can be observed that **as the degree of the polynomial increases, the approximations made are closer to the real values.**

### 3.1 KPI-KPI Stock

- Using a second degree polynomial

```

1 %Εκτίμηση τιμής κλεισίματος για την μετοχή της KPI-KPI ΒΙΟΜ/ΝΙΑ ΓΑΛΑΚΤΟΣ Α.Β.Ε.Ε. (KPI)
2 %Λαμβάνονται δεδομένες οι τιμές κλεισίματος από 21 Ιουνίου έως 5 Ιουλίου του 2024.
3 %Γίνεται πρόβλεψη για τις επόμενες 6 τιμές κλεισίματος (6 Ιουλίου - 15 Ιουλίου)
4 function a = least_squares()
5     points = [1,2,3,4,5,6,7,8,9,10];
6     values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
7     n = length(points);
8
9     A = zeros(n, 3);
10    for i = 1:n
11        A(i, :) = [1, points(i), points(i)^2];
12    end
13
14    b = values(:);
15    a = (A' * A) \ (A' * b);
16 end
17 a = least_squares();
18
19 for x_test=11:15
20     p_x = a(1) + a(2)*x_test + a(3)*x_test^2;
21     fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n',x_test-3, p_x);
22 end
23 x_test = x_test +1;
24 p_x = a(1) + a(2)*x_test + a(3)*x_test^2;
25 fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n',x_test-1, p_x);
26
27 %Για την ποσοτική σύγκριση
28 values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
29 for x_test = 1:10
30     p_x = a(1) + a(2) * x_test + a(3) * x_test^2;
31     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
32 end
33

```

*Note: Code implementing the Least Squares method with a **second degree polynomial** using 10 known closing prices of KPI-KPI's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for KPI-KPI's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of KPI's Stock
8/7/2024	11.5475
9/7/2024	11.5437
10/7/2024	11.5236
11/7/2024	11.4873
12/7/2024	11.4346
15/7/2024	11.3657

Also for the **already known** values the **approximations** using the second degree polynomial are the following:



Day	Approximated Value	Real Value
1	10.6895	10.6500
2	10.8486	10.9500
3	10.9914	10.9000
4	11.1180	11.0500
5	11.2282	11.3500
6	11.3221	11.4500
7	11.3998	11.2500
8	11.4611	11.4500
9	11.5062	11.4500
10	11.5350	11.6000

- Using a third degree polynomial

```

1 %KPI-KPI με πολυώνυμο τρίτου βαθμού
2 function a = least_squares_third()
3     points = [1,2,3,4,5,6,7,8,9,10];
4     values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
5     n = length(points);
6
7     A = zeros(n, 4);
8     for i = 1:n
9         A(i, :) = [1, points(i), points(i)^2, points(i)^3];
10    end
11
12    b = values(:);
13    a = (A' * A) \ (A' * b);
14 end
15 a = least_squares_third();
16
17 for x_test=11:15
18     p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3;
19     fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n', x_test-3, p_x);
20 end
21 x_test = x_test +1;
22 p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3;
23 fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n', x_test-1, p_x);
24
25 %Για την ποιοτική σύγκριση
26 values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
27 for x_test = 1:10
28     p_x = a(1) + a(2) * x_test + a(3) * x_test^2 + a(4)*x_test^3;
29     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
30 end
31
32
33

```

*Note: Code implementing the Least Squares method with a **third degree polynomial** using 10 known closing prices of KPI-KPI's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for KPI-KPI's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of KPI's Stock
8/7/2024	11.6517
9/7/2024	11.7615
10/7/2024	11.9024
11/7/2024	12.0817
12/7/2024	12.3066
15/7/2024	12.5844

Also for the **already known** values the **approximations** using the third degree polynomial are the following:

Day	Approximated Value	Real Value
1	10.6590	10.6500
2	10.8588	10.9500
3	11.0169	10.9000
4	11.1405	11.0500
5	11.2369	11.3500
6	11.3134	11.4500
7	11.3772	11.2500
8	11.4356	11.4500
9	11.4960	11.4500
10	11.5656	11.6000

- Using a fourth degree polynomial

```

Editor - C:\Users\sabbi\OneDrive\Desktop\Second Assignment Numerical Analysis\exercise7a_fourth_degree.m
exercise7a.m exercise7a.m exercise7a_third_degree.m exercise7b_third_degree.m exercise7a_fourth_degree.m exerci
1 %KPI-KPI με πολυώνυμο τέταρτου βαθμού
2 function a = least_squares_fourth()
3     points = [1,2,3,4,5,6,7,8,9,10];
4     values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
5     n = length(points);
6
7     A = zeros(n, 5);
8     for i = 1:n
9         A(i, :) = [1, points(i), points(i)^2, points(i)^3, points(i)^4];
10    end
11
12    b = values(:);
13    a = (A' * A) \ (A' * b);
14 end
15 a = least_squares_fourth();
16
17
18 for x_test=1:15
19     p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 + a(5)*x_test^4 ;
20     fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n',x_test-3, p_x);
21 end
22 x_test = x_test +1;
23 p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 + a(5)*x_test^4 ;
24 fprintf('Approximated value of KPIs stock for %d/7/2024 %.4f\n',x_test-1, p_x);
25
26 %Για την ποσοτική σύγκριση
27 values = [10.6500,10.9500,10.9000,11.0500,11.3500,11.4500,11.2500,11.4500,11.4500,11.6000];
28 for x_test = 1:10
29     p_x = a(1) + a(2) * x_test + a(3) * x_test^2 + a(4)*x_test^3 + a(5)*x_test^4;
30     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
31 end
32

```

*Note: Code implementing the Least Squares method with a **fourth degree polynomial** using 10 known closing prices of KPI-KPI's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for KPI-KPI's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of KPI's Stock
8/7/2024	11.9042
9/7/2024	12.4731
10/7/2024	13.4174
11/7/2024	14.8680
12/7/2024	16.9734
15/7/2024	19.8998

Also for the **already known** values the **approximations** using the fourth degree polynomial are the following:

Day	Approximated Value	Real Value
1	10.6907	10.6500
2	10.8200	10.9500
3	10.9869	10.9000
4	11.1458	11.0500
5	11.2687	11.3500
6	11.3452	11.4500
7	11.3825	11.2500
8	11.4056	11.4500
9	11.4572	11.4500
10	11.5974	11.6000

## 3.2 LAMDA Stock

- Using a second degree polynomial

```

1 %Εκτίμηση τιμής κλεισίματος για την μετοχή της LAMDA -Α.Ε.ΣΥΜ. & ΑΣΙΟΠ. ΑΚΙΝΗΤΩΝ (ΛΑΜΔΑ)
2 %Λαμβάνονται δεδομένες οι τιμές κλεισίματος από 21 Ιουνίου έως 5 Ιουλίου του 2024.
3 function a = least_squares_lambda()
4     points = [1,2,3,4,5,6,7,8,9,10];
5     values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
6     n = length(points);
7
8     A = zeros(n, 3);
9     for i = 1:n
10         A(i, :) = [1, points(i), points(i)^2];
11     end
12
13     b = values(:);
14     a = (A' * A) \ (A' * b);
15 end
16 a = least_squares_lambda();
17
18 for x_test=11:15
19     p_x = a(1) + a(2)*x_test + a(3)*x_test^2;
20     fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n',x_test-3, p_x);
21 end
22 x_test = x_test + 1;
23 p_x = a(1) + a(2)*x_test + a(3)*x_test^2;
24 fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n',x_test-1, p_x);
25
26 %Για την ποιοτική σύγκριση
27 values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
28 for x_test = 1:10
29     p_x = a(1) + a(2) * x_test + a(3) * x_test^2;
30     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
31 end

```

*Note: Code implementing the Least Squares method with a **second degree polynomial** using 10 known closing prices of LAMDA's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for LAMDA's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of LAMDA's Stock
8/7/2024	7.2230
9/7/2024	7.3711
10/7/2024	7.5355
11/7/2024	7.7160
12/7/2024	7.9128
15/7/2024	8.1257

Also for the **already known** values the **approximations** using the second degree polynomial are the following:

Day	Approximated Value	Real Value
1	6.6335	6.6800
2	6.6195	6.6100
3	6.6217	6.5800
4	6.6401	6.5600
5	6.6747	6.7400
6	6.7256	6.7500
7	6.7926	6.7400
8	6.8759	6.9000
9	6.9754	7.0900
10	7.0911	7.0000

- Using a third degree polynomial

```

1 %LAMBDA με πολυώνυμο τρίτου βαθμού
2
3 function a = least_squares_lambda_third()
4     points = [1,2,3,4,5,6,7,8,9,10];
5     values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
6     n = length(points);
7
8     A = zeros(n, 4);
9     for i = 1:n
10         A(i, :) = [1, points(i), points(i)^2, points(i)^3];
11     end
12
13     b = values(:);
14     a = (A' * A) \ (A' * b);
15 end
16 a = least_squares_lambda_third();
17
18
19 for x_test=1:15
20     p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 ;
21     fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n', x_test-3, p_x);
22 end
23
24 x_test = x_test +1;
25 p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 ;
26 fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n', x_test-1, p_x);
27
28 %Για την ποιοτική σύγκριση
29 values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
30 for x_test = 1:10
31     p_x = a(1) + a(2) * x_test + a(3) * x_test^2 + a(4)*x_test^3;
32     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
33 end

```

*Note: Code implementing the Least Squares method with a **third degree polynomial** using 10 known closing prices of LAMDA's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for LAMDA's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of LAMDA's Stock
8/7/2024	7.0533
9/7/2024	7.0164
10/7/2024	6.9185
11/7/2024	6.7478
12/7/2024	6.4925
15/7/2024	6.1407

Also for the **already known** values the **approximations** using the third degree polynomial are the following:

Day	Approximated Value	Real Value
1	6.6833	6.6800
2	6.6028	6.6100
3	6.5801	6.5800
4	6.6033	6.5600
5	6.6605	6.7400
6	6.7398	6.7500
7	6.8294	6.7400
8	6.9174	6.9000
9	6.9920	7.0900
10	7.0413	7.0000

- Using a fourth degree polynomial

```

Editor - C:\Users\sabbi\OneDrive\Desktop\Second Assignment Numerical Analysis\exercise7b_fourth_degree.m
exercise7a.m exercise7b.m exercise7a_third_degree.m exercise7b_third_degree.m exercise7b_fourth_degree.m exercise7c.m
1 %LAMDA με πολυώνυμο τετάρτου βαθμού
2
3 function a = least_squares_lambda_fourth()
4     points = [1,2,3,4,5,6,7,8,9,10];
5     values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
6     n = length(points);
7
8     A = zeros(n, 5);
9     for i = 1:n
10         A(i, :) = [1, points(i), points(i)^2, points(i)^3, points(i)^4];
11     end
12
13     b = values(:);
14     a = (A' * A) \ (A' * b);
15 end
16 a = least_squares_lambda_fourth();
17
18
19 for x_test=1:15
20     p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 + a(5)*x_test^4;
21     fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n', x_test-3, p_x);
22 end
23 x_test = x_test +1;
24 p_x = a(1) + a(2)*x_test + a(3)*x_test^2 + a(4)*x_test^3 + a(5)*x_test^4;
25 fprintf('Approximated value of LAMDA's stock for %d/7/2024 %.4f\n', x_test-1, p_x);
26
27 %Για την ποιοτική σύγκριση
28 values = [6.6800,6.6100,6.5800,6.5600,6.7400,6.7500,6.7400,6.9000,7.0900,7.0000];
29 for x_test = 1:10
30     p_x = a(1) + a(2) * x_test + a(3) * x_test^2 + a(4)*x_test^3 + a(5)*x_test^4;
31     fprintf('Day %d: Approximated value %.4f vs Real value %.4f\n', x_test, p_x, values(x_test));
32 end

```

*Note: Code implementing the Least Squares method with a **fourth degree polynomial** using 10 known closing prices of LAMDA's stock.*

As it is displayed below, the **predictions** for the day after the known closing prices as well as the other 5 consecutive closing price predictions for LAMDA's stock for the dates after 5/7/2024 with an open stock market are:

Date	Predicted Value of LAMDA's Stock
8/7/2024	6.9733
9/7/2024	6.7909
10/7/2024	6.4385
11/7/2024	5.8650
12/7/2024	5.0139
15/7/2024	3.8230

Also for the **already known** values the **approximations** using the fourth degree polynomial are the following:

Day	Approximated Value	Real Value
1	6.6732	6.6800
2	6.6152	6.6100
3	6.5897	6.5800
4	6.6016	6.5600
5	6.6504	6.7400
6	6.7297	6.7500
7	6.8277	6.7400
8	6.9269	6.9000
9	7.0043	7.0900
10	7.0312	7.0000