Problem set 1a: MATLAB Fundamentals and Graphics

Name: Stanciu Iulia-Cristina

MATLAB release used: R2020a

Collaboration Info:

- https://ch.mathworks.com/help/matlab/polynomials.html
- https://ch.mathworks.com/matlabcentral/answers/23982-plotting-a-polynomial-function
- https://ch.mathworks.com/matlabcentral/answers/16963-how-to-print-in-the-same-line
- https://ch.mathworks.com/help/matlab/ref/linspace.html
- https://ch.mathworks.com/help/matlab/matlab_prog/formatting-strings.html#responsive_offcanvas

Exercise 1. Polygons and colours.

Explanation:

I first defined the colors that I needed to use, both for the board and for the 7 different pieces of Tetris. After, I defined the surfaces that needed to be filled in order to make up the game board: I chose to make it the same size as the one on the guide (12 horizontal squares * 20 vertical squares). The board size is 12.2*20.2. The reason behind that is that I made the grey squares 0.8*0.8 and added 0.2 spacing between them as well as between the marginal sqares and the edge of the board. I used fill for drawing the board. For the small squares, the fill function was used in a "for" loop.
I also defined each of the 7 pieces with coordinates and colors.
The "tetris" function creates the board, then it analyses every piece that it has to put on the board. The first criteria (used as a "switch" statement) is the number of 90-degree counterclockwise rotation. The second one is the shape and with this information as well as the x and y displacement of the piece, the piece is drawn. Each piece is made of 1*1 squares that are placed over the light grey squares and half of the lateral spacing.

The board is drawn at certain coordinates, the axis were removed with "axis off" and a title was added.

MATLAB Code:

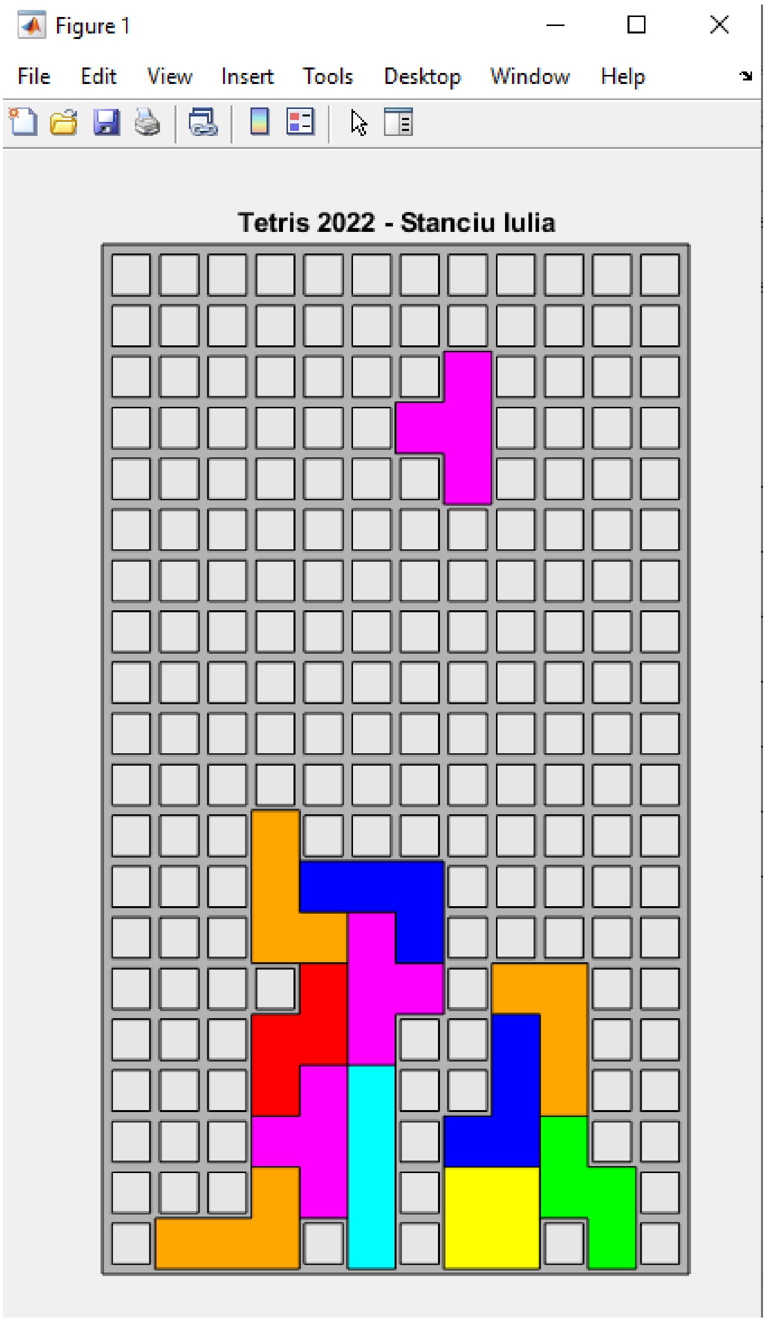
```
close all
clear
clc

tetris([1,6,0,1;2,5,1,1;4,7,0,0;3,7,8,2;2,5,7,3;5,10,3,1;
5,3,9,3;2,8,15,1;7,5,3,1;6,11,0,1;5,1,0,0;3,9,2,1]);

function tetris(Matrix)

    Squares_horizontal = 12;
    Squares_vertical = 20;
    Spacing = 0.2;

    Grey = [0.7 0.7 0.7];
    LightGrey = [0.9 0.9 0.9];

    color1 = [0 1 1]; %cyan
    X1 = [0 0 4 4];
    Y1 = [0 1 1 0];

    color2 = [1 0 1]; %magenta
    X2 = [0 0 1 1 2 2 3 3];
    Y2 = [0 1 1 2 2 1 1 0];

    color3 = [0 0 1]; %blue
    X3 = [0 0 1 1 3 3];
    Y3 = [0 2 2 1 1 0];
```

```
color4 = [1 1 0]; %yellow
X4 = [0 0 2 2];
Y4 = [0 2 2 0];

color5 = [1 0.65 0]; %orange
X5 = [0 0 2 2 3 3];
Y5 = [0 1 1 2 2 0];

color6 = [0 1 0]; %green
X6 = [0 0 1 1 3 3 2 2];
Y6 = [0 1 1 2 2 1 1 0];

color7 = [1 0 0]; %red
X7 = [0 0 2 2 3 3 1 1];
Y7 = [1 2 2 1 1 0 0 1];

X = [0 0 Squares_horizontal + Spacing Squares_horizontal + Spacing];
Y = [0 Squares_vertical + Spacing Squares_vertical + Spacing 0];

f = figure;
f.Position = [200 0 420 700];
fill(X, Y, Grey)
hold on

for n = 0:Squares_horizontal-1
   for m = 0:Squares_vertical-1
      Xb = [(Spacing + n) (Spacing + n) (1 + n) (1 + n)];
      Yb = [(Spacing + m) (1 + m) (1 + m) (Spacing + m)];
      fill(Xb, Yb, LightGrey)
   end
end

fill(Xb, Yb, LightGrey)
hold on

for row = 1:size(Matrix, 1)
   switch Matrix(row, 4)
      case 0
         switch Matrix(row, 1)
            case 1
               fill(X1+Matrix(row, 2)+Spacing/2, Y1+Matrix(row, 3)+Spacing/2, color1)
            case 2
               fill(X2+Matrix(row, 2)+Spacing/2, Y2+Matrix(row, 3)+Spacing/2, color2)
            case 3
               fill(X3+Matrix(row, 2)+Spacing/2, Y3+Matrix(row, 3)+Spacing/2, color3)
            case 4
               fill(X4+Matrix(row, 2)+Spacing/2, Y4+Matrix(row, 3)+Spacing/2, color4)
            case 5
               fill(X5+Matrix(row, 2)+Spacing/2, Y5+Matrix(row, 3)+Spacing/2, color5)
            case 6
               fill(X6+Matrix(row, 2)+Spacing/2, Y6+Matrix(row, 3)+Spacing/2, color6)
            case 7
               fill(X7+Matrix(row, 2)+Spacing/2, Y7+Matrix(row, 3)+Spacing/2, color7)
         end
```

```
        case 1
            switch Matrix(row, 1)
                case 1
                    fill(-Y1+Matrix(row, 2)+Spacing/2, X1+Matrix(row, 3)+Spacing/2, color1)
                case 2
                    fill(-Y2+Matrix(row, 2)+Spacing/2, X2+Matrix(row, 3)+Spacing/2, color2)
                case 3
                    fill(-Y3+Matrix(row, 2)+Spacing/2, X3+Matrix(row, 3)+Spacing/2, color3)
                case 4
                    fill(-Y4+Matrix(row, 2)+Spacing/2, X4+Matrix(row, 3)+Spacing/2, color4)
                case 5
                    fill(-Y5+Matrix(row, 2)+Spacing/2, X5+Matrix(row, 3)+Spacing/2, color5)
                case 6
                    fill(-Y6+Matrix(row, 2)+Spacing/2, X6+Matrix(row, 3)+Spacing/2, color6)
                case 7
                    fill(-Y7+Matrix(row, 2)+Spacing/2, X7+Matrix(row, 3)+Spacing/2, color7)
            end

        case 2
            switch Matrix(row, 1)
                case 1
                    fill(-X1+Matrix(row, 2)+Spacing/2, -Y1+Matrix(row, 3)+Spacing/2, color1)
                case 2
                    fill(-X2+Matrix(row, 2)+Spacing/2, -Y2+Matrix(row, 3)+Spacing/2, color2)
                case 3
                    fill(-X3+Matrix(row, 2)+Spacing/2, -Y3+Matrix(row, 3)+Spacing/2, color3)
                case 4
                    fill(-X4+Matrix(row, 2)+Spacing/2, -Y4+Matrix(row, 3)+Spacing/2, color4)
                case 5
                    fill(-X5+Matrix(row, 2)+Spacing/2, -Y5+Matrix(row, 3)+Spacing/2, color5)
                case 6
                    fill(-X6+Matrix(row, 2)+Spacing/2, -Y6+Matrix(row, 3)+Spacing/2, color6)
                case 7
                    fill(-X7+Matrix(row, 2)+Spacing/2, -Y7+Matrix(row, 3)+Spacing/2, color7)
            end

        case 3
            switch Matrix(row, 1)
                case 1
                    fill(Y1+Matrix(row, 2)+Spacing/2, -X1+Matrix(row, 3)+Spacing/2, color1)
                case 2
                    fill(Y2+Matrix(row, 2)+Spacing/2, -X2+Matrix(row, 3)+Spacing/2, color2)
                case 3
                    fill(Y3+Matrix(row, 2)+Spacing/2, -X3+Matrix(row, 3)+Spacing/2, color3)
                case 4
                    fill(Y4+Matrix(row, 2)+Spacing/2, -X4+Matrix(row, 3)+Spacing/2, color4)
                case 5
                    fill(Y5+Matrix(row, 2)+Spacing/2, -X5+Matrix(row, 3)+Spacing/2, color5)
                case 6
                    fill(Y6+Matrix(row, 2)+Spacing/2, -X6+Matrix(row, 3)+Spacing/2, color6)
                case 7
                    fill(Y7+Matrix(row, 2)+Spacing/2, -X7+Matrix(row, 3)+Spacing/2, color7)
            end
    end
end
```

```
    title("Tetris 2022 - Stanciu Iulia")
    axis([0 Squares_horizontal + Spacing 0 Squares_vertical + Spacing])
    axis off
end
```

Results:



Comments:

I tried but I found it difficulty to add a darker shade on the edge of the pieces. I also had problems with setting a size to the figure; even though the fill was explicit enough, the axix didn't have the same proportion at first.

Exercise 2. Distorted plane.

Explanation:

I first defined the radius of the sphere, the depth of the distorsion and alpha (a in the code). After that I defined the range for the coordonates and used meshgrid to create the matrices needed for the z=0 plane. The z coordinate follows the function given in the exercise file. Using mesh, the distorted plane was created.

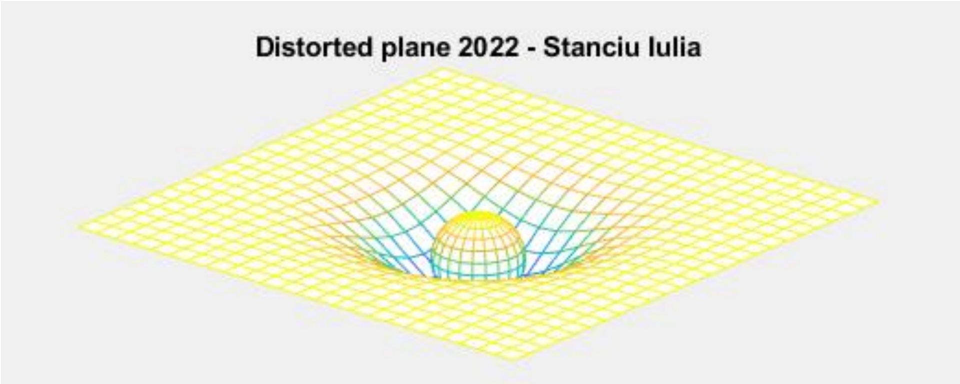
For the sphere two matrices “theta” and “psi” were created, with meshgrid, and then I needed to compute the matrices X, Y and Z. I used mesh to create the sphere, The Z coordinate was modified in order for the sphere to be tangent to the plane.

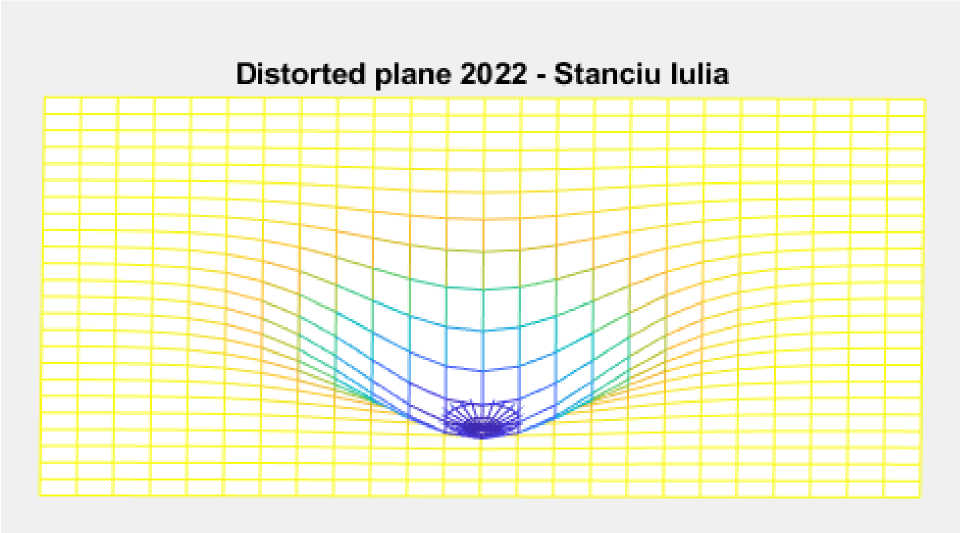
Hold on was used to display both the plane and the sphere.
A title was added, the axis were removed (axis off) and set equal (axis equal).

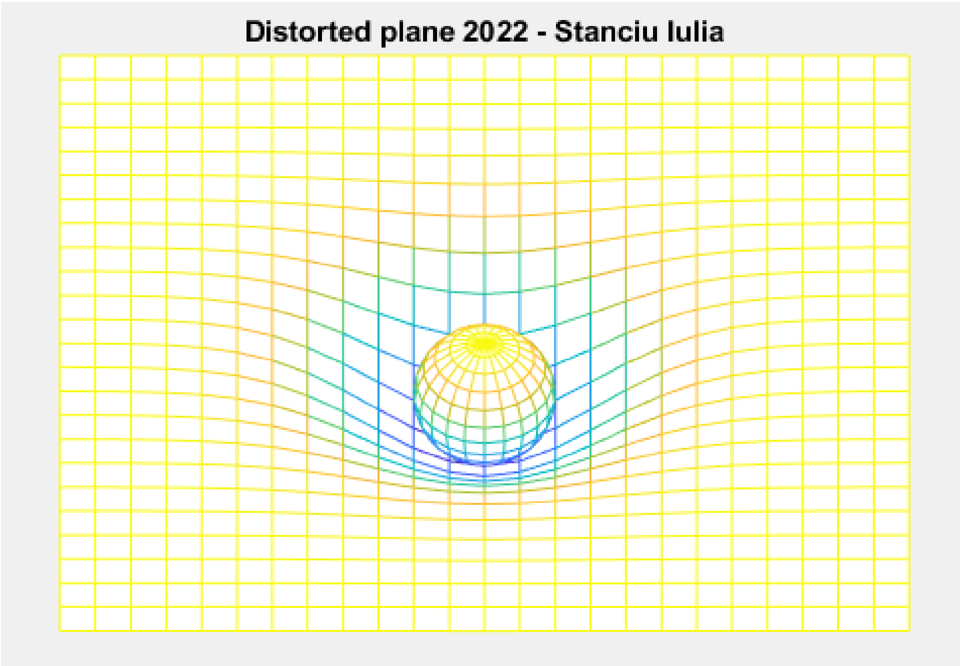
MATLAB Code:

```
close all
clear
clc

R = 10;
K = 20;
a = 5 .* R.^2;

x = -60:5:60;
y = x;
[xx,yy] = meshgrid(x,y);

z = -1 .* K .* exp((xx.^2+yy.^2)/-a);
mesh(x,y,z); hold on

psi = linspace(0,pi,13);
theta = linspace(0,2*pi,23);
psi = [psi; psi];
theta = [theta; theta];
[psi,theta] = meshgrid(psi,theta);

X = R.*sin(psi).*cos(theta);
Y = R.*sin(psi).*sin(theta);
Z = R.*cos(psi) - K + R;
mesh(X,Y,Z)

title("Distorted plane 2022 - Stanciu Iulia")
axis off
axis equal
```

Results:







Comments:

At first I tried to use surf instead of mesh, thinking it would do the same thing, but it took me some time to realize the differences.

Exercise 3. Sonine polynomials (a.k.a. generalized Laguerre polynomials).

Explanation:

I first defined the known parameters: alpha, and the first two Sonine polynomials. Thei used conv and addition of polynomials to apply the recursive formula. I printed the coefficients using fprintf and display. For the generalization, a random n and alpha are used and, with a “for” loop the Sonine polynomial is calculated with conv from the previous two polynomials. The coefficients of this polynomial are also printed.

To be able to perform adding/ subtracting of polynomials as well as plotting of the functions, the coefficient vectors needed to have the same length. I used length to determine the size and added zeros to the left (not modifying the formula).

For the plotting part, the range for the axis was set (t between 0 and 8, and also axis([0 8 -10 10])). Grid on was used to see the grid. The colors for each function plot were chosen by MATLAB and the legend is meant to be placed in the best location for the given figure.
A title was added, describing the contents of the picture.

Observation:

The generalized function example is not on the figure.

MATLAB Code:

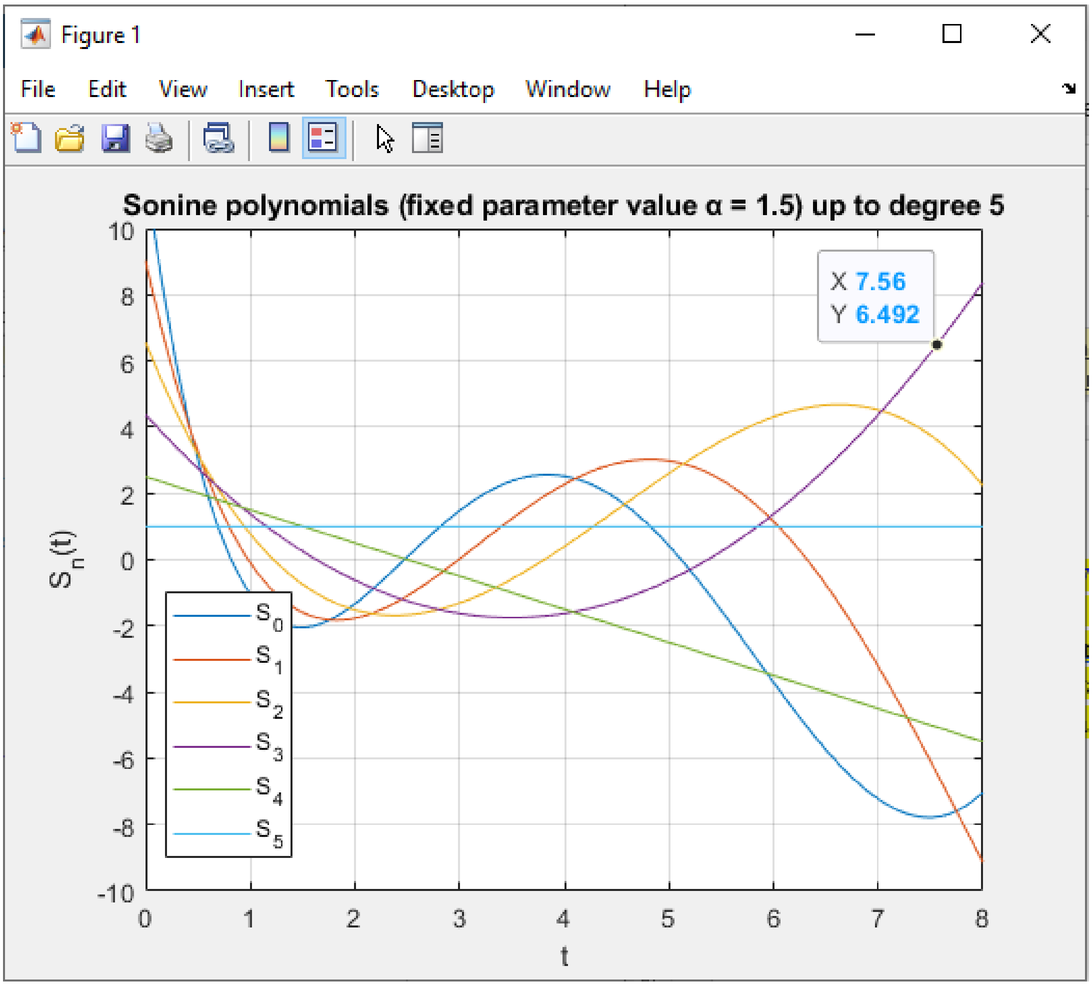
```
close all
clear
clc

alpha = 1.5;
t = 0:0.01:8;
n = 0;

S0 = [0 1];
fprintf('S0:')
disp(S0)

n = n + 1;
S1 = [-1 1+alpha];
fprintf('S1:')
disp(S1)

n = n + 1;
S2 = conv([(-1/n) (2-1/n+alpha/n)], S1) + conv([0 (-1+1/n-alpha/n)], S0);
fprintf('S2:')
disp(S2)

n = n + 1;
S3 = conv([(-1/n) (2-1/n+alpha/n)], S2) + conv([0 0 (-1+1/n-alpha/n)], S1);
fprintf('S3:')
disp(S3)
```

```
n = n + 1;
S4 = conv([(-1/n) (2-1/n+alpha/n)], S3) + conv([0 0 (-1+1/n-alpha/n)], S2);
fprintf('S4:')
disp(S4)

n = n + 1;
S5 = conv([(-1/n) (2-1/n+alpha/n)], S4) + conv([0 0 (-1+1/n-alpha/n)], S3);
fprintf('S5:')
disp(S5)

%for ramdom n and alpha
n = 9;
alpha = 3;
S_before_last = S0;
S_last = S1;

for i = 2:n
    Coefficient1 = [(-1/i) (2-1/i+alpha/i)];
    Coefficient0 = [0 (-1+1/i-alpha/i)];

    l1 = length(S_last);
    l0 = length(S_before_last);

    S_before_last = [zeros(1, l1-l0), S_before_last];

    S = conv(Coefficient1, S_last) + conv(Coefficient0, S_before_last);

    S_before_last = S_last;
    S_last = S;
end

fprintf('Polynomial function of grade %d and alpha %f is S%d:', n, alpha, n)
disp(S)

ls5 = length(S5);
ls4 = length(S4);
ls3 = length(S3);
ls2 = length(S2);
ls1 = length(S1);
ls0 = length(S0);

S4 = [zeros(1, ls5-ls4), S4];
S3 = [zeros(1, ls5-ls3), S3];
S2 = [zeros(1, ls5-ls2), S2];
S1 = [zeros(1, ls5-ls1), S1];
S0 = [zeros(1, ls5-ls0), S0];

plot(t,polyval(S5,t)); hold on
plot(t,polyval(S4,t));
plot(t,polyval(S3,t))
plot(t,polyval(S2,t))
plot(t,polyval(S1,t))
plot(t,polyval(S0,t))
```

```
axis([0 8 -10 10])
grid on

xlabel('t')
ylabel('S_n(t)')
title('Sonine polynomials (fixed parameter value ? = 1.5) up to degree 5')
legend('S_0', 'S_1', 'S_2', 'S_3', 'S_4', 'S_5','Location','best')
```

Results:

```
S0:      0      1

S1:   -1.0000    2.5000

S2:    0.5000   -3.5000    4.3750

S3:   -0.1667    2.2500   -7.8750    6.5625

S4:    0.0417   -0.9167    6.1875  -14.4375    9.0234

S5:   -0.0083    0.2708   -2.9792   13.4063  -23.4609   11.7305

Polynomial function of grade 9 and alpha 3.000000 is S9:   -0.0000    0.0003   -0.0127    0.2876   -3.7368   28.3031 -122.0933  280.5000 -300.0000  110.5000
```



Comments:

I had some problems understanding that the sum and difference of polynomial functions could only be done if the vectors were the same length. I also struggled to plot the functions, before I discovered the polyval function.