Assignment 4

Table of Contents

Name: Claudio Maggioni

Date: 2020-05-17

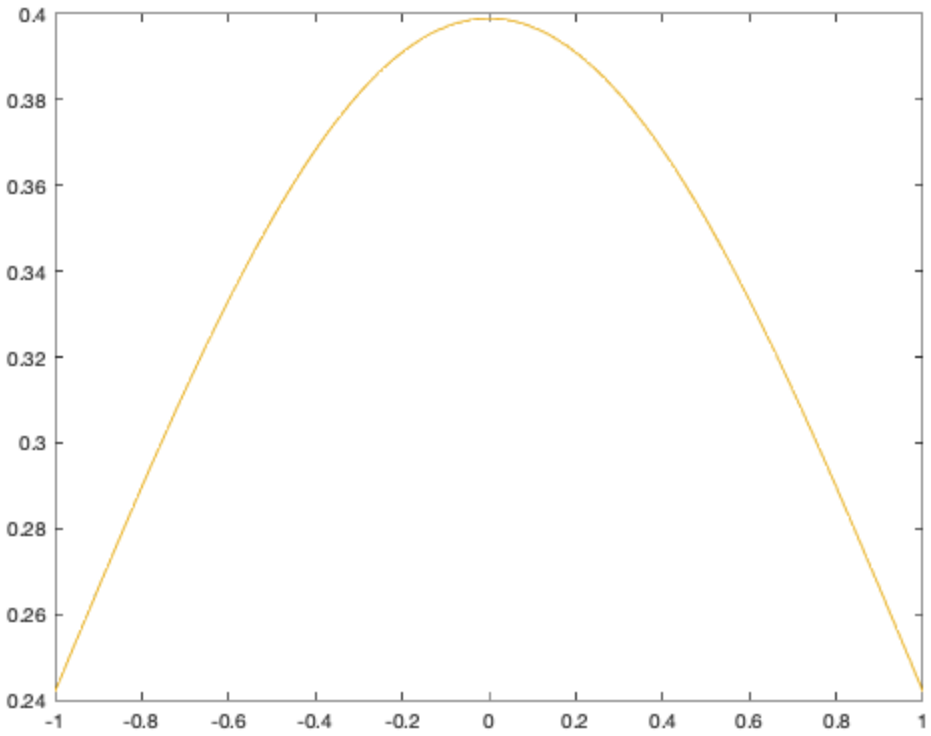
This is a template file for the first assignment to get started with running and publishing code in Matlab. Each problem has its own section (delineated by %%) and can be run in isolation by clicking into the particular section and pressing `Ctrl` + `Enter` (evaluate current section).

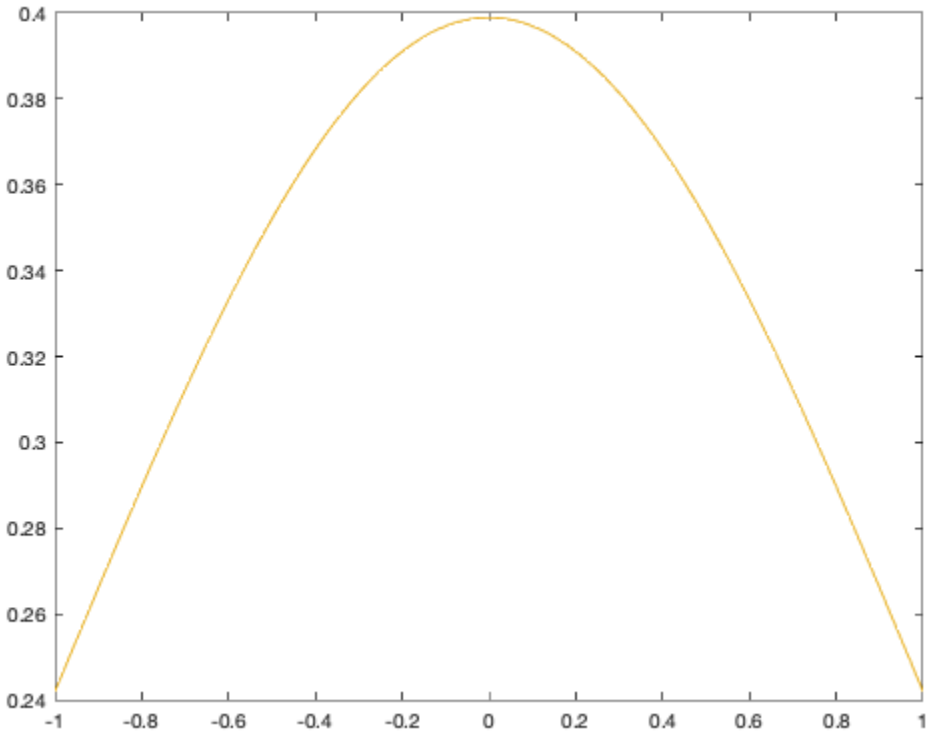
To generate a pdf for submission in your current directory, use the following three lines of code at the command window:

```
>> options.format = 'pdf'; options.outputDir = pwd; publish('assignment4.m', options)
```

Problem 3

```
clear all;
clc;
clf reset;

n=10;
f = @(x) (exp(-(x^2)/2)/sqrt(2*pi));
x_5 = computeEquidistantXs(5);
x_10 = computeEquidistantXs(10);
x_5c = computeChebyshevXs(5);
x_10c = computeChebyshevXs(10);

xe = (-1:0.01:1)';
y = zeros(size(xe));
for i = 1:size(xe, 1)
    y(i) = f(xe(i));
end

p_5 = computePolypoints(f, xe, x_5, 5);
p_10 = computePolypoints(f, xe, x_10, 10);

p_5c = computePolypoints(f, xe, x_5c, 5);
p_10c = computePolypoints(f, xe, x_10c, 10);

plot(xe, p_5, xe, p_10, xe, y);
figure;
plot(xe, p_5c, xe, p_10c, xe, y);
```


0.4
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


0.4
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Question 6

```
y = [-0.0044; -0.0771; -0.2001; -0.3521; -0.3520; 0; 0.5741;
 0.8673; ...
    0.5741; 0; 0.3520; -0.3521; 0.2001; -0.0771; -0.0213; -0.0044];
figure;
alpha = b3interpolate(y);
x = (0:0.01:16)';
y_c = spline_curve(alpha, x);
plot(x, y_c);
```

Question 3 (continued)

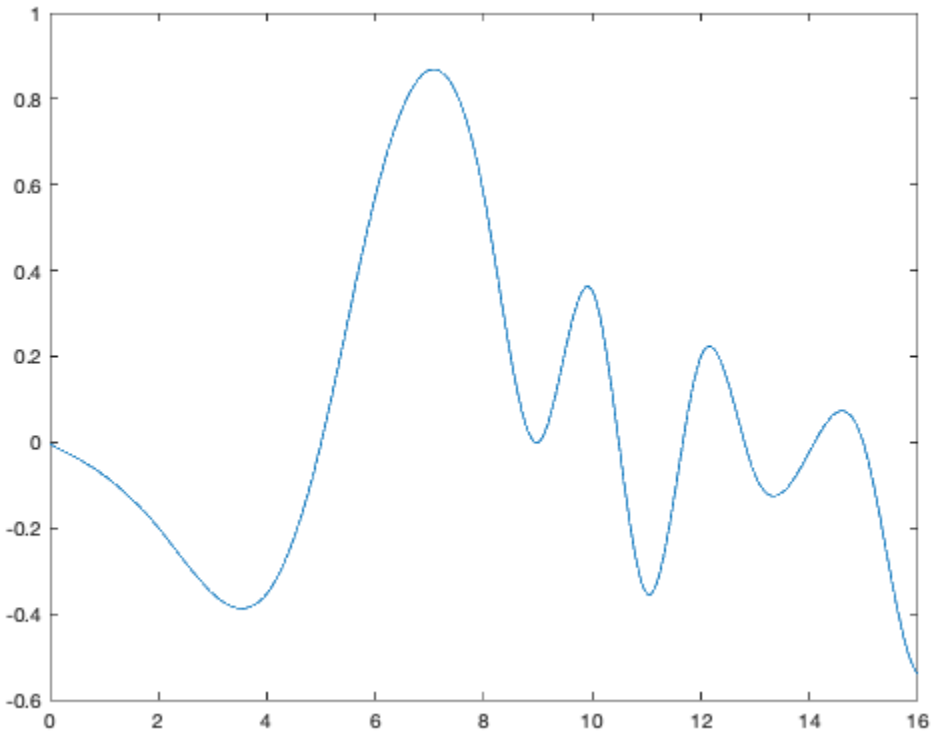
```
function x = computeEquidistantXs(n)
    x = zeros(2*n+1,1);
    for i = 1:2*n+1
        x(i) = (i-n-1)/n;
    end
end


function x = computeChebyshevXs(n)
    x = zeros(2*n+1,1);
    for i = 1:2*n+1
        x(i) = cos((2*i - 1) *pi / (4*n + 2));
    end
end


function p = computePolypoints(f, xe, x, n)
    p = zeros(size(xe));

    for i = 1:(2*n+1)
        e_i = zeros(2*n+1, 1);
        e_i(i) = 1;
        N = NewtonInterpolation(x, e_i);
        p = p + f(x(i)) * HornerNewton(N, x, xe);
    end
end

% Assuming x and y are column vectors with the same length
function N = NewtonInterpolation (x,y)
    n = size(x, 1);
    N = y;
    for i = 1:n
        N(n:-1:i+1) = (N(n:-1:i+1) - N(n-1:-1:i)) ./ (x(n:-1:i+1) -
 x(n-i:-1:1));
    end
end

% N is the array of coefficients
%
```

```
% xi evaluation points
function p = HornerNewton(N, x, xe)
    n = size(x, 1);
    p = ones(size(xe, 1), 1) * N(n);
    for i = n-1:-1:1
        p = p .* (xe - x(i)) + N(i);
    end
end
```

Problem 6 (continued)

```
% assuming x_i = i - 1
function [alpha] = b3interpolate(y)
    n = size(y, 1);
    A = zeros(n+2);
    B = [y; 0; 0];
    for x_i = 0:n-1
        for j = -1:n
            %fprintf("A(%d, %d) = B3(%d - %d)\n", x_i + 1, j+2, x_i,
 j);
            A(x_i + 1, j + 2) = B3(x_i - j);
        end
    end
    A(n+1, 1:3) = [1 -2 1];
    A(n+2, n-1:n+1) = [1 -2 1];
    alpha = A \ B;
```

```
end

function [v] = spline_curve(alpha, x)
    n = size(alpha, 1) - 2;
    v = zeros(size(x));
    for i = 1:size(x,1)
        for j = -1:n
            v(i) = v(i) + alpha(j + 2) * B3(x(i) - j);
        end
    end
end

function [y] = B3(x)
y = zeros (size (x));
i1 = find (-2 < x & x < -1);
i2 = find (-1 <= x & x < 1);
i3 = find (1 <= x & x < 2);

y(i1) = 0.5 * (x(i1) + 2) .^ 3;
y(i2) = 0.5 * (3 * abs (x(i2)) .^ 3 - 6 * x(i2) .^ 2 + 4);
y(i3) = 0.5 * (2 - x(i3)) .^ 3;
y = y / 3;
end
```



Published with MATLAB® R2019b