Assignment 4 - Introduction to Computational Science

Table of Contents

Exercise 3 1
Exercise 6 6
"Newton Interpolation" 6
"Horner Newton" 7
"b3interpolate" 7
"spline_curve" 7

Name: Stefano Gonçalves Simao

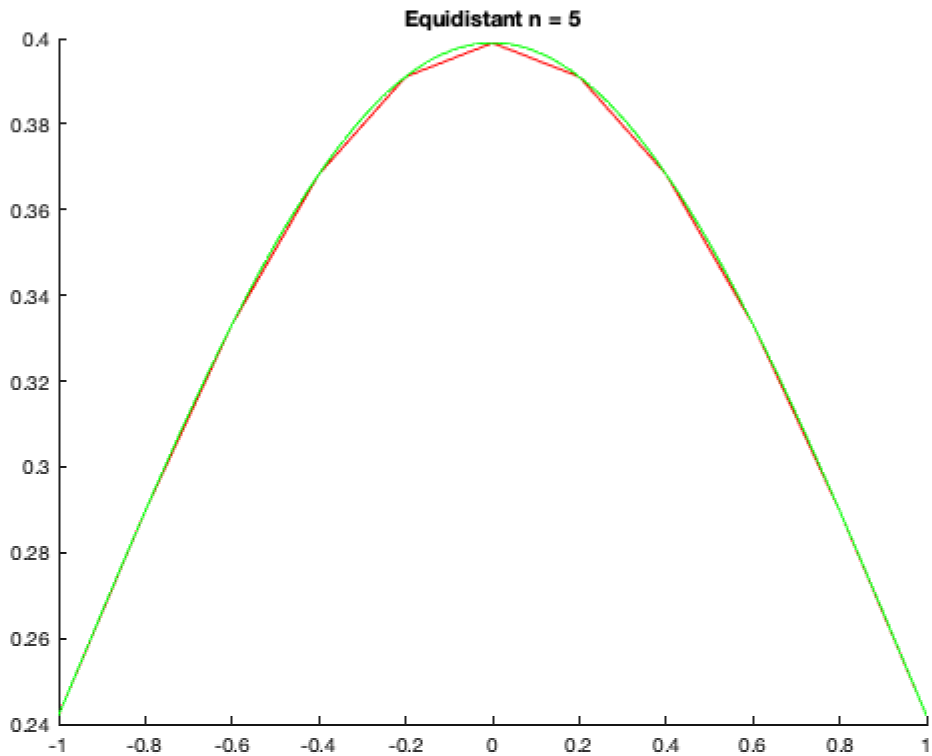
Date: 14/5/2021

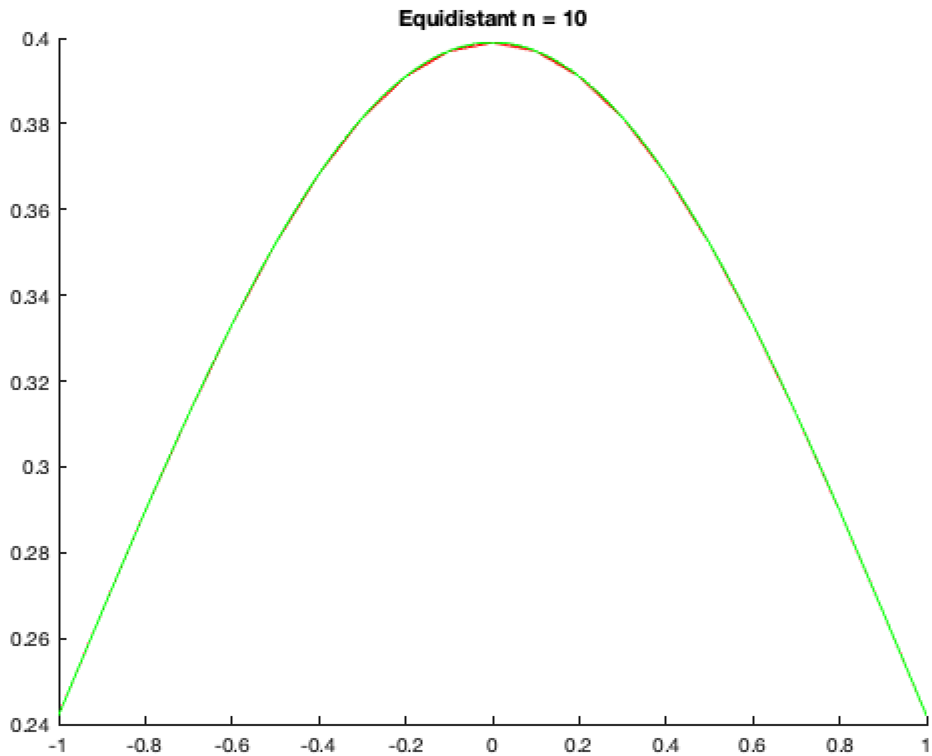
Exercise 3

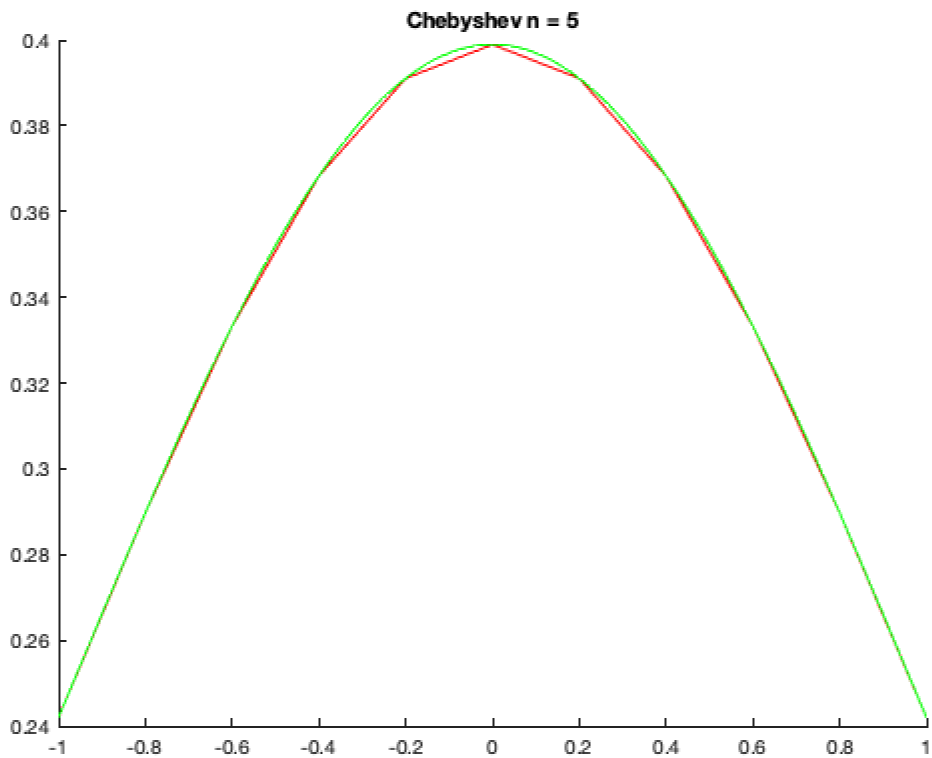
```
clc
clear

% Gaussian Density function
f = @(x) ((1 / sqrt(2 * pi)) * exp(- (x .^ 2) ./ 2));
xi = (-1:0.01:1);

% Equidistant n = 5
n1 = 5;
x1 = (-n1:n1)./ n1;
xi_11 = ((-1:0.2:1));
p1 = zeros(size(xi_11));
for i = 1: (2 * n1 + 1)
    e = zeros(1, 2 * n1 + 1);
    e(i) = 1;
    N1 = NewtonInterpolation(x1, e);
    p1 = p1 + f(x1(i)) * HornerNewton(N1, x1, xi_11);
end
figure();
title("Equidistant n = 5")
hold on
plot(xi_11, p1, "-", "color", "red");
plot(xi, f(xi), "-", "color", "green");
hold off

% Equidistant n = 10
n2 = 10;
x2 = (-n2:n2)./ n2;
xi_21 = ((-1:0.1:1));
p2 = zeros(size(xi_21));
```

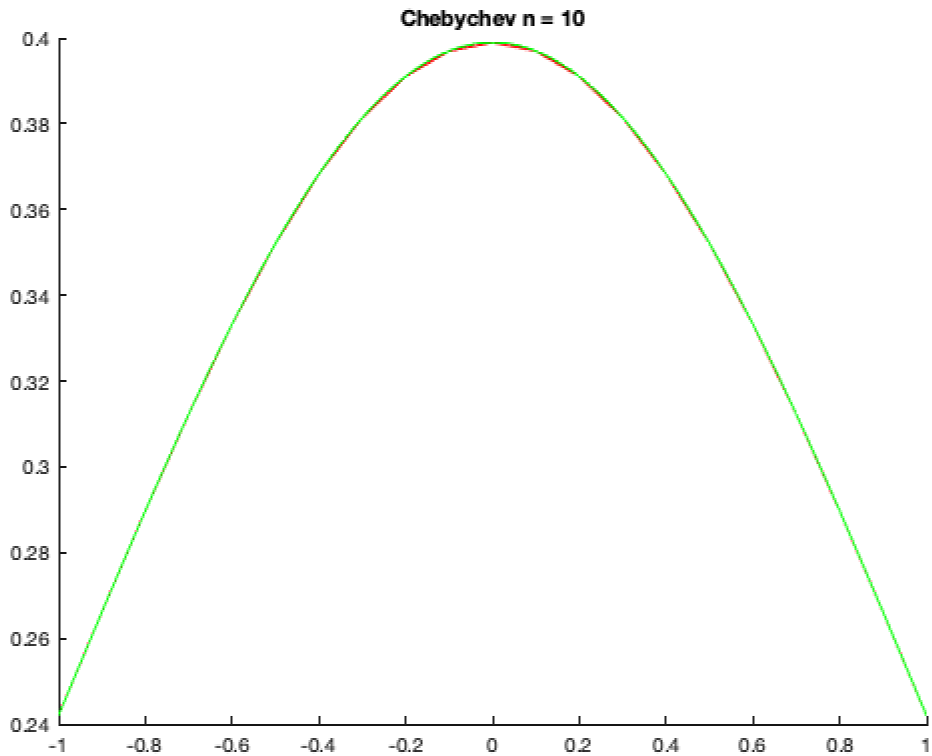
```
for i = 1: (2 * n2 + 1)
    e = zeros(1, 2 * n2 + 1);
    e(i) = 1;
    N2 = NewtonInterpolation(x2, e);
    p2 = p2 + f(x2(i)) * HornerNewton(N2, x2, xi_21);
end
figure();
title("Equidistant n = 10")
hold on
plot(xi_21, p2, "-", "color", "red");
plot(xi, f(xi), "-", "color", "green");
hold off


% Chebyshev n = 5
n3 = 5;
x3 = (cos((2 * (1: (2 * n3 + 1)) - 1) / (4 * n3 + 2) * pi));
xi_11c = ((-1:0.2:1));
p3 = zeros(size(xi_11c));
for i = 1: (2 * n3 + 1)
    e = zeros(1, 2 * n3 + 1);
    e(i) = 1;
    N3 = NewtonInterpolation(x3, e);
    p3 = p3 + f(x3(i)) * HornerNewton(N3, x3, xi_11c);
end
figure();
title("Chebyshev n = 5")
hold on
plot(xi_11c, p3, "-", "color", "red");
plot(xi, f(xi), "-", "color", "green");
hold off

% Chebyshev n = 10
n4 = 10;
x4 = (cos((2 * (1: (2 * n4 + 1)) - 1) / (4 * n4 + 2) * pi));
xi_21c = ((-1:0.1:1));
p4 = zeros(size(xi_21c));
for i = 1: (2 * n4 + 1)
    e = zeros(1, 2 * n4 + 1);
    e(i) = 1;
    N4 = NewtonInterpolation(x4, e);
    p4 = p4 + f(x4(i)) * HornerNewton(N4, x4, xi_21c);
end
figure();
title("Chebychev n = 10")
hold on
plot(xi_21c, p4, "-", "color", "red");
plot(xi, f(xi), "-", "color", "green");
hold off

% Lagrange
n5 = 10;
x5 = (cos((2 * (1: (2 * n5 + 1)) - 1) / (4 * n5 + 2) * pi));
xi_21l = ((-1:0.1:1));
```

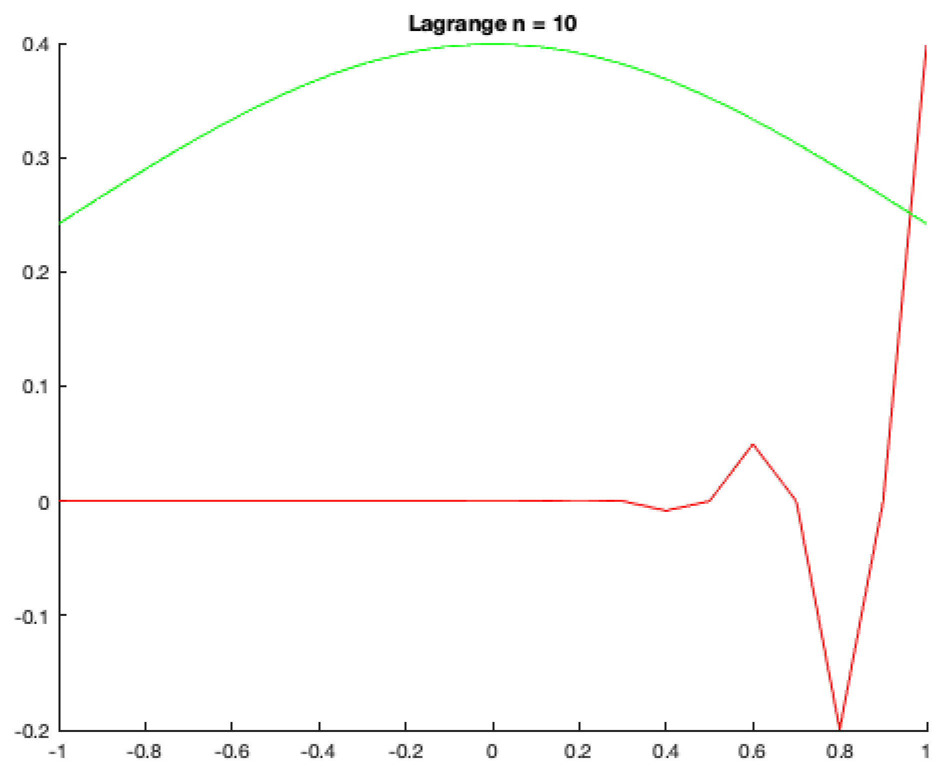
```
p5 = zeros(size(xi_21l));
for i=1:(2 * n5 + 1)
    p=1;
    for j=1:(2 * n5 + 1)
        if j ~= i
            c = poly(x5(j))/(x5(i)-x5(j));
            p = conv(p,c);
        end
    end
    p5 = p5 + p*f(x5(i));
end
figure();
title("Lagrange n = 10")
hold on
plot(xi_21l, p5, "-", "color", "red");
plot(xi, f(xi), "-", "color", "green");
hold off
```




Equidistant n = 10
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

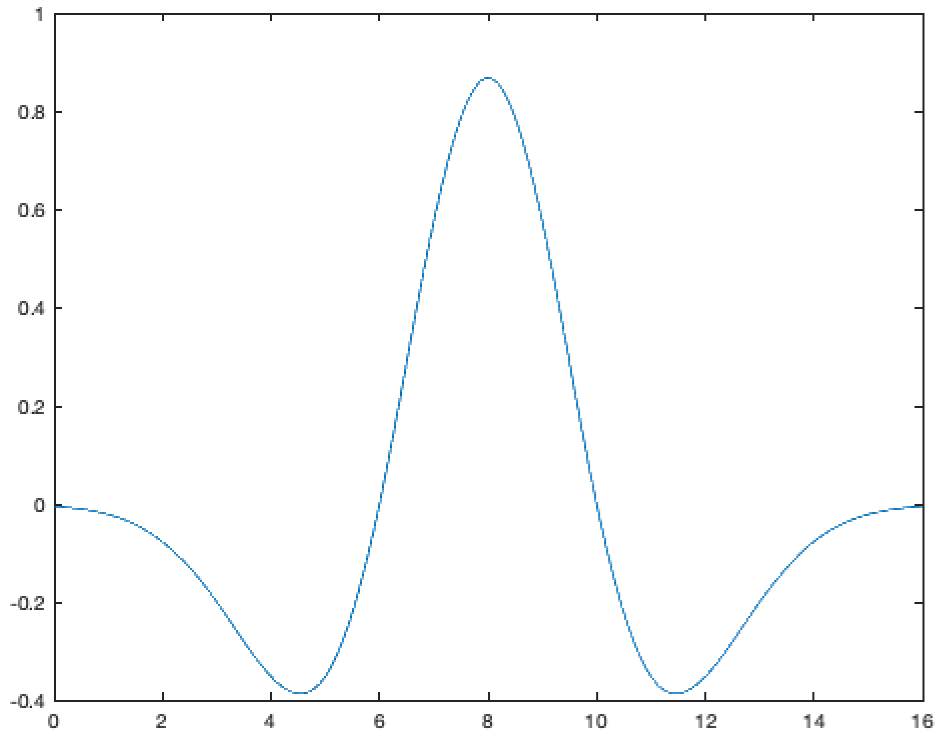

Chebyshev n = 5
0.4
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Chebychev n = 10
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


Lagrange n = 10
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Exercise 6

```
x_i = (0:1:16)';

y = [ -0.0044, -0.0213, -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 ]';
x = (0:0.01:16);

alpha = b3interpolate(y);
v = spline_curve(alpha, x);
figure();
plot(x, v);
```



"Newton Interpolation"

```
function N = NewtonInterpolation (x, y)
 n = length(x);
    N = y;
    for i = 1:(n-1)
        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
```

"Horner Newton"

```
function p = HornerNewton (N, x, xi)
    n = length(N);
 p = N(n);
    for i = (n-1):-1:1
        p = p .* (xi - x(i)) + N(i);
    end
end
```

"b3interpolate"

```
function [alpha] = b3interpolate(y)
   dim = size(y, 1);

   % Populate Natural Boundary matrix
   M = zeros(dim + 2);
   M(1,1:3) = [1, -2, 1];
   M(dim+2, dim:end) = [1, -2, 1];
   for i = 1:dim
       M(i+1, i:i+2) = [1/6, 2/3, 1/6];
   end
   alpha = M \ [0; y; 0];
end
```

"spline_curve"

```
function [v] = spline_curve(alpha,x)
    v = zeros(size(x));
    % + 2 to compensate for i = 1 at the beginning
    for i = 1:size(alpha, 1)
       v = v + alpha(i) * B3(x-i+2);
    end
end
```

Published with MATLAB® R2020b