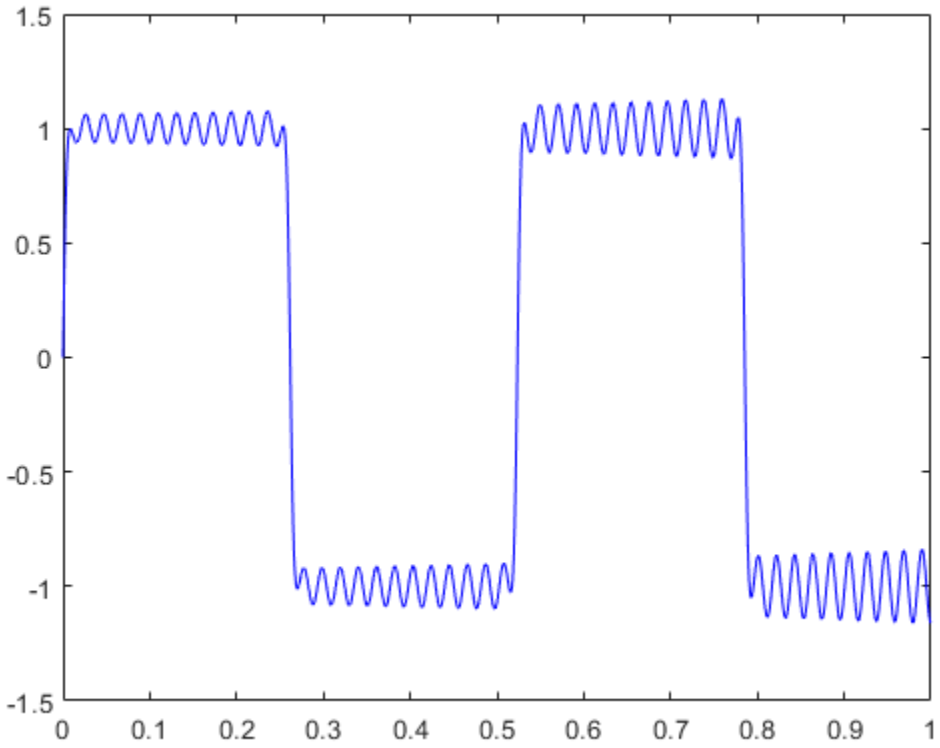
Table of Contents

a) 1
b) 1
c) 6

a)

```
clear all;
format long g;


x = [0+10^-12:1/2000:1-10^-12];
%x = [-10:1/2000:10];
plot(x, fun(x), 'b');
```

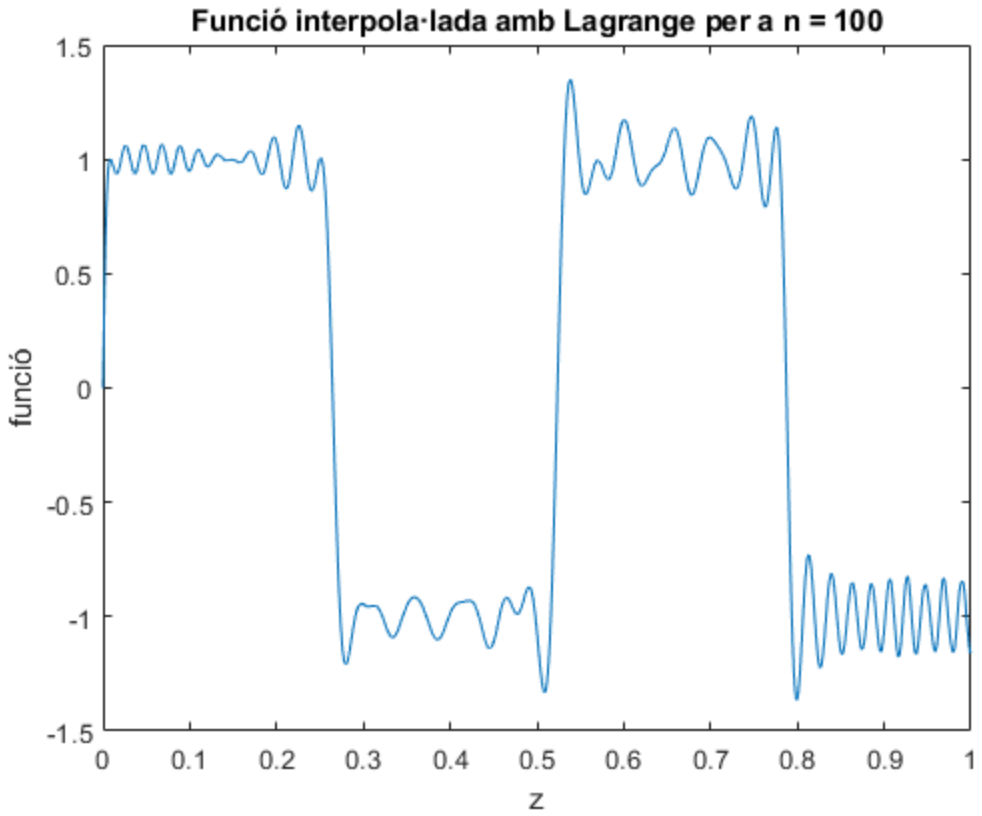


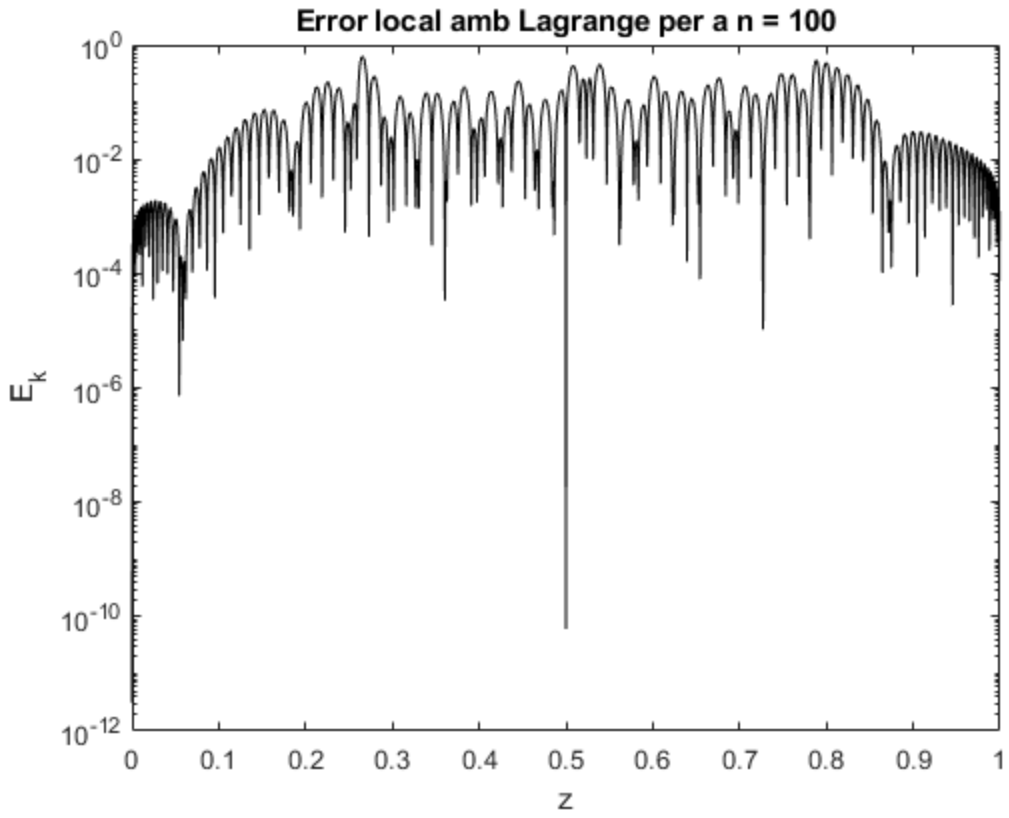
b)

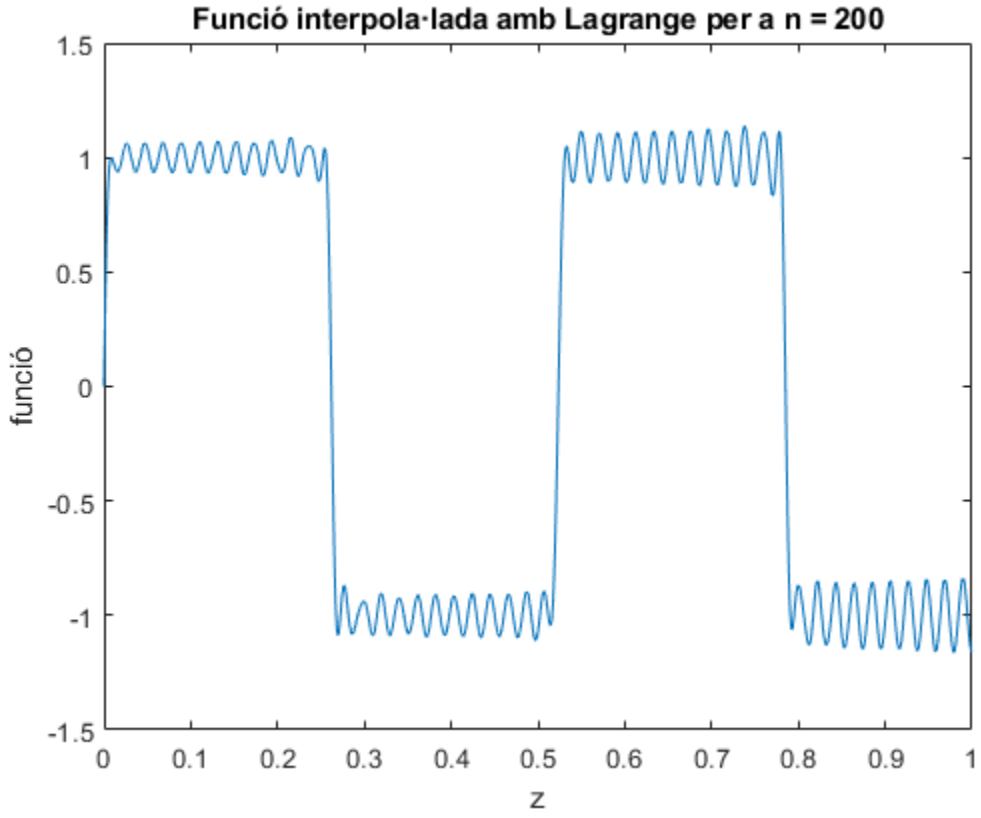
Vector z on avaluem els punts

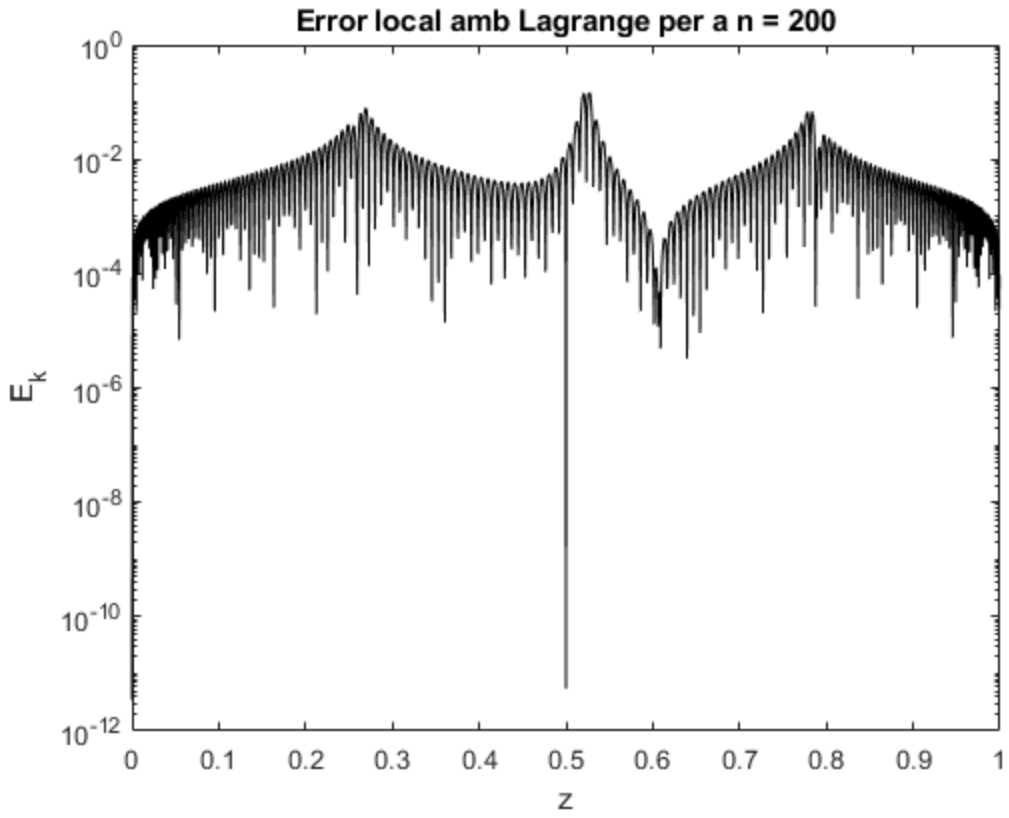
```
z = [0+10^-12:1/2000:1-10^-12];
```

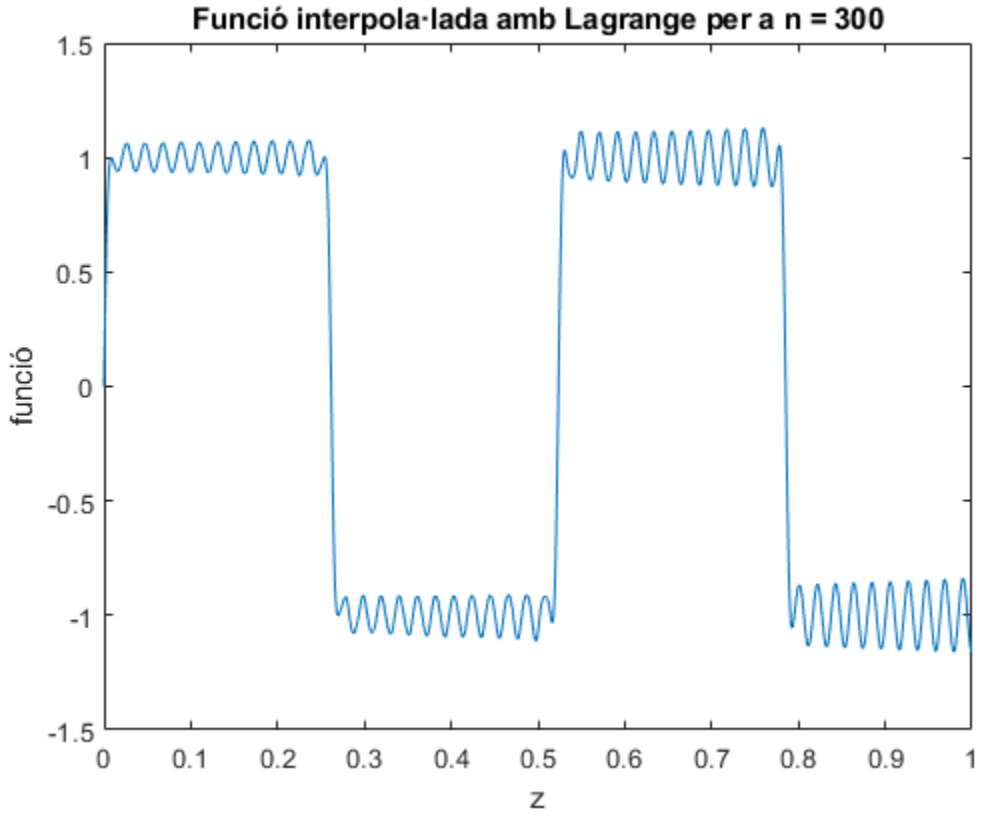
```
for n = 100:100:400
    j=[0:1:n];
    zj = cos((j*pi)/n);
    xj = (1/2)*(1+zj);
    b = baricentrica2(z, xj, fun(xj));
    figure;
    plot(z, b);
    titol_plot_int = sprintf('Funció interpola·lada amb Lagrange per a
 n = %d', n);
    title(titol_plot_int);
    xlabel('z');
    ylabel('funció');
    hold on;
    error = abs(fun(z)- b');
    figure;
    semilogy(z,error,'k');
    titol_plot_err = sprintf('Error local amb Lagrange per a n = %d',
 n);
    title(titol_plot_err);
    xlabel('z');
    ylabel('E_k')
    hold on;
end
hold off;
```

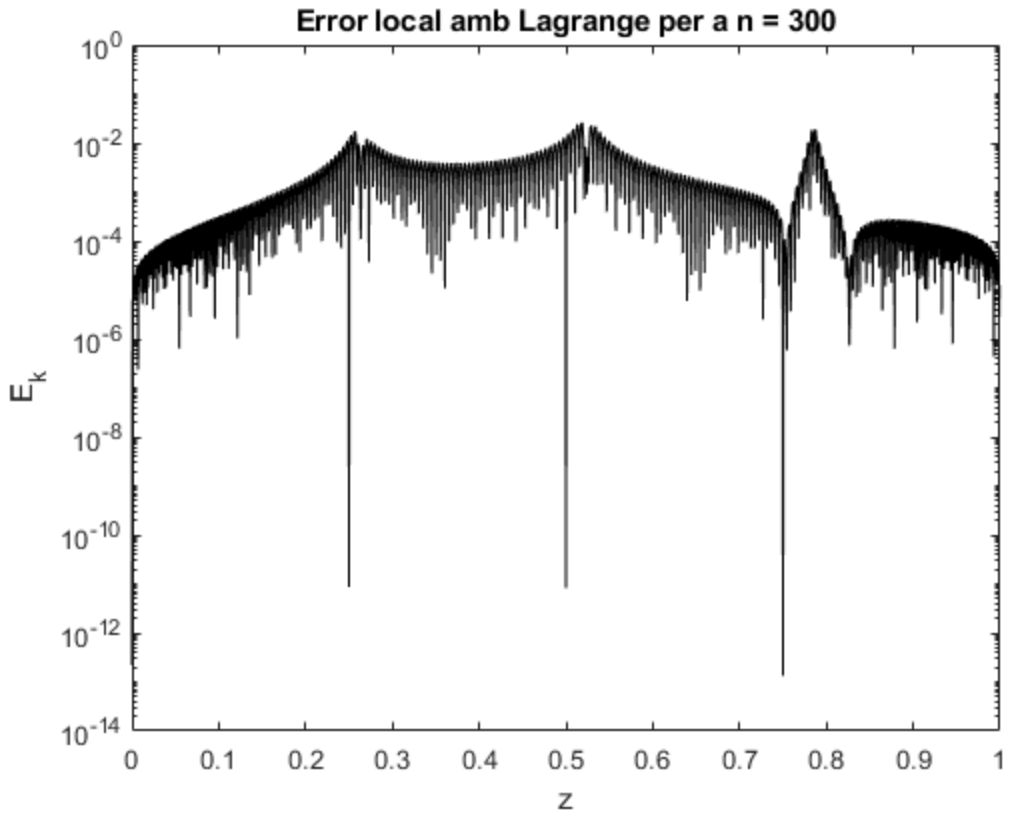


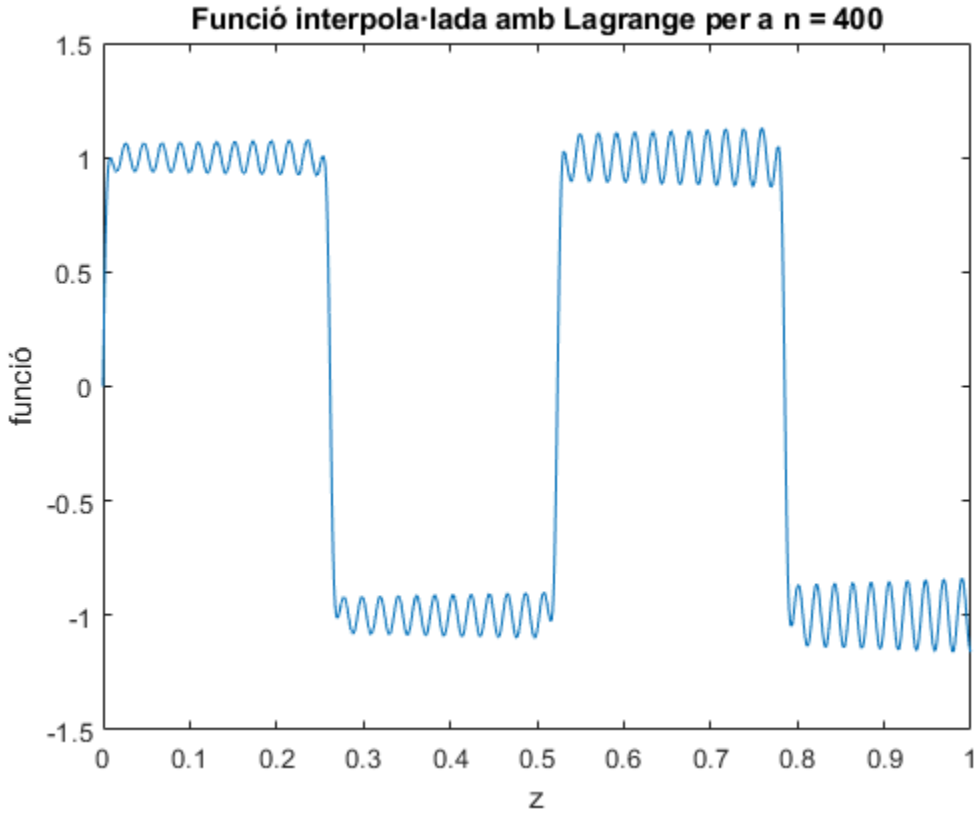

Error local amb Lagrange per a n = 100
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
E_k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z

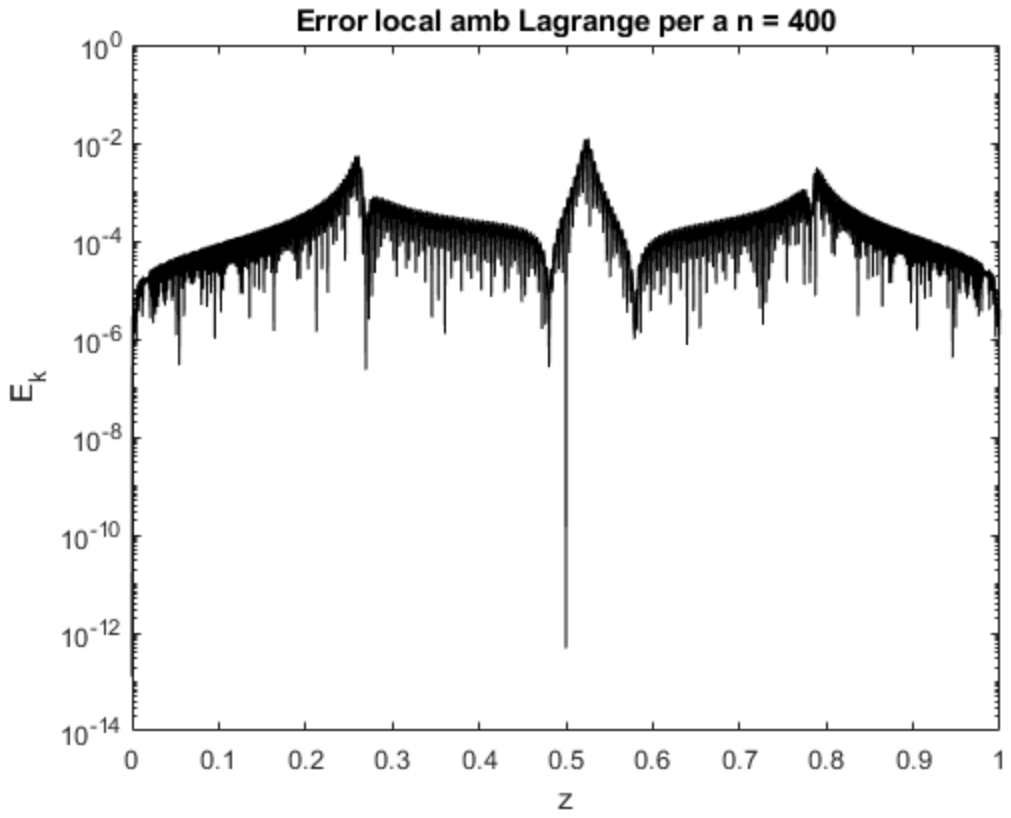

Funció interpola·lada amb Lagrange per a n = 200
1.5
1
0.5
0
-0.5
-1
-1.5
funció
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z


Error local amb Lagrange per a n = 200
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
E_k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z


Funció interpola·lada amb Lagrange per a n = 300
1.5
1
0.5
0
-0.5
-1
-1.5
funció
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z


Error local amb Lagrange per a n = 300
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
E_k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z


Funció interpola·lada amb Lagrange per a n = 400
1.5
1
0.5
0
-0.5
-1
-1.5
funció
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
z



c)

Per trobar el nombre de nodes necessari utilitzarem un bucle while amb un fucnionament similar a la biseccio per més precisió y velocitat.

```
clear all;
z = [0+10^-12:1/2000:1-10^-12];

salt = 1000;
n = salt; % Resolucio de la interpolacio.
prevN = 0;
maxN = 100000;
direccio = 1; % Saber si estabem augmentant o disminuint resolucio.
maxError = 10^-6;
errorInter = 1;

while n <= maxN && prevN ~= n
    n
    prevN = n;
    j=[0:1:n];
    zj = cos((j*pi)/n);
    xj = (1/2)*(1+zj);
    b = baricentrica2(z, xj, fun(xj));
    errorInter = max(abs(fun(z)- b'))
    %En cas que l'error sigui massa gran, cal augmentar resolució de
 les n:
```

```
    if errorInter > maxError
        if direccio == 1
            n = n + salt;
        elseif direccio == -1
            direccio = 1;
            % Divisio entera per 2:
            salt = fix(salt/2);
            n = n + salt;
        end
    % Si l'error ja es mes petit que el maxim d'error, vol dir que ens
 hem
    % passat amb la resolució i podem ajustar més.
    elseif errorInter < maxError
        if direccio == 1
            direccio = -1;
            % Divisio entera per 2:
            salt = fix(salt/2);
            n = n - salt;
        elseif direccio == -1
            n = n - salt;
        end
    end
end

n


n =

        1000


errorInter =

      4.92296884446741e-06


n =

        2000


errorInter =

      1.04753011820335e-11


n =

        1500


errorInter =
```

```
      3.76042352812078e-09

n =

        1000

errorInter =

      4.92296884446741e-06

n =

        1250

errorInter =

      1.75701686588869e-07

n =

        1125

errorInter =

      6.45875859017764e-07

n =

        1000

errorInter =

      4.92296884446741e-06

n =

        1062

errorInter =

      2.27551324699649e-06

n =
```

```
        1124

errorInter =

      9.96582830009585e-07

n =

        1093

errorInter =

      1.52671354625156e-06

n =

        1108

errorInter =

      9.09976264451551e-07

n =

        1101

errorInter =

      1.25480102436404e-06

n =

        1104

errorInter =

      7.74351891497105e-07

n =

        1103
```

```
errorInter =

      1.26267042938055e-06


n =

        1103
```

Published with MATLAB® R2018b