Contents

CS 375 HW 8

October 25th 2021 Juan A. Ormaza

```
clear all; clc; close all;
```

Problem 1

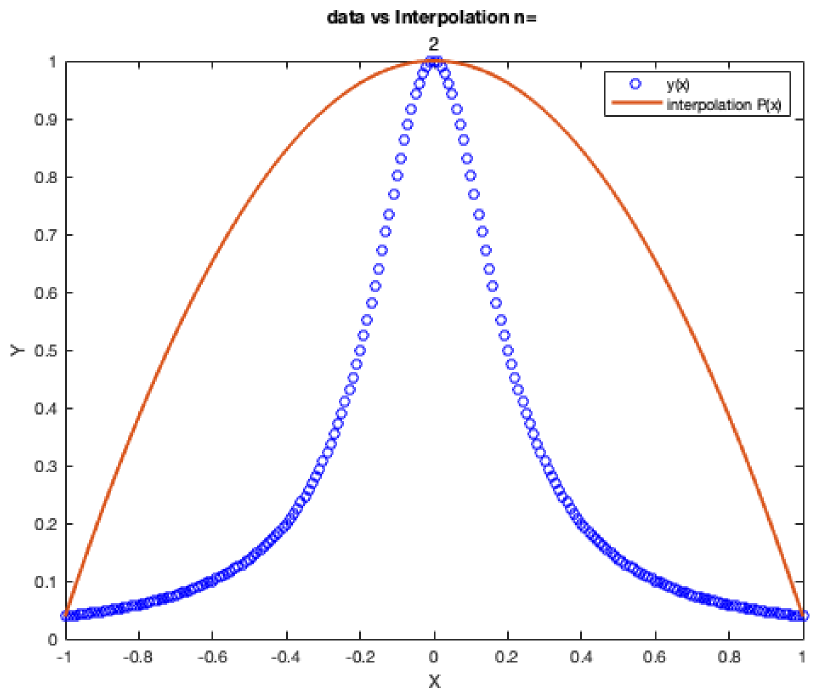
Part 1 written by hand (see attachments)

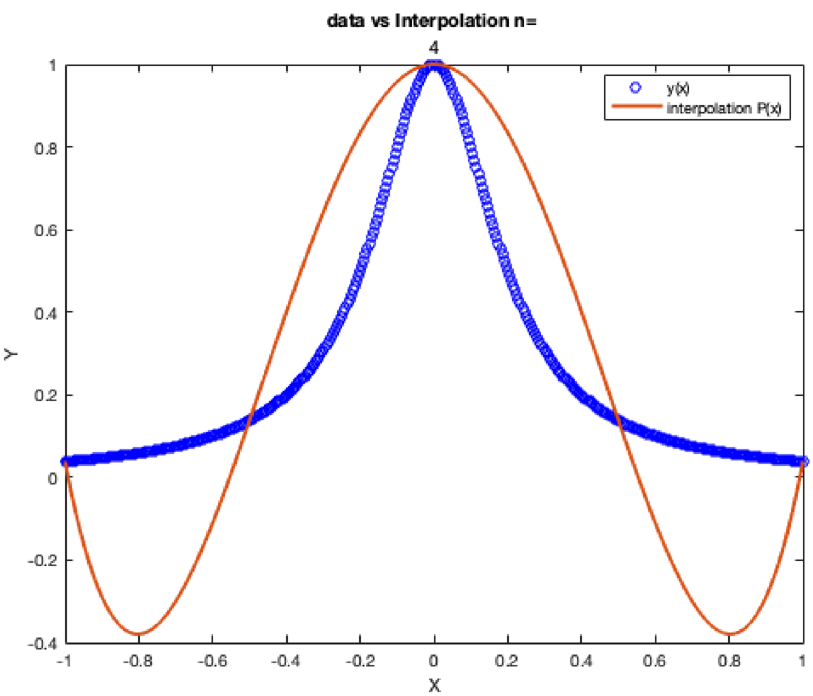
Problem 2

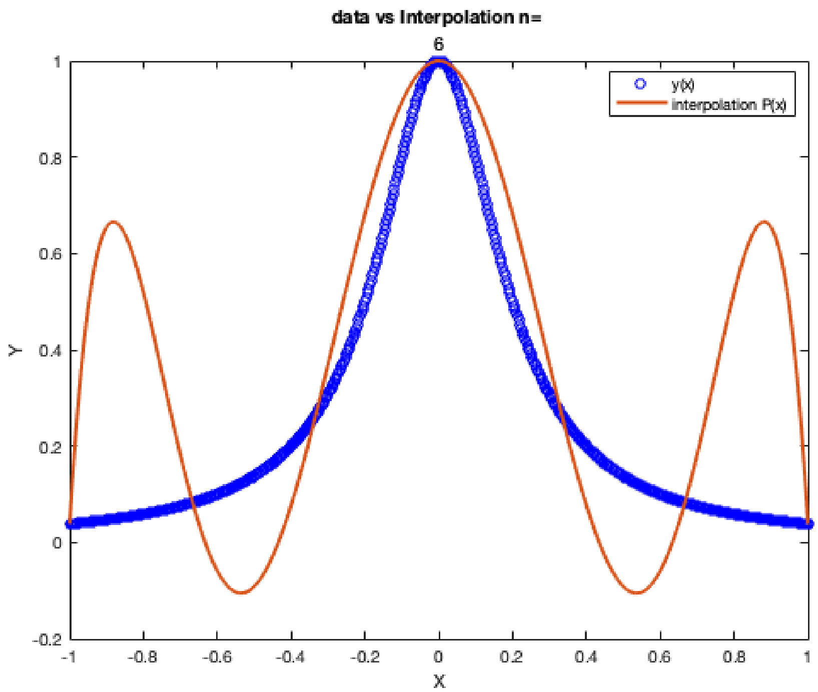
```
% Part a attached (see attachments);

% Part b attached (see attachments);

% Part C

x=@(i,n)-1+ i*(2/n);

f=@(x) 1./(1+25*x.^2);

n=2:2:20;

nfine=100*n;

for i=1:length(nfine)

    xValPoints=0:1:nfine(i);
    xIntPoints=0:1:n(i);

    xInt=x(xIntPoints,n(i));
    yInt=f(xInt);

    xVal=x(xValPoints,nfine(i));
    yVal=f(xVal);

    figure(i)
    plot(xVal,yVal,'bo','LineWidth',1);
    hold on;
    c=interp_monomials(xInt,yInt);
    c=rot90(c);
    c=rot90(c);
    plot(xVal,polyval(c,xVal),'LineWidth',2);
    xlabel('X');
    ylabel('Y');
    legend('y(x)','interpolation P(x)');
    title('data vs Interpolation n= ',num2str(n(i)));

end

% Part D

n_int=33;
N = n_int*100;
points = 0:1:N;
```

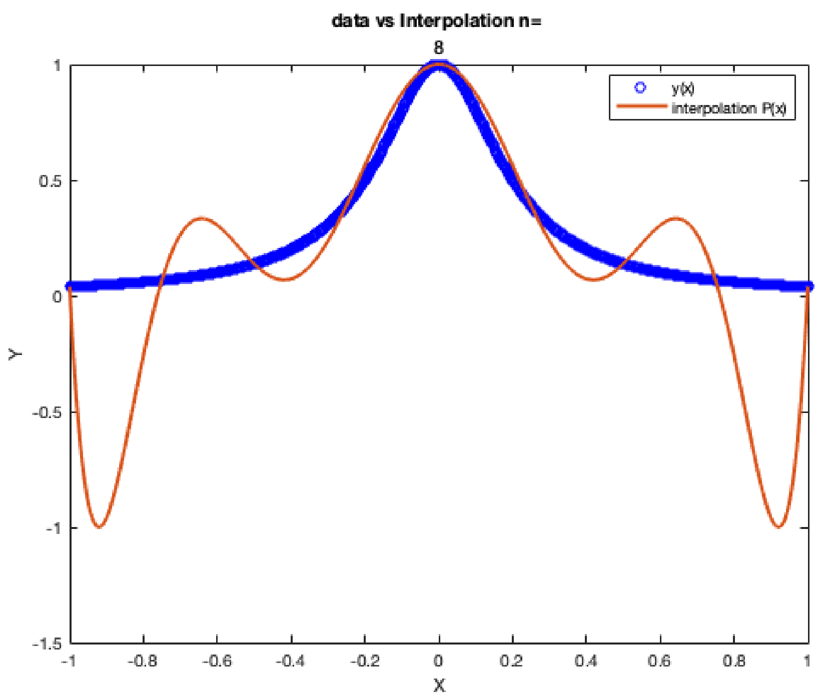
```
xIntPoints=0:1:n_int;

xInt=x(xIntPoints,n_int);
yInt=f(xInt);

xValues = x(points,N);
yValues = f(xValues);
figure(11)
plot(xValues,yValues,'bo','LineWidth',1);
hold on;
c=interp_monomials(xInt,yInt);
c=rot90(c);
c=rot90(c);
plot(xValues,polyval(c,xValues),'LineWidth',2);
xlabel('X');
ylabel('Y');
legend('y(x)','interpolation P(x)');
title('data vs Interpolation n= ',num2str(n_int));

fprintf("for n>3=3 it is possible to see how the interpolation fails to adjust to the values of f(x)\n")
fprintf("this, could be due to the size of the V matrix and the ill-conditioning of the solution.\n")
fprintf("moreover, the matrix is nearly singular or not scaled correctly at all.\n ")
```

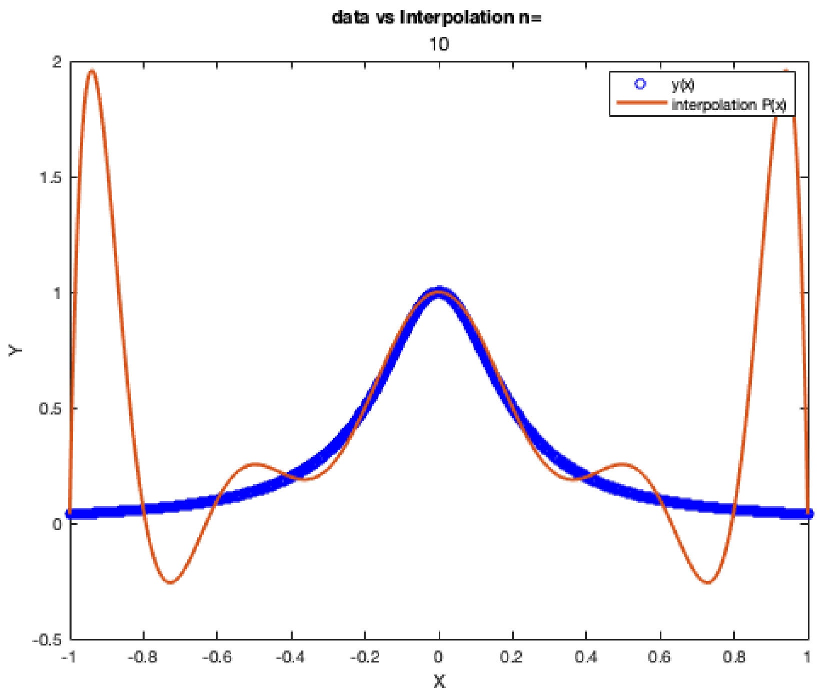
```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate. RCOND =  1.267771e-16.
for n>3=3 it is possible to see how the interpolation fails to adjust to the values of f(x)
this, could be due to the size of the V matrix and the ill-conditioning of the solution.
moreover, the matrix is nearly singular or not scaled correctly at all.
```

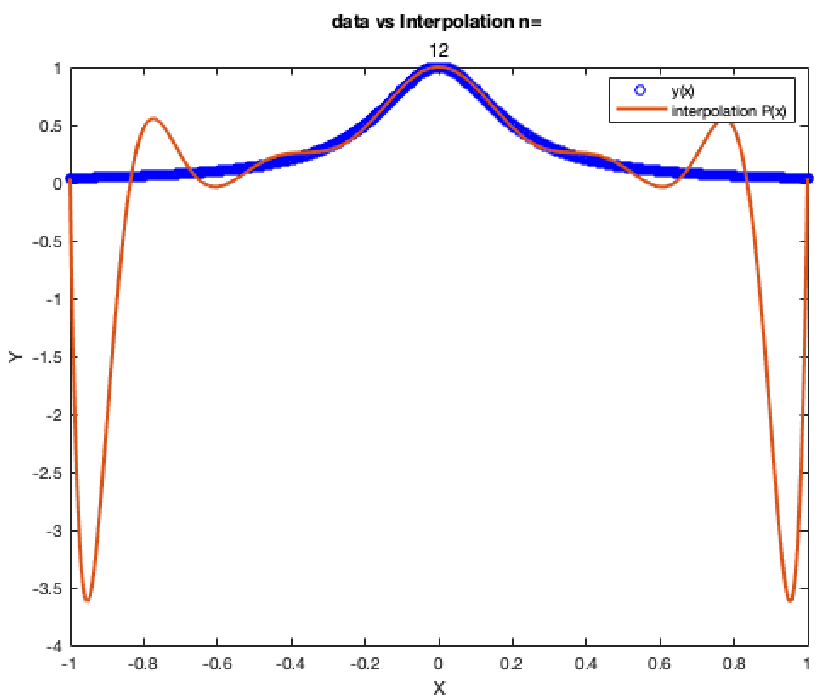

data vs Interpolation n=
2
y(x)
interpolation P(x)
Y
X

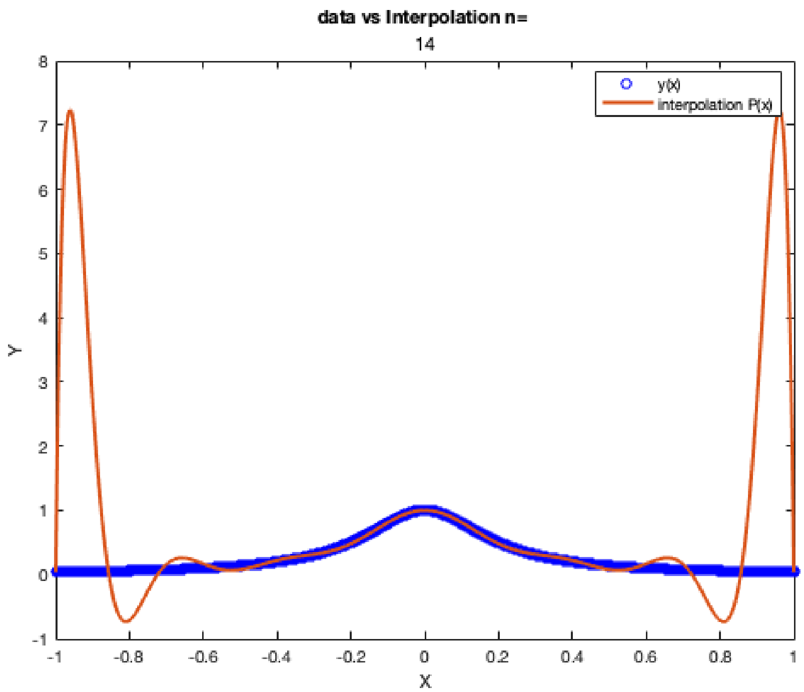

data vs Interpolation n=
4
y(x)
interpolation P(x)
Y
X

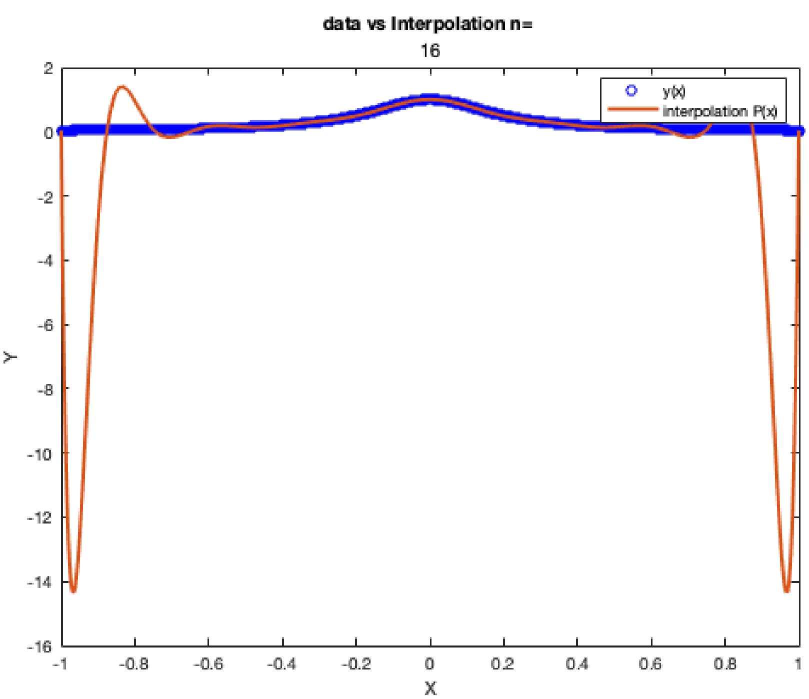

data vs Interpolation n=
6
y(x)
interpolation P(x)
Y
X
1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

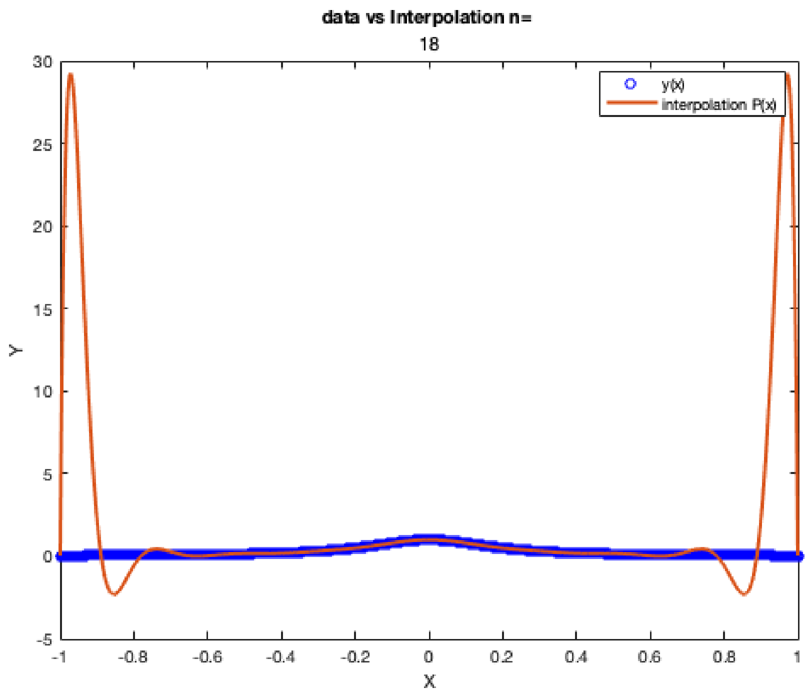

data vs Interpolation n=
8
y(x)
interpolation P(x)
Y
X
1
0.5
0
-0.5
-1
-1.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

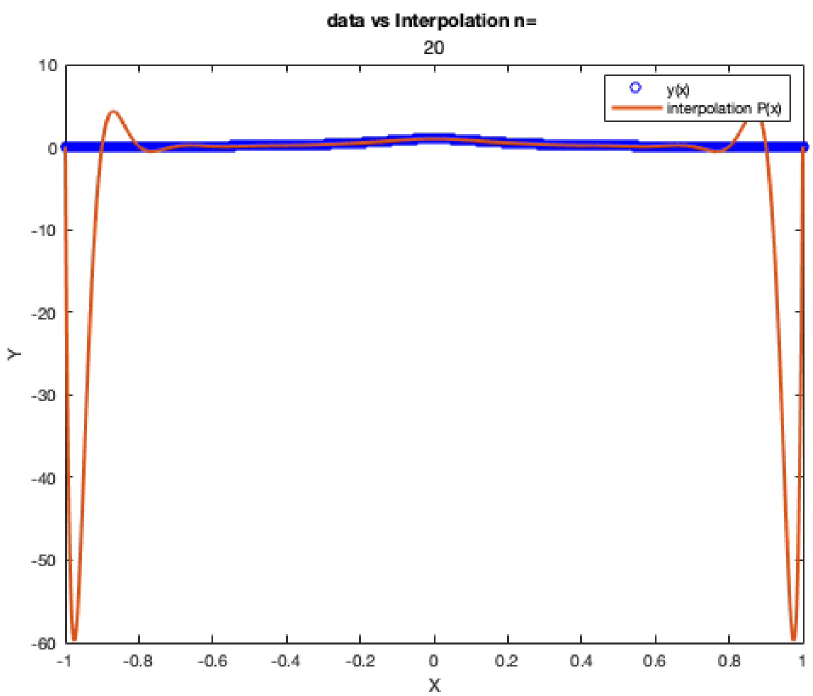

data vs Interpolation n=
10
y(x)
interpolation P(x)
Y
X

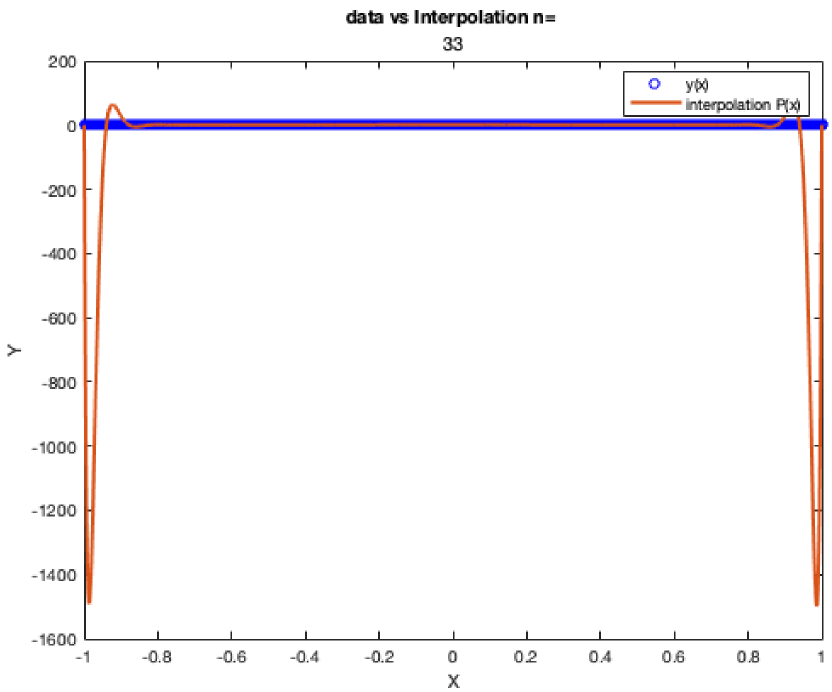

data vs Interpolation n=
12
y(x)
interpolation P(x)
Y
X


data vs Interpolation n=
14
y(x)
interpolation P(x)
Y
X


data vs Interpolation n=
16
y(x)
interpolation P(x)
Y
X


data vs Interpolation n=
18
y(x)
interpolation P(x)
Y
X

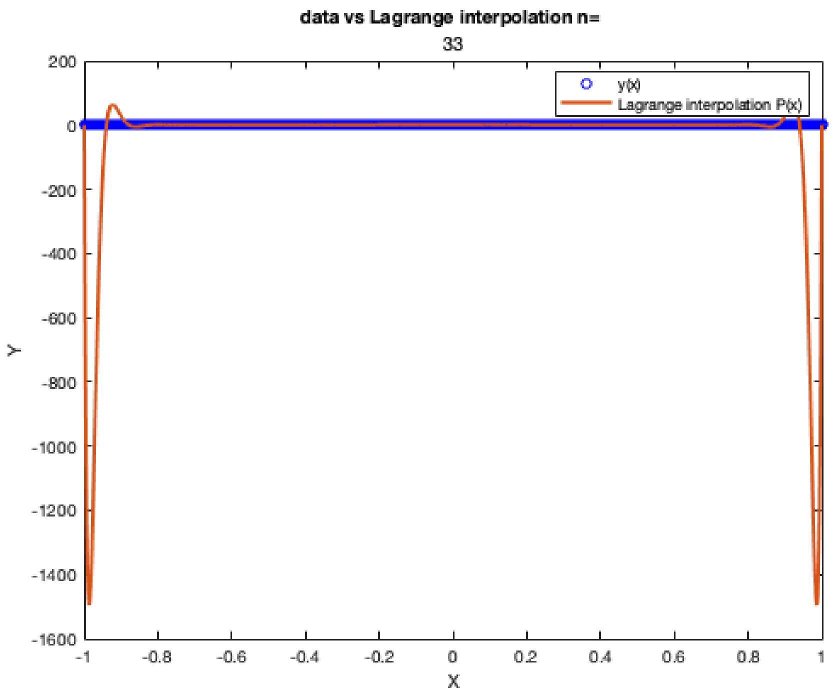

data vs Interpolation n=
20
y(x)
interpolation P(x)
Y
X



Problem 3

```
N = 33;
n_fine = 100*N;

xIntPoints=0:1:N;
xInt=x(xIntPoints,N);
yInt=f(xInt);

points = 0:1:n_fine;
xValues = x(points,n_fine);
yValues = f(xValues);

figure(12)
plot(xValues,yValues,'bo','LineWidth',1);
hold on;
intValues = zeros(1,n_fine);

for i=1:n_fine+1
    intValues(i)=Lagrange(xInt,yInt,xValues(i));
end
plot(xValues,intValues,'LineWidth',2);
xlabel('X');
ylabel('Y');
legend('y(x)','Lagrange interpolation P(x)');
title('data vs Lagrange interpolation n= ',num2str(N));

fprintf('as seen in figure 12, the same bad behavior near the ends is found in the lagrange interpolation\n')
fprintf('this is because lagrange interpolation also creates a polynomial that has to adjust to the curve.\n')
fprintf('moreover, the equispaced interpolation points make the error increase as seen in the graph.\n')
fprintf('therefore, higher ns do not increase accuracy of the interpolation\n')
```

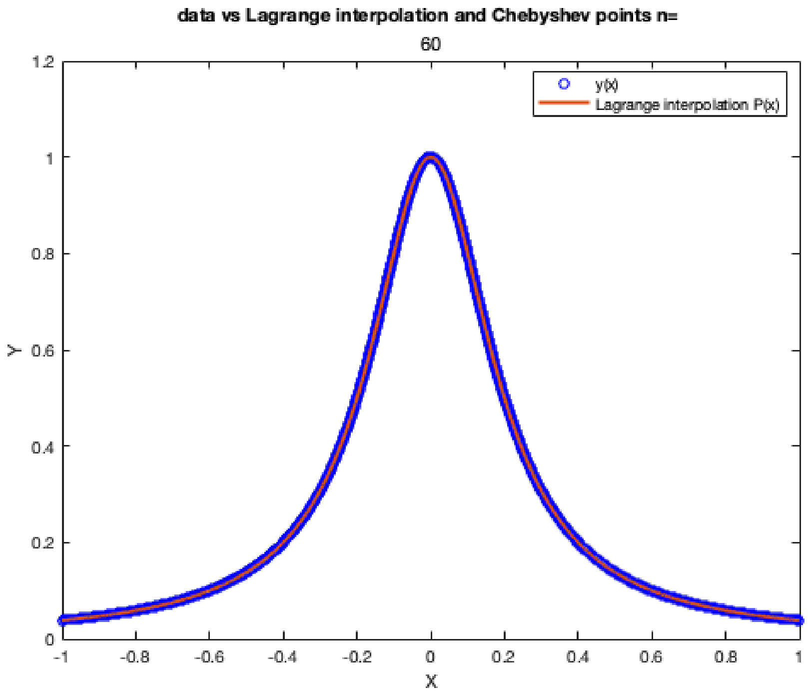
```
as seen in figure 12, the same bad behavior near the ends is found in the lagrange interpolation
this is because lagrange interpolation also creates a polynomial that has to adjust to the curve.
moreover, the equispaced interpolation points make the error increase as seen in the graph.
therefore, higher ns do not increase accuracy of the interpolation
```



Problem 4

```
x_cheby=@(i,n) cos(((2*i + 1)*pi)./(2*n+2));

N = 60;
n_fine = 100*N;

xIntPoints=0:1:N;
xInt=x_cheby(xIntPoints,N);
yInt=f(xInt);

points = 0:1:n_fine;
xValues = x(points,n_fine);
yValues = f(xValues);

figure(13)
plot(xValues,yValues,'bo','LineWidth',1);
hold on;
intValues = zeros(1,n_fine);

for i=1:n_fine+1
    intValues(i)=Lagrange(xInt,yInt,xValues(i));
end

plot(xValues,intValues,'LineWidth',2);
xlabel('X');
ylabel('Y');
legend('y(x)','Lagrange interpolation P(x)');
title('data vs Lagrange interpolation and Chebyshev points n= ',num2str(N));


fprintf('As seen in this example chebyshev points, in comparison to equispaced one do converge\n')
fprintf('and there is a limit to the error the interpolation can have. Because of this, the interpolation\n')
fprintf('behaves much better at the ends and will keep on improving as n increases.\n')
fprintf('to show this, I used an n that is almost double the one in previous examples\n')
```

```
As seen in this example chebyshev points, in comparison to equispaced one do converge
and there is a limit to the error the interpolation can have. Because of this, the interpolation
behaves much better at the ends and will keep on improving as n increases.
to show this, I used an n that is almost double the one in previous examples
```



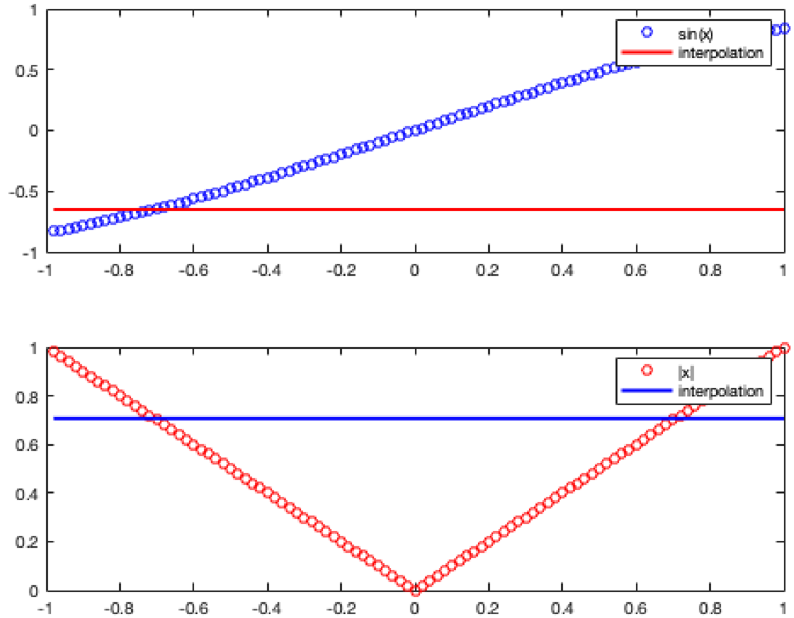
Problem 5

```
f1=@(x) sin(x);
f2=@(x) abs(x);

x_cheby=@(i,n) cos(((2*i + 1)*pi)./(2*n+2));

n_array=1:1:16;

fprintf('Error\t sin(x)\t\t abs(x)\n');

for i=1:length(n_array)
    n_fine=n_array(i)*100;

    intPoints = 1:1:n_array(i);

    x_int=x_cheby(intPoints,n_array(i));
    y_int1=f1(x_int);
    y_int2=f2(x_int);

    plotPoints = 1:1:n_fine;
    xValues = x(plotPoints,n_fine);
    y1Values = f1(xValues);
    y2Values = f2(xValues);

    sin_int=zeros(1,n_fine);
    abs_int=zeros(1,n_fine);

    for j=1:n_fine
        sin_int(j)=Lagrange(x_int,y_int1,xValues(j));
        abs_int(j)=Lagrange(x_int,y_int2,xValues(j));
    end


    figure(13+i)
    subplot(2,1,1)
    plot(xValues,y1Values,'bo');
    hold on
    plot(xValues,sin_int,'r','LineWidth',2);
    title('data vs Lagrange interpolation and Chebyshev points n= ',num2str(n_array(i)));
    legend('sin(x)','interpolation');
    subplot(2,1,2)
    plot(xValues,y2Values,'ro');
```

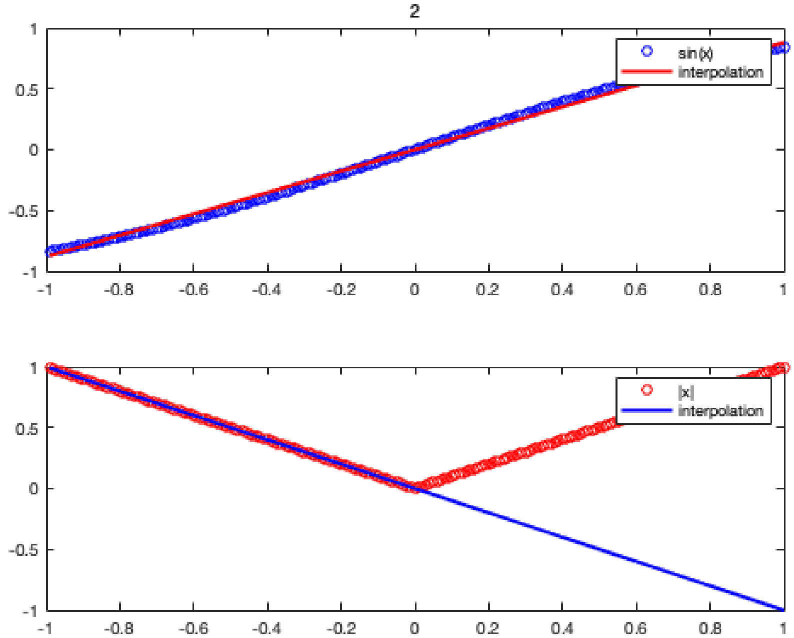
```
    hold on
    plot(xValues,abs_int,'b','LineWidth',2);
    legend('|x|','interpolation')

    fprintf(' n=%2.0f\t %10.10f\t %10.10f\n',n_array(i),max(abs(y1Values-sin_int)),max(abs(y2Values-abs_int)));

end

fprintf('\n\nAs seen from the plots, it is possible to notice that the interpolation adjusts\n')
fprintf('correctly to sin(x), but has trouble adjusting to abs(x). One main reason of this is because\n')
fprintf('|x| is not a continuous function, thus, the interpolation polynomial will have trouble interpolation |x|\n')
fprintf('this can be seen at the endpoints of the plots where the interpolation suddenly increases or decreases\n')
```

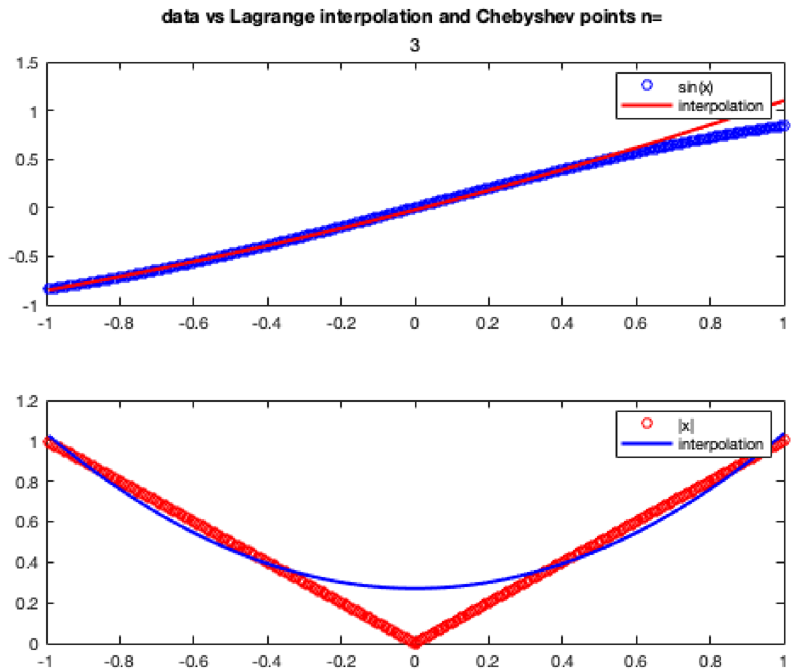
```
Error    sin(x)          abs(x)
 n= 1    1.4911079239    0.7071067812
 n= 2    0.0396232083    2.0000000000
 n= 3    0.2592932162    0.2705980501
 n= 4    0.0005043694    1.4249084145
 n= 5    0.0073679265    0.1725460301
 n= 6    0.0000030235    1.3408569429
 n= 7    0.0000784384    0.1274488948
 n= 8    0.0000000105    1.3119484318
 n= 9    0.0000004273    0.1012465126
 n=10    0.0000000000    1.2984369017
 n=11    0.0000000014    0.0840524134
 n=12    0.0000000000    1.2909913934
 n=13    0.0000000000    0.0718805412
 n=14    0.0000000000    1.2864385142
 n=15    0.0000000000    0.0628024108
 n=16    0.0000000000    1.2834459799


As seen from the plots, it is possible to notice that the interpolation adjusts
correctly to sin(x), but has trouble adjusting to abs(x). One main reason of this is because
|x| is not a continuous function, thus, the interpolation polynomial will have trouble interpolation |x|
this can be seen at the endpoints of the plots where the interpolation suddenly increases or decreases
```

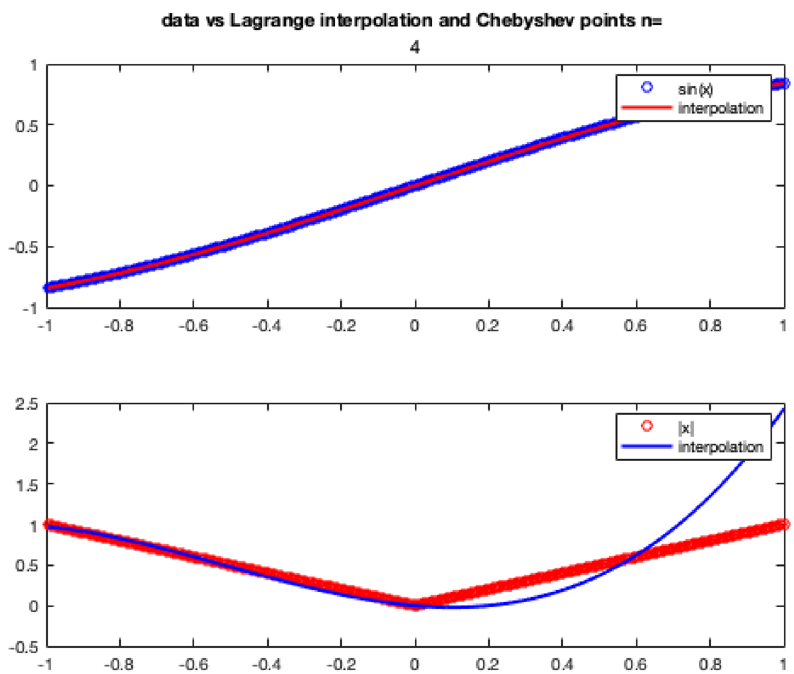
data vs Lagrange interpolation and Chebyshev points n=
1

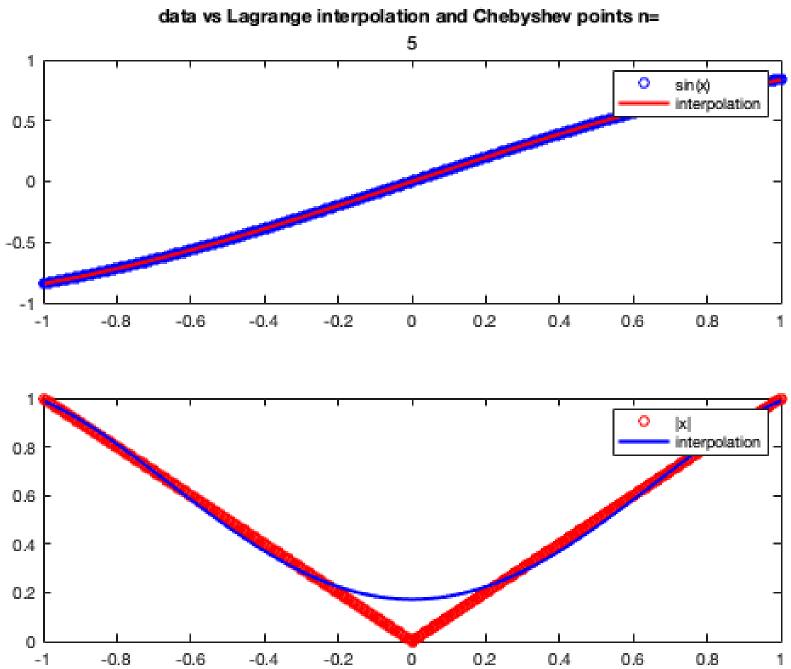


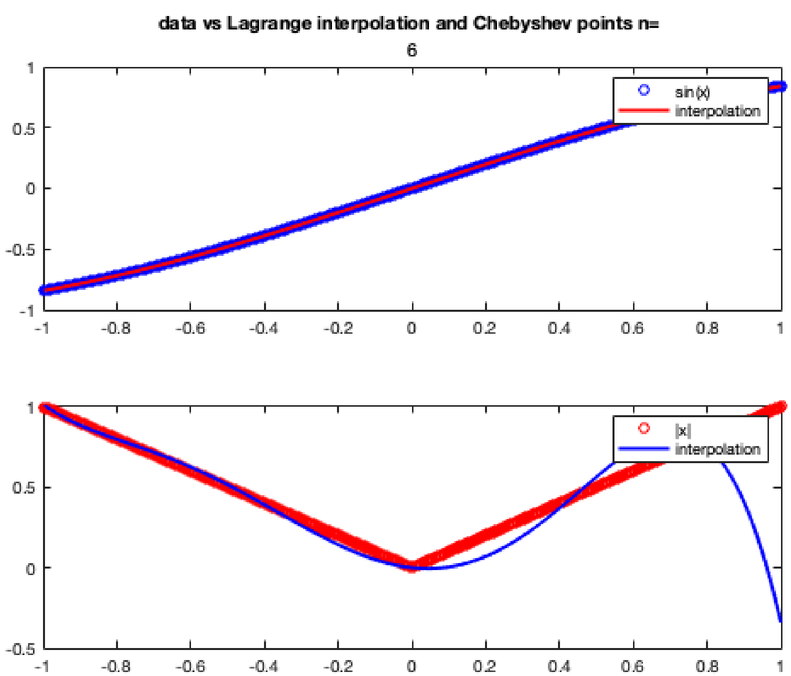
data vs Lagrange interpolation and Chebyshev points n=
2

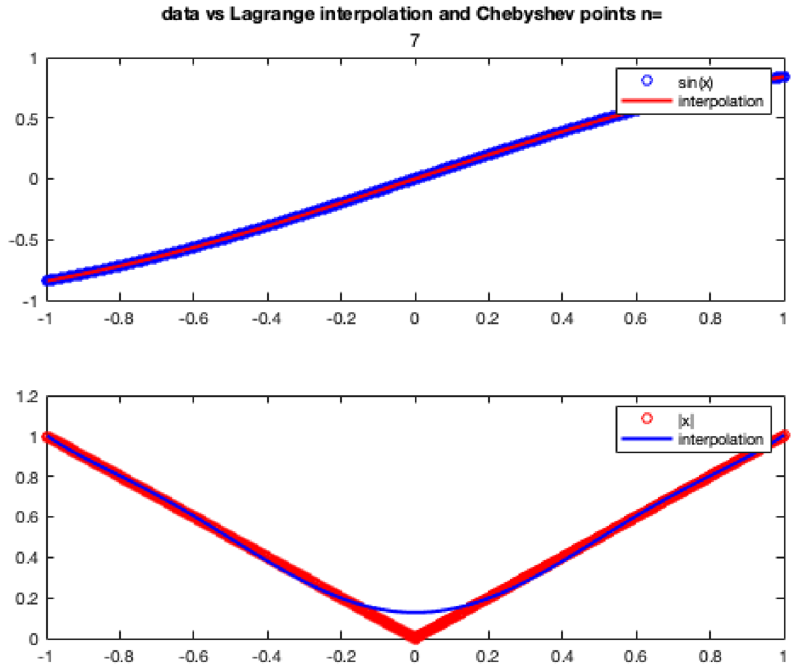


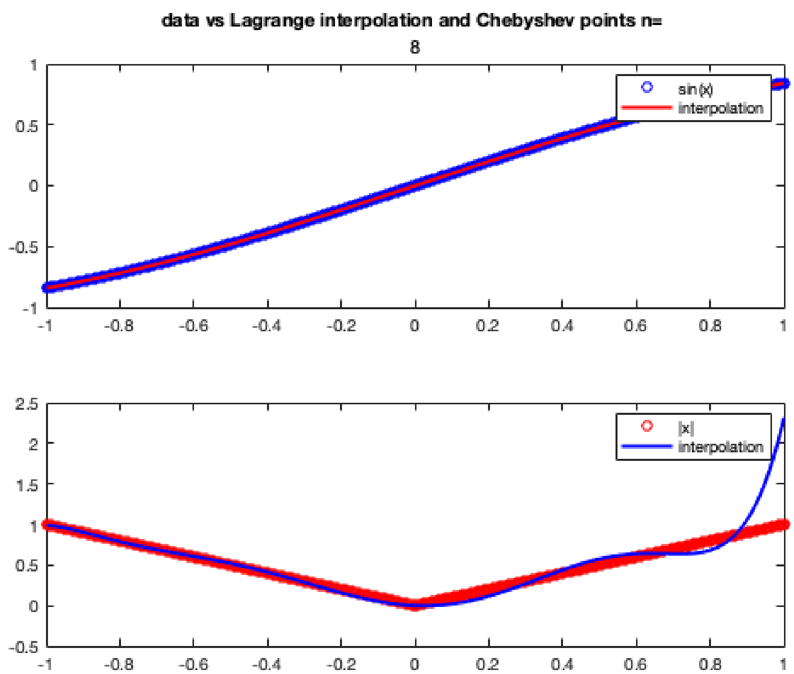

data vs Lagrange interpolation and Chebyshev points n=
3
sin(x)
interpolation
|x|
interpolation


data vs Lagrange interpolation and Chebyshev points n=
4
sin(x)
interpolation
|x|
interpolation

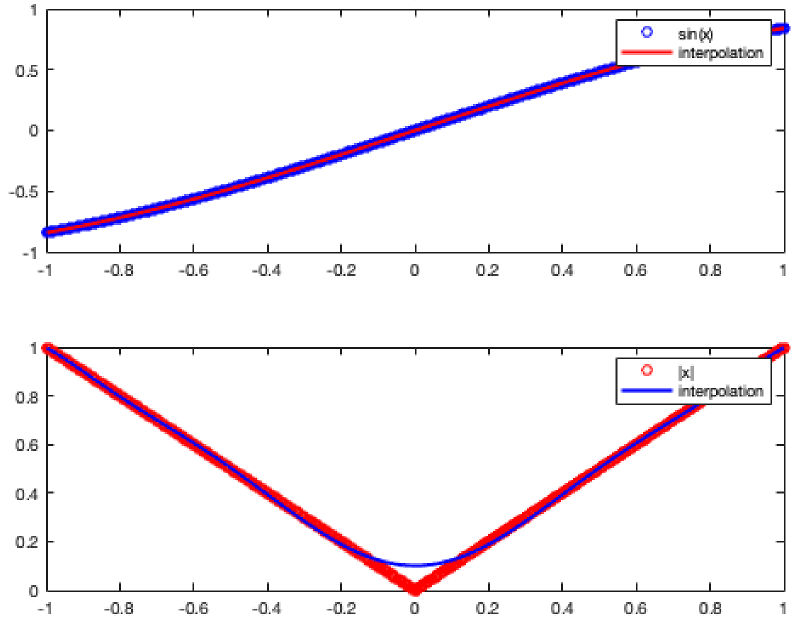

data vs Lagrange interpolation and Chebyshev points n=
5
sin(x)
interpolation
|x|
interpolation


data vs Lagrange interpolation and Chebyshev points n=
6
sin(x)
interpolation
|x|
interpolation

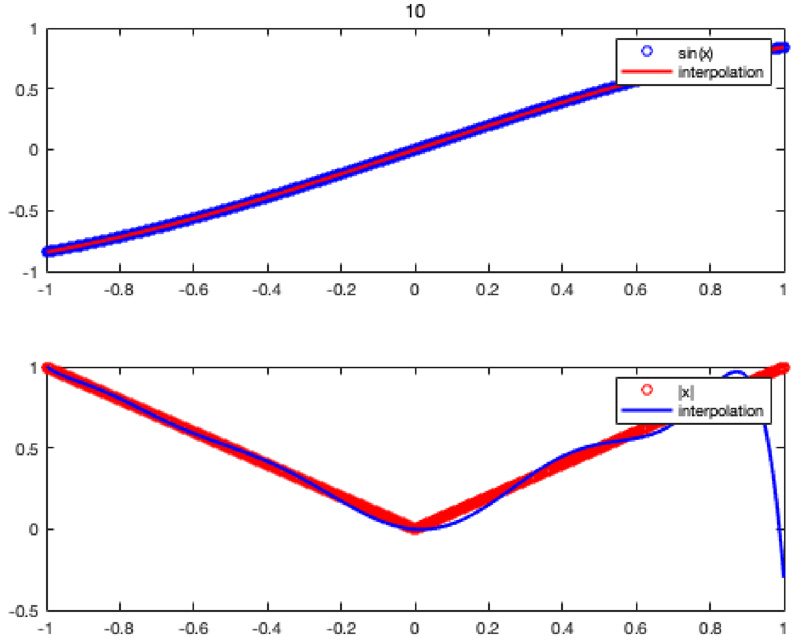

data vs Lagrange interpolation and Chebyshev points n=
7
sin(x)
interpolation
|x|
interpolation

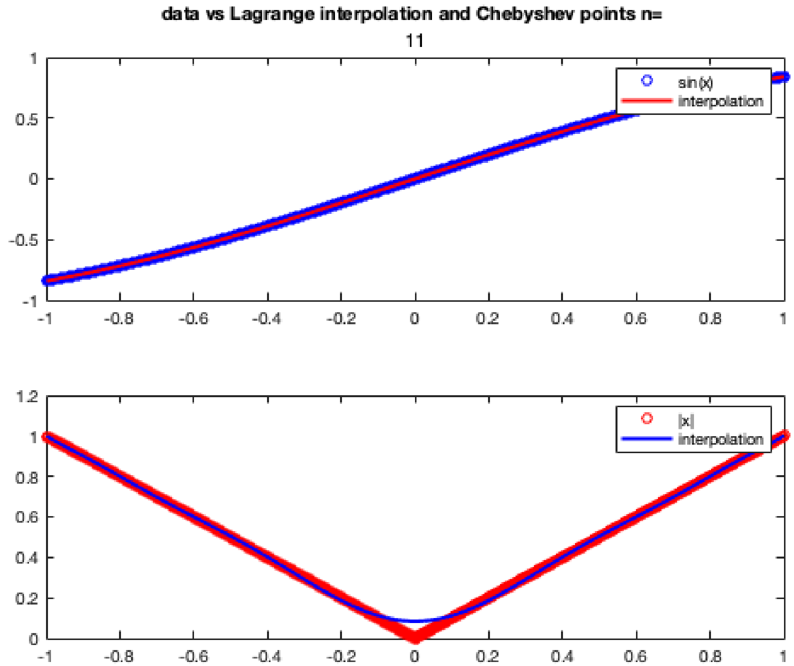

data vs Lagrange interpolation and Chebyshev points n=
8
sin(x)
interpolation
|x|
interpolation

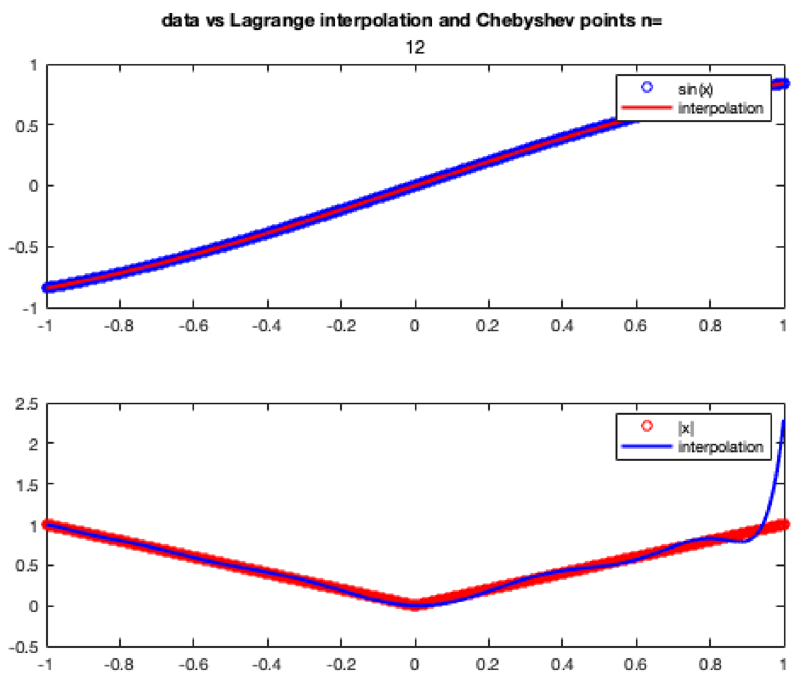
data vs Lagrange interpolation and Chebyshev points n=
9

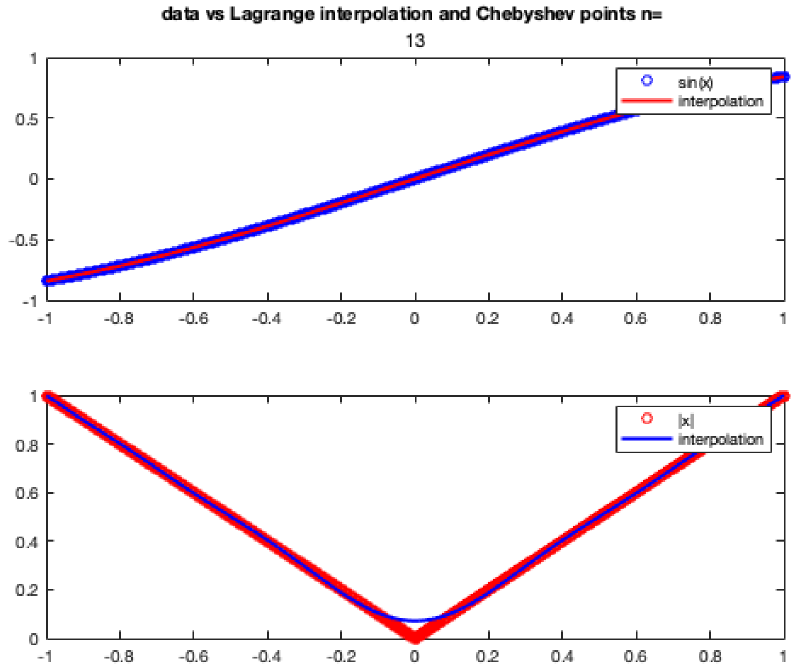


data vs Lagrange interpolation and Chebyshev points n=
10

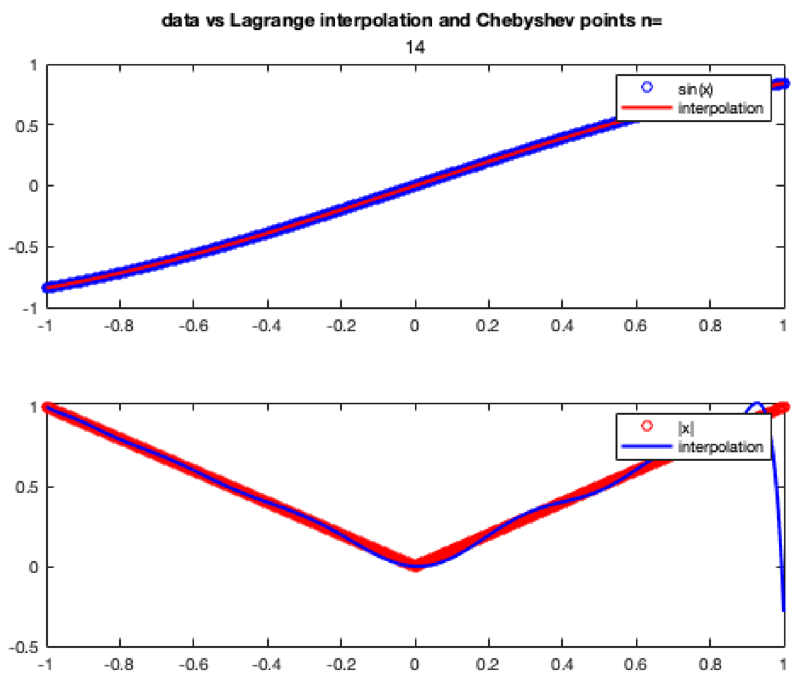



data vs Lagrange interpolation and Chebyshev points n=
11
sin(x)
interpolation
|x|
interpolation

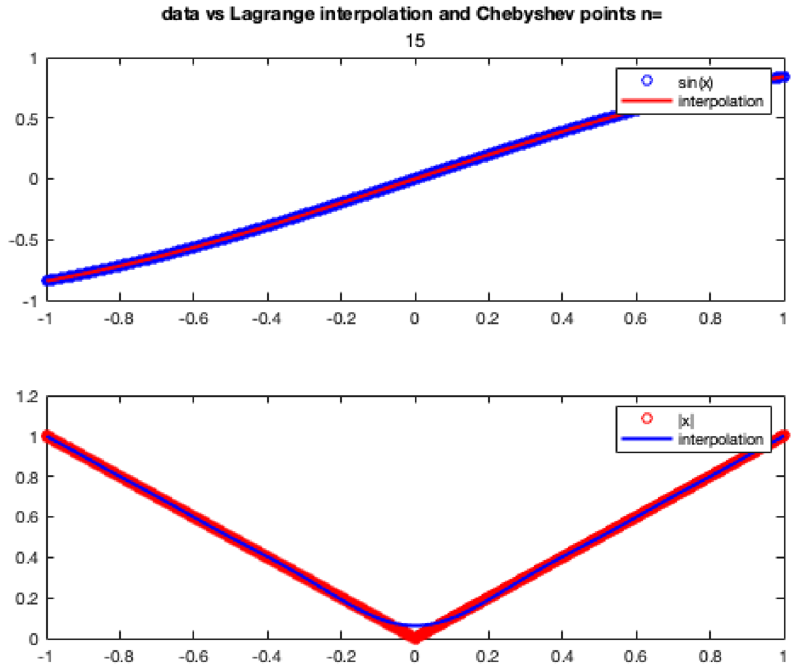

data vs Lagrange interpolation and Chebyshev points n=
12
sin(x)
interpolation
|x|
interpolation

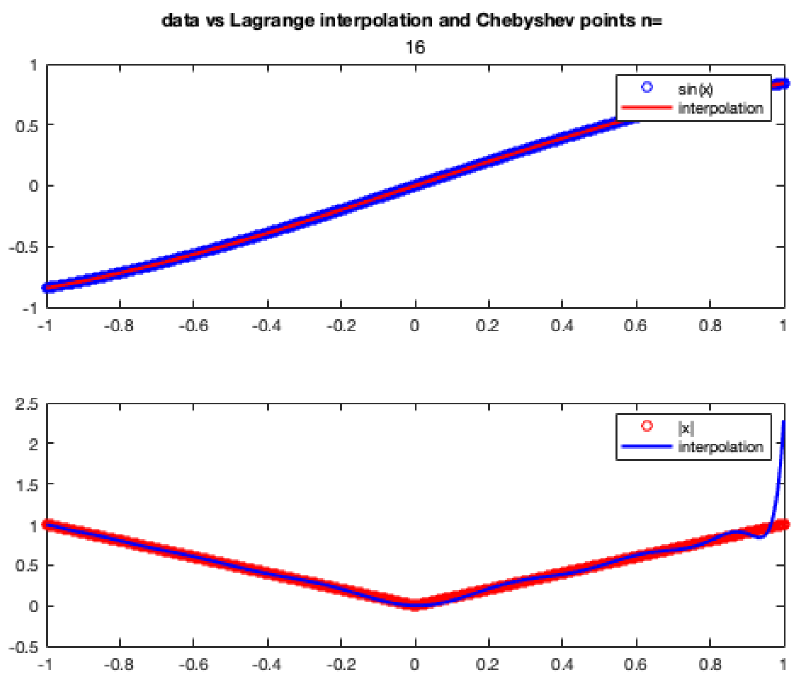


data vs Lagrange interpolation and Chebyshev points n=
13
sin(x)
interpolation
|x|
interpolation



data vs Lagrange interpolation and Chebyshev points n=
14
sin(x)
interpolation
|x|
interpolation





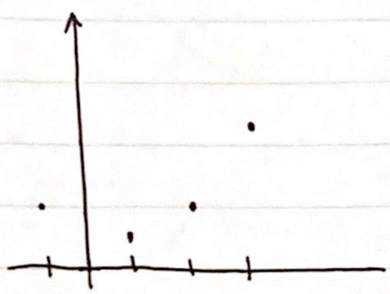
Published with MATLAB® R2021a

Juan Alejandro Ormaza   October 25^{th} 2021

HW8

1. Polynomial interpolation

How many degree d polynomials pass through $(-1,3), (1,1), (2,3), (3,7)$, for $d = 2, 3, 6$? If at least one polynomial passes through these points, write it down. If no such polynomial exists, explain why.



per theorem:

if points $x_0, \dots, x_n$ are distinct, then for arbitrary $y_0, \dots y_n$, there is a unique polynomial $p(x)$ of degree at most n such that $p(x_i) = y_i$ for $i = 0, \dots, n$

$\therefore$ for $n+1$ points $= 4$ $n = 3$ and $p(x)$ is of degree at most 3.

Therefore $d = 2$ and $d = 3$ are possible polynomials.

we can use the Vandermonde Matrix to visualize this.

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix}}_{\text{Degree 3 Polynomial}} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \left.\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}\right\} \text{4 elements}$$

2.2 By hand problem Assume (x_j, y_j), $j = 0, \dots, n$ are given. Derive the system $Va = y$ that determines the coefficients $a = [a_0, \dots a_n]^T$ (here $y = [y_0, y_1, \dots, y_n]^T$), that is, find the structure (i.e., general form) of V.

for each point

y_j we can assume a polynomial $p(x)$ that when evaluated at x_j yields y_j

$\therefore$

$$a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n = y_0$$
$$a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n = y_1$$
$$\vdots \qquad\qquad \vdots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n = y_n$$

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & & \ddots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{y}$$

```
% Juan Alejandro Ormaza
% October 25 2021

function [c] = interp_monomials(x,y)

n=length(x)-1;
V=zeros(n+1,n+1);

for i=1:n+1
    for j=1:n+1
        if(j==1)
            V(i,j)=1;
        else
            V(i,j)=x(i)^(j-1);
        end
    end
end

c=V\y';



end
```

```
Not enough input arguments.

Error in interp_monomials (line 6)
n=length(x)-1;
```

Published with MATLAB® R2021a

```
% Juan Alejandro Ormaza
% October 25 2021

function [y] = Lagrange(X,Y,desiredX)
%Lagrange returns the value P(X) of a polynomial P that interpolates
% the data from X and Y. desiredX is an input that will calculate
% y = P(desiredX).

n=length(X);
l=zeros(1,n);

for k=1:n
    l(k)=1;
    for i=1:n
        if i~=k
            l(k)=l(k)*(desiredX-X(i))/(X(k)-X(i));
        end
    end
end

y=sum(l.*Y);




end
```

```
Not enough input arguments.

Error in Lagrange (line 9)
n=length(X);
```

Published with MATLAB® R2021a