Math 104A: Homework 2

Raghav Thirumulu, Perm 3499720
rrajuthirumulu@umail.ucsb.edu

July 13, 2018

1. (a) We must construct the basis polynomials first.

$$L_{2,0}(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{x^2-4x+3}{3}$$
$$L_{2,1}(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0)(x-3)}{(1-0)(1-3)} = \frac{x^2-3x}{-2}$$
$$L_{2,2}(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{x^2-x}{6}$$

Now use the Lagrange interpolation formula to find $P_2(x)$

$$\begin{aligned} P_2(x) &= f(x_0)L_{2,0}(x) + f(x_1)L_{2,1}(x) + f(x_2)L_{2,2}(x) \\ &= 1(\frac{x^2-4x+3}{3}) + 1(\frac{x^2-3x}{-2}) - 5(\frac{x^2-x}{6}) \\ &= -x^2+x+1 \end{aligned}$$

(b) Solve for $P_2(x)$ where $x = 2$

$$P_2(2) = -(2)^2 + 2 + 1 = -1$$

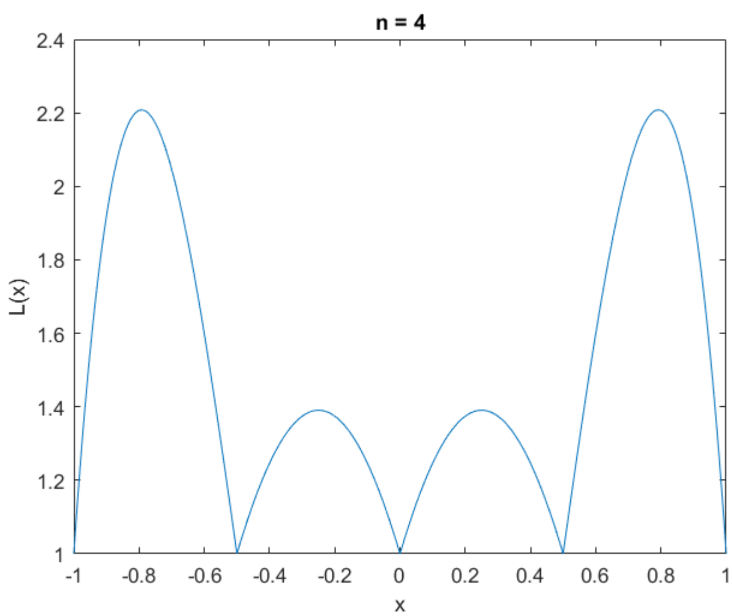
2. (a)

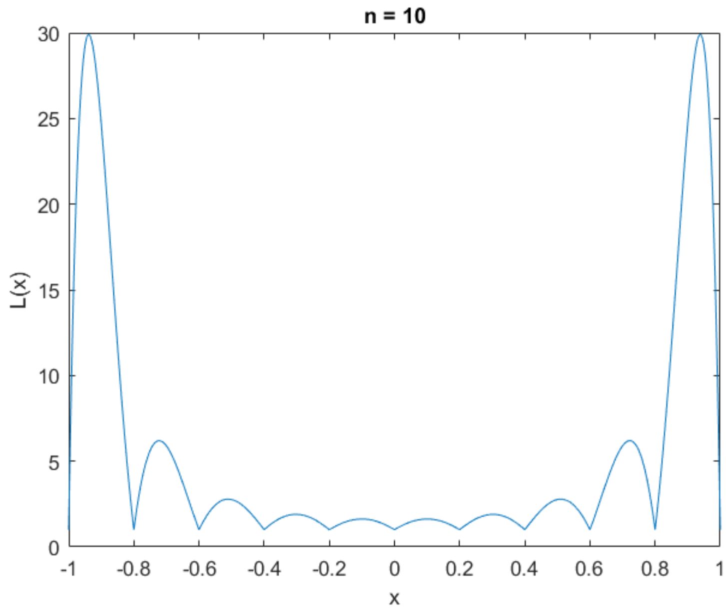
```
function T = lebesgue(x,xbar)
% Computer code for evaluating Lebesque function
% Input:  x     --- nodes for interpolation;
%         xbar  --- point for evaluation;
% Output: T     --- the evaluation of the Lebesgue function
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/10/2018

% Store length of x in n, Create 1xn array of ones for storing values
n=length(x);
L=ones(1,n);

% Iterate through the interpolation points
for i=1:n
    for j=1:n
        if j~=i
            L(i) = L(i) * ( (xbar-x(j)) / (x(j)-x(i)) ) ;
        end
    end
end
```

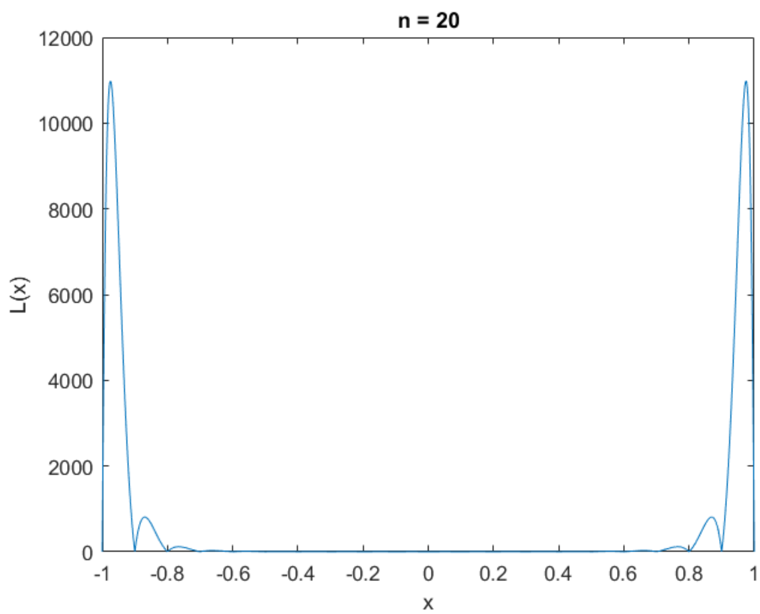
```
% Take the absolute value of the sum of the elements of L
T = sumabs(L);

end
```

(b)

```
function F = lebesgue_run(n)
% Computer code for calling lebesque.m and finding the constant
% based on number of evaluation points
% Input:  n     --- number of evaluation points;
% Output: F     --- the Lebesgue constant
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/10/2018

% Create a row vector for storing interpolation points
x=zeros(1,n+1);

% Find interpolation points and store in x using given formula
for j=1:n+1
    x(j)=-1+(j-1)*(2/n);
end

% Adjust parameter for plotting resolution
m=1000;
T=zeros(1,m+1);
xbar=zeros(1,m+1);

% Find points for xbar through iteration with plotting resolution
for k=1:m+1
    xbar(k)=-1+(k-1)*(2/m);
    T(k)=lebesgue(x,xbar(k));
end

% Plot the Lebesgue function
plot(xbar,T);
xlabel('x');
ylabel('L(x)');
title(['n = ' num2str(n)]);

% Find the norm using the data, F will be Lebesgue constant
F=norm(T,Inf);
end
```





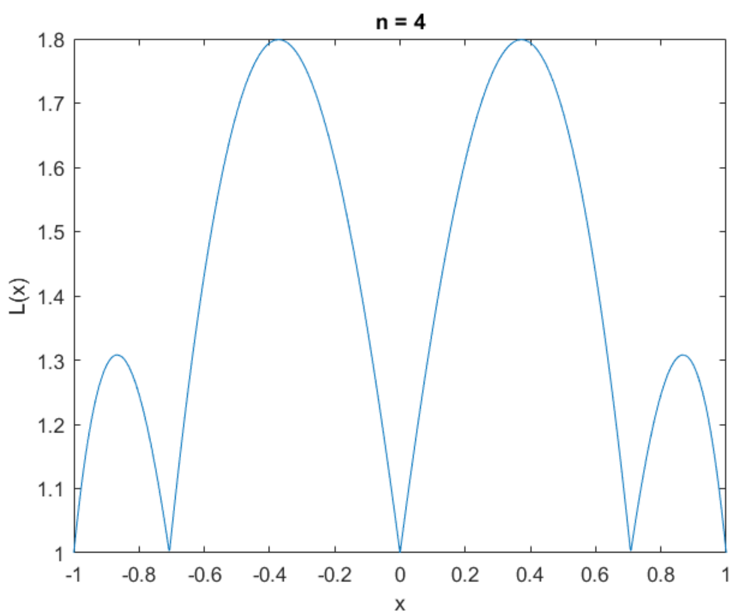


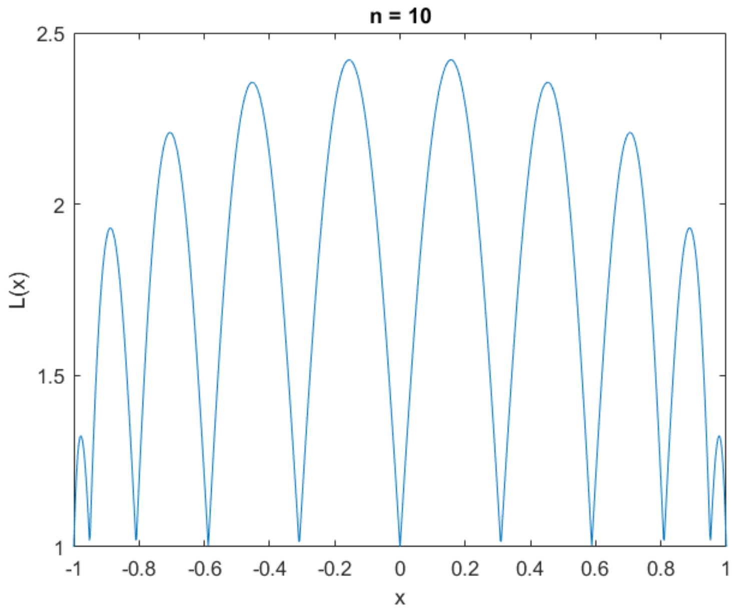
The Lebesgue constant for n=4 is 2.2078
The Lebesgue constant for n=10 is 29.8981
The Lebesgue constant for n=20 is 10979

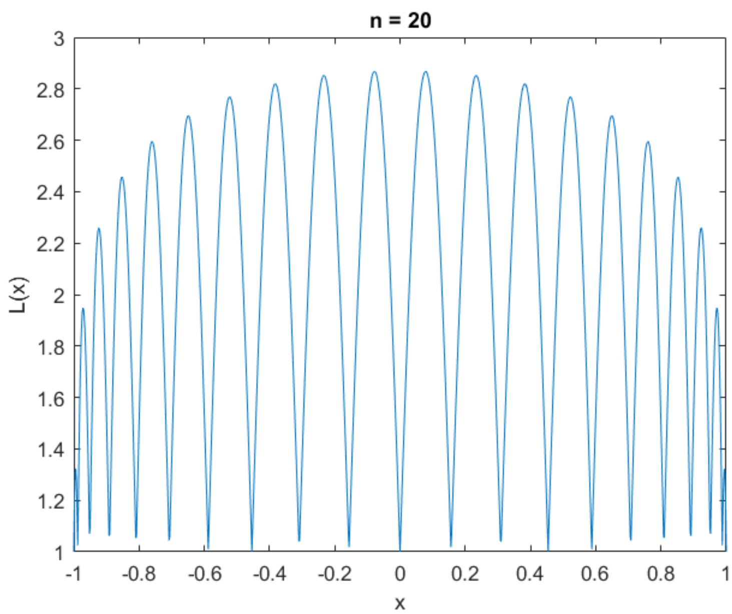
(c)

```
function F = lebesgue_run2(n)
% Computer code for calling lebesque.m and finding the constant
% based on number of evaluation points
% Input:  n     --- number of evaluation points;
% Output: F     --- the Lebesgue constant
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/10/2018

% Create a row vector for storing interpolation points
```

```
x=zeros(1,n+1);

% Find interpolation points and store in x using given formula
for j=1:n+1
    x(j)=cos((j-1)*pi/n);
end

% Adjust parameter for plotting resolution
m=1000;
T=zeros(1,m+1);
xbar=zeros(1,m+1);

% Find points for xbar through iteration with plotting resolution
for k=1:m+1
    xbar(k)=-1+(k-1)*(2/m);
    T(k)=lebesgue(x,xbar(k));
end

% Plot the Lebesgue function
plot(xbar,T);
xlabel('x');
ylabel('L(x)');
title(['n = ' num2str(n)]);

% Find the norm using the data, F will be Lebesgue constant
F=norm(T,Inf);
end
```







The Lebesgue constant for n=4 is 1.7988
The Lebesgue constant for n=10 is 2.4210
The Lebesgue constant for n=20 is 2.8677

The Lebesgue function fluctuates more as n increases with Chebyshev points rather than equidistributed points. The Lebesgue constant also increases faster with equidistributed points than it does with Chebyshev points.

3. (a)

```
function T = barycentric_weights(x)
% Computer code for finding the weights of the Barycentric Formula
% Input:  x --- input points;
% Output: T --- vector of weights to plug into barycentric.m
```

```
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Find length of our input list, create another vector to substitute
% weights
n=length(x);
T=zeros(1,n);

% Iterate through the input list and with interpolation, use the formula
% for calculating the weights
for j=1:n
    w=1;
    for k=1:n
        if j~=k
    w = w*(x(j)-x(k));
    end
end
T(j)=w;
end

% We want the reciprocal of the calculation of w
T=1./T;

end
```

```
function T = barycentric(x,y,w,z)
% Computer code for implmenting Barycentric Formula
% Input:  x --- sample input points
%         y --- sample output points
%         w --- weights calculated through barycentric_weights.m
%         z --- point we are trying to approximate
% Output: T --- approximation for f(x) using Barycentric Formula
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Find length of the lists we are working with, create vector for
% approximating Barycentric
n=length(x);
m=length(z);
T=zeros(1,m);

% Iterate through the lists, finding the interpolation values at each node
for i=1:m
    k=0;
    num=0;
    den=0;
    for j=1:n
        if z(i)==x(j)
            T(i)=y(j);
            k=1;
        else
            % Solve for numerator and denomintor of Barycentric formula
            num=num+(w(j)*y(j))/(z(i)-x(j));
            den=den+(w(j)/(z(i)-x(j)));
        end
    end
    if k==0
        T(i)=num/den;
    end
end
```

(b)

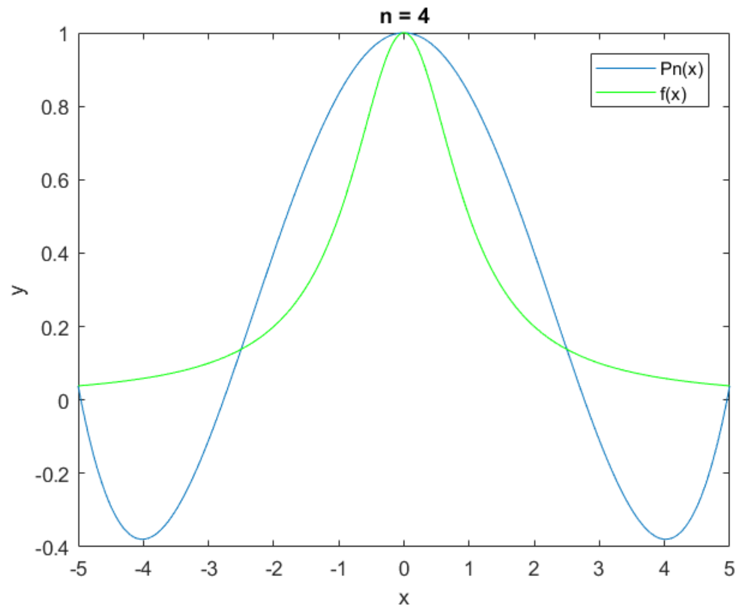
```
function T = barycentric_run(x,y,z)
% Computer code for applying Barycentric Formula
% Input:  x --- sample input points
%         y --- sample output points
%         z --- point we are trying to approximate
% Output: T --- approximation for f(x) using Barycentric Formula
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Calculate weights using barycentric_weights.m
w=barycentric_weights(x);
```

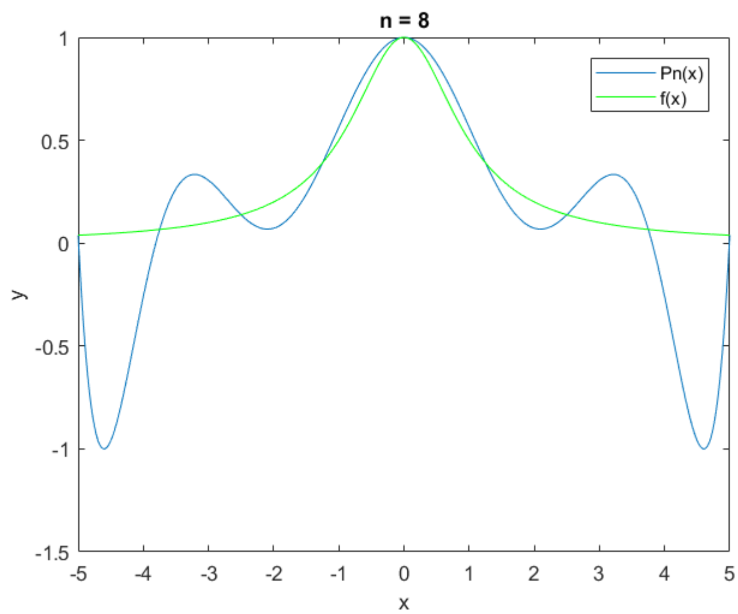
```
% Pass calculated weights along with sample points to barycentric
p=barycentric(x,y,w,z);
disp(p);

end
```

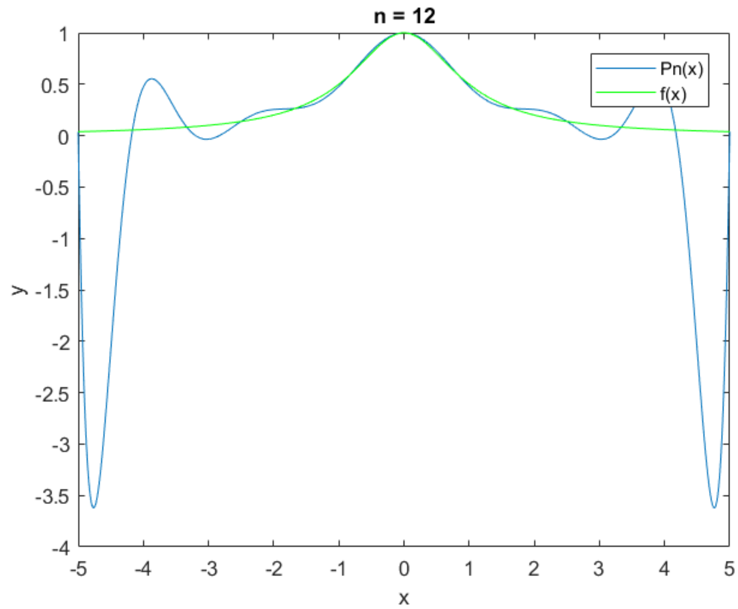
Running the function above with the column x_j stored in a vector called x and column $f(x_j)$ stored in a vector called y, along with z=2 gives us an approximation of $f(2)$=0.8520

4. (a)

```
function T = runge(n)
% Computer code for finding the weights of the Barycentric Formula
% Input:  x --- input points;
% Output: T --- vector of weights to plug into barycentric.m
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Create row vectors based on n or number of nodes
x=zeros(1,n+1);
y=zeros(1,n+1);
w=zeros(1,n+1);

% Iterate through the function, finding the x and y values based on the
% given formula
for j=1:n+1
    x(j)=-5+(j-1)*(10/n);
    y(j)=1/(1+(x(j))^2);
end

% For equidistributed nodes we can use binomial coefficient
for i=1:n+1
    w(i)=((-1)^(i-1))*nchoosek(n,i-1);
end

% We now will create a function with a very high resolution
m=1000;
z=zeros(1,m+1);
F=zeros(1,m+1);

% Iterate through the formula given for larger number m
for k=1:m+1
    z(k)=-5+(k-1)*(10/m);
    F(k)=1/(1+(z(k))^2);
end

% Plot approximation versus actual function to visualize accuracy
T = barycentric(x,y,w,z);
plot(z,T); hold on
plot(z,F,'g');
xlabel('x');
ylabel('y');
legend('Pn(x)','f(x)');
title(['n = ' num2str(n)]);
hold off
    x(j)=-5+(j-1)*(10/n);
    y(j)=1/(1+(x(j))^2);
end
```


n = 4
Pn(x)
f(x)
y
x

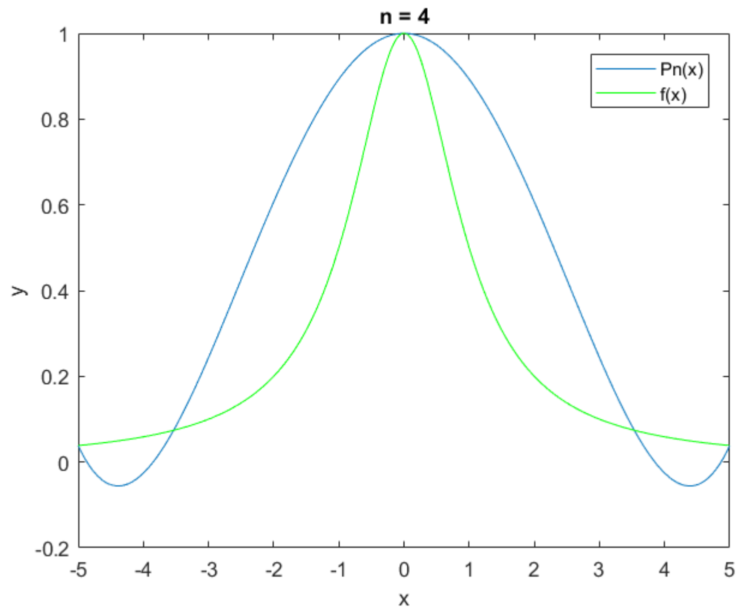

n = 8
Pn(x)
f(x)
y
x

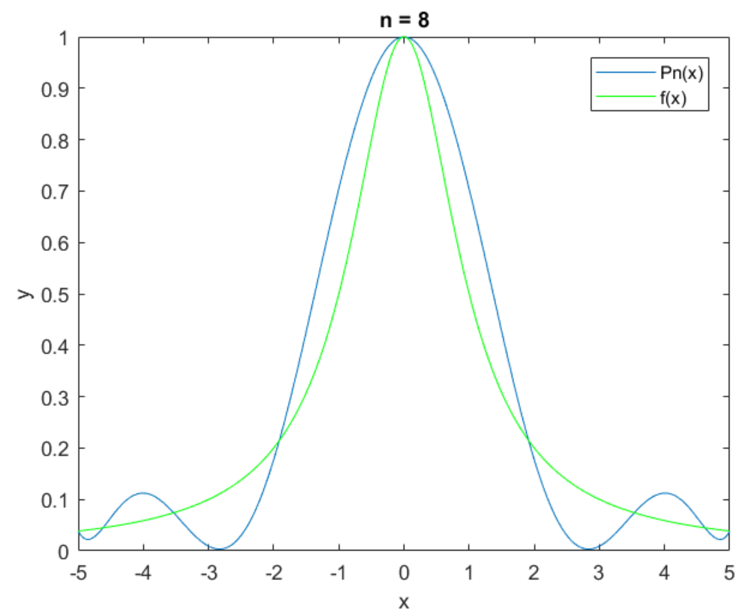


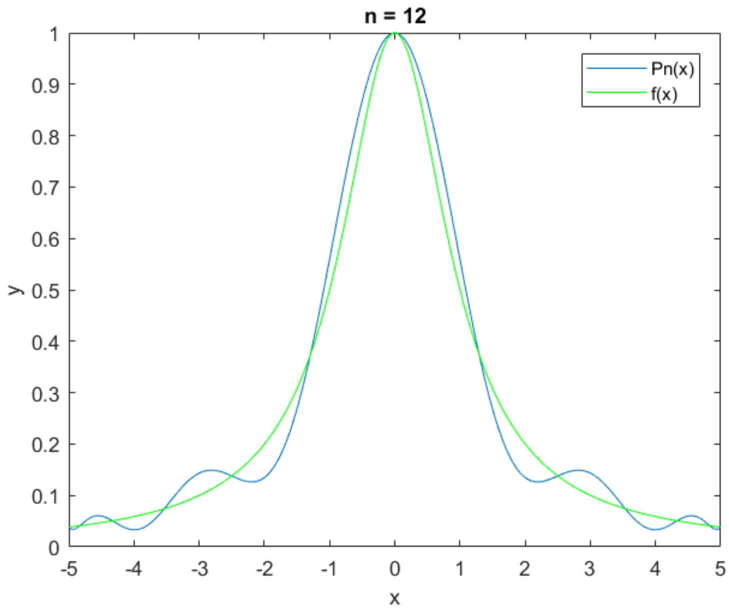
(b)

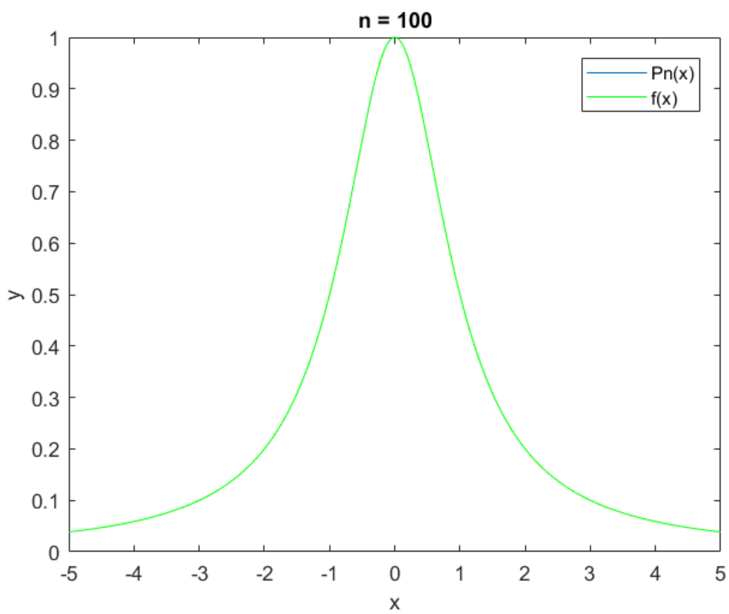
```
function T = runge2(n)
% Computer code for finding the weights of the Barycentric Formula
% using Chebyshev points
% Input:  x --- input points;
% Output: T --- vector of weights to plug into barycentric.m
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Create row vectors based on n or number of nodes
x=zeros(1,n+1);
y=zeros(1,n+1);
w=zeros(1,n+1);

% Iterate through the function, finding the x and y values based on the
% given formula
for j=1:n+1
    x(j)=5*cos((n+1-j)*pi/n);
    y(j)=1/(1+(x(j))^2);
end

w(1)=0.5*((-1)^n);
w(n+1)=0.5;
for i=2:n
    w(i)=(-1)^(n+1-i);
end

% We now will create a function with a very high resolution
m=1000;
z=zeros(1,m+1);
F=zeros(1,m+1);

% Iterate through the formula given for larger number m
for k=1:m+1
    z(k)=5*cos((m+1-k)*pi/m);
    F(k)=1/(1+(z(k))^2);
end

% Plot approximation versus actual function to visualize accuracy
T = barycentric(x,y,w,z);
plot(z,T); hold on
plot(z,F,'g');
xlabel('x');
ylabel('y');
legend('Pn(x)','f(x)');
title(['n = ' num2str(n)]);
```

```
    hold off
end
```


n = 4
Pn(x)
f(x)
y
x
1
0.8
0.6
0.4
0.2
0
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5


n = 8
Pn(x)
f(x)
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5

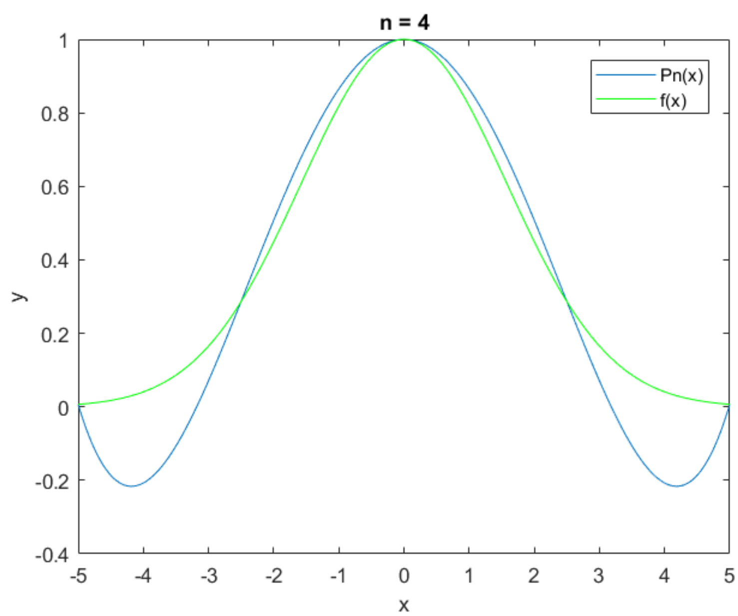


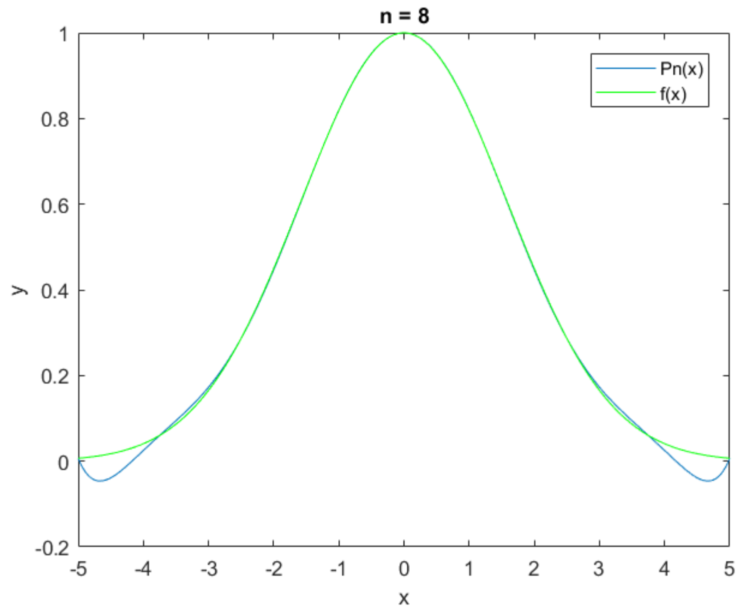


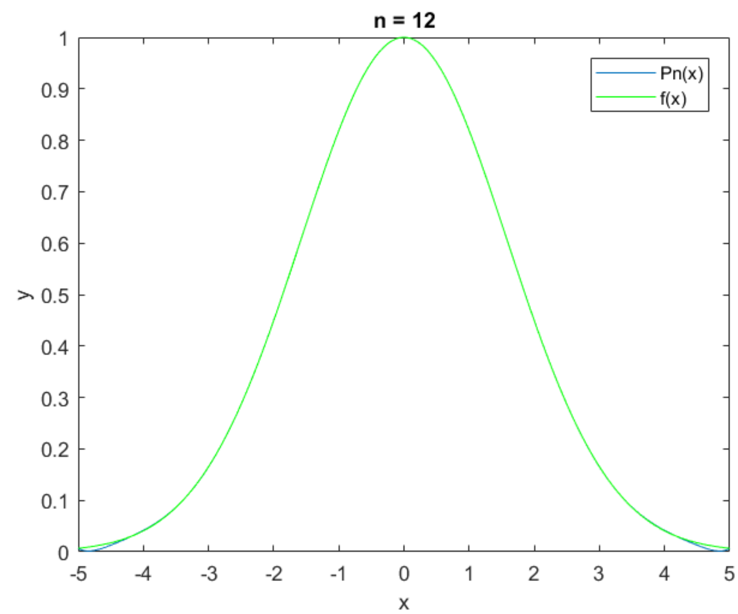
(c)

```
function T = runge3(n)
% Computer code for finding the weights of the Barycentric Formula
% using equidistributed nodes. We are approximating a different function
% than from part a
% Input:  x --- input points;
% Output: T --- vector of weights to plug into barycentric.m
%
% Author: Raghav Thirumulu, Perm 3499720
% Date:   07/11/2018

% Create row vectors based on n or number of nodes
x=zeros(1,n+1);
y=zeros(1,n+1);
w=zeros(1,n+1);

% Iterate through the function, finding the x and y values based on the
```

```
% given formula
for j=1:n+1
    x(j)=-5+(j-1)*(10/n);
    y(j)=exp(-(x(j)^2)/5);
end

% For equidistributed nodes we can use binomial coefficient
for i=1:n+1
    w(i)=((-1)^(i-1))*nchoosek(n,i-1);
end

% We now will create a function with a very high resolution
m=1000;
z=zeros(1,m+1);
F=zeros(1,m+1);

% Iterate through the formula given for larger number m
for k=1:m+1
    z(k)=-5+(k-1)*(10/m);
    F(k)=exp(-(z(k)^2)/5);
end

% Plot approximation versus actual function to visualize accuracy
T = barycentric(x,y,w,z);
plot(z,T); hold on
plot(z,F,'g');
xlabel('x');
ylabel('y');
legend('Pn(x)','f(x)');
title(['n = ' num2str(n)]);
hold off
    x(j)=-5+(j-1)*(10/n);
    y(j)=1/(1+(x(j))^2);
end
```







As we increase n, the approximations using the Barycentric Formula converge to the exact function. Approximations using Chebshev points tend to be more accurate.