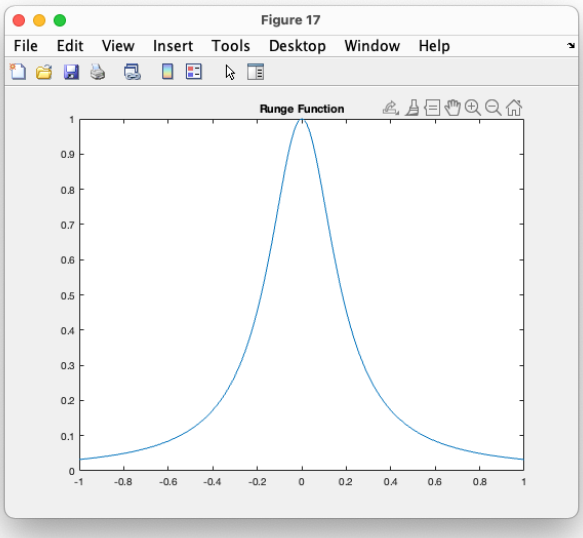
1.

On the right is the plot for the Runge function on [-1,1] in Matlab for a = 30.



2.

a. Created the Vandermonde matrix using the vander() function, and solved it using \.

b. Used Professor provided codes to help compute the barycentric form of the interpolation polynomial.

c. One of the harder interpolations to implement. Used the textbook formula to find the coefficients which is $\frac{F_{i,j-1}-F_{i-1,j-1}}{X_i-X_{i-j}}$ and used it in p(x) = $F_{0,0} + \sum_{i=1}^{n} F_{i,i} \prod_{j=0}^{i-1}(x - x_j)$ to calculate the Newton Method.

d. I tested the speed with n=2500. The fastest was newton at 1.411 seconds. Second fastest was the barycentric form at 1.498s, and the slowest was the vandermonde form at 1.640 seconds.

3. All plots are n=10, n=15, n=25, n=40

a. For equidistant points, the larger the n values, the bigger the oscillation gets at the end. This makes the sides less accurate, as we also noticed this in class.

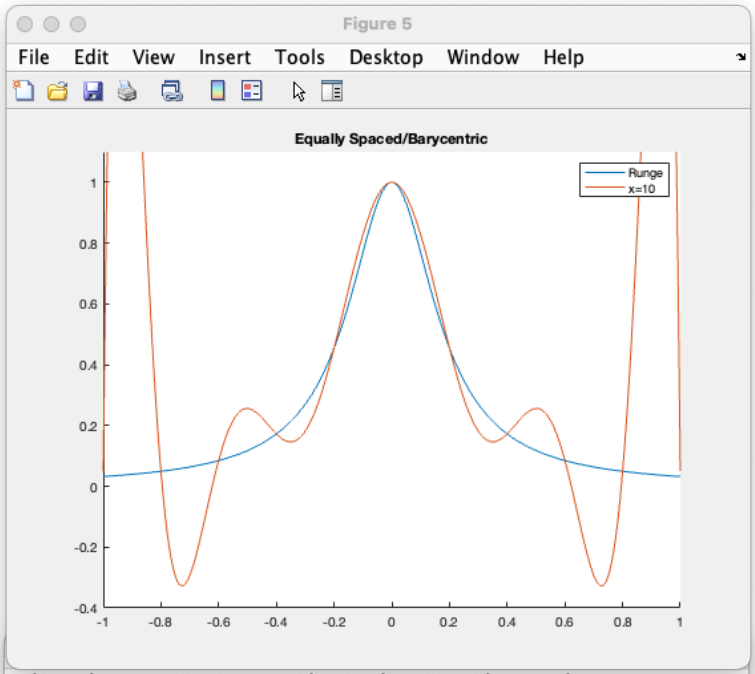
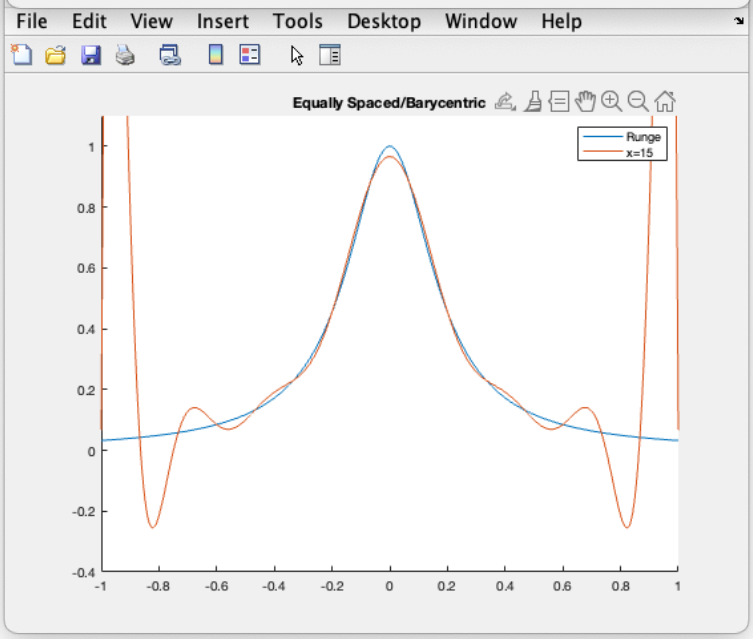


Figure 5
File Edit View Insert Tools Desktop Window Help
Equally Spaced/Barycentric
Runge
x=10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1


File Edit View Insert Tools Desktop Window Help
Equally Spaced/Barycentric
Runge
x=15
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

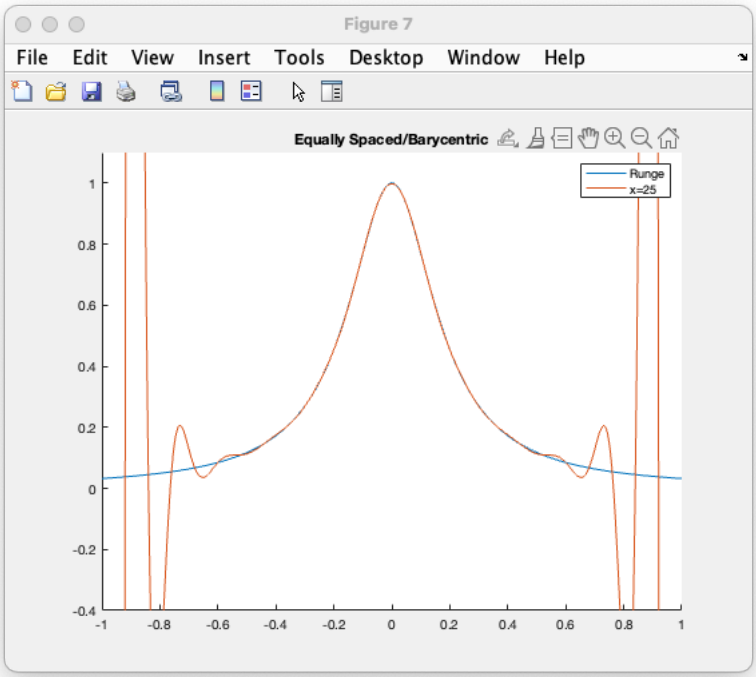
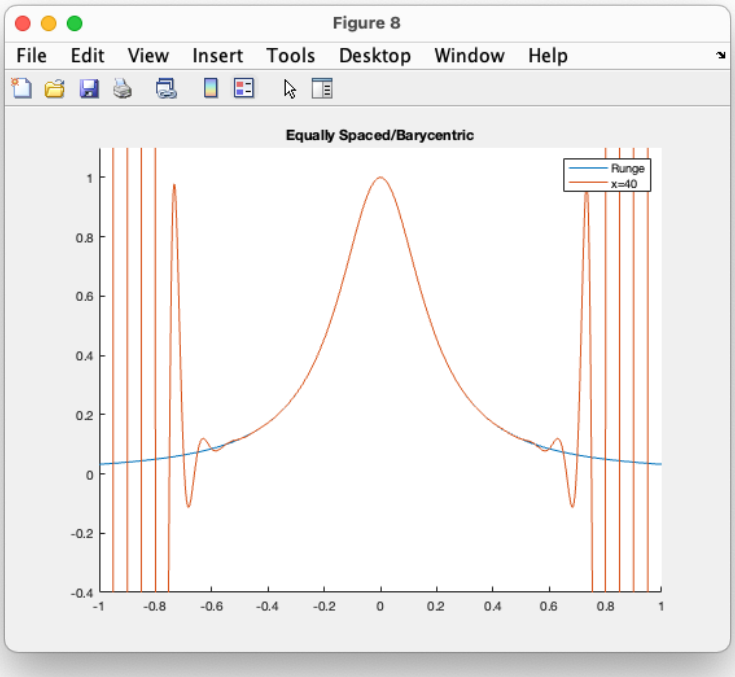
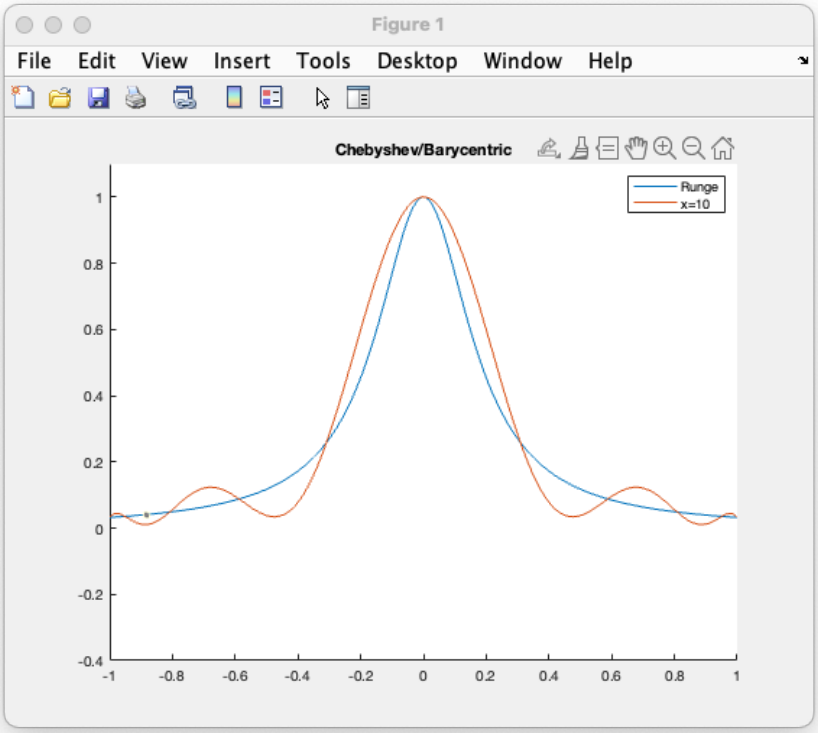


Figure 7
File Edit View Insert Tools Desktop Window Help
Equally Spaced/Barycentric
Runge
x=25
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1



b. When using Chebyshev points, the sides were significantly more accurate than equidistant points. Then increasing the n to a reasonable amount, we were able to get the graph to be extremely close to the Runge function, including around the ends of the interval. This is much different than equidistant points.



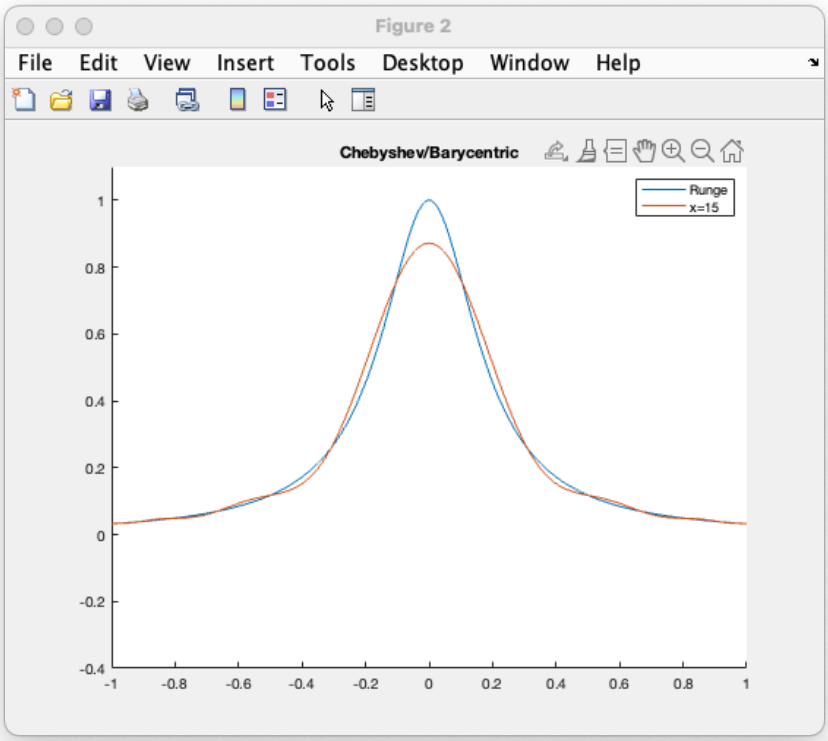


Figure 2
File Edit View Insert Tools Desktop Window Help
Chebyshev/Barycentric
Runge
x=15
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

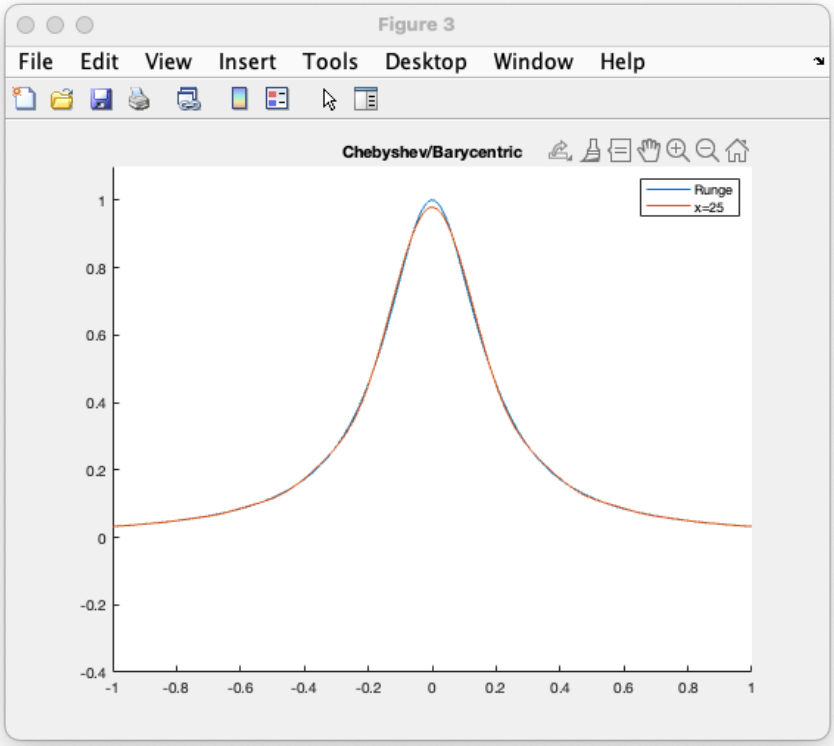
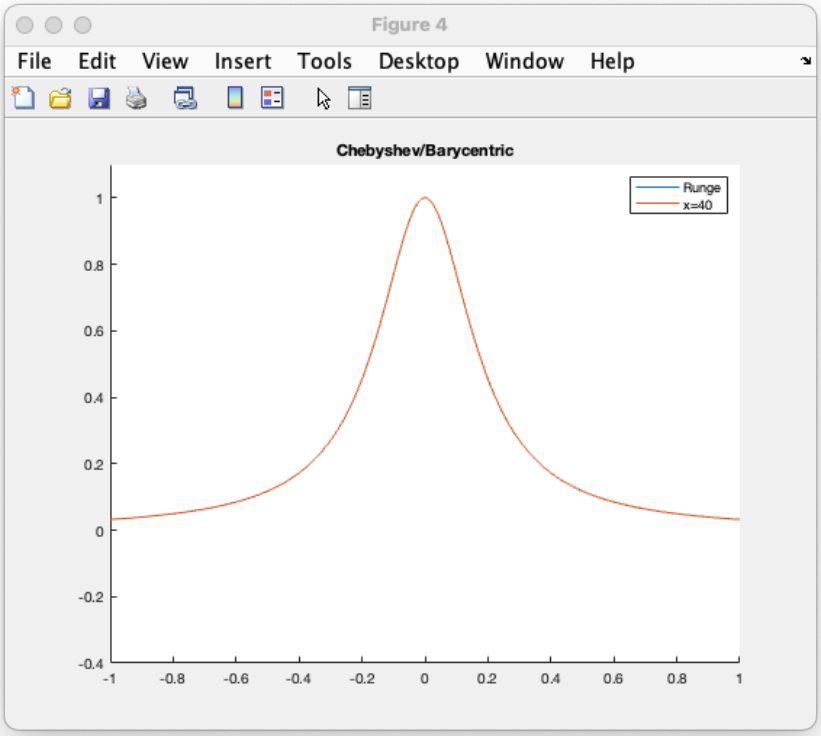
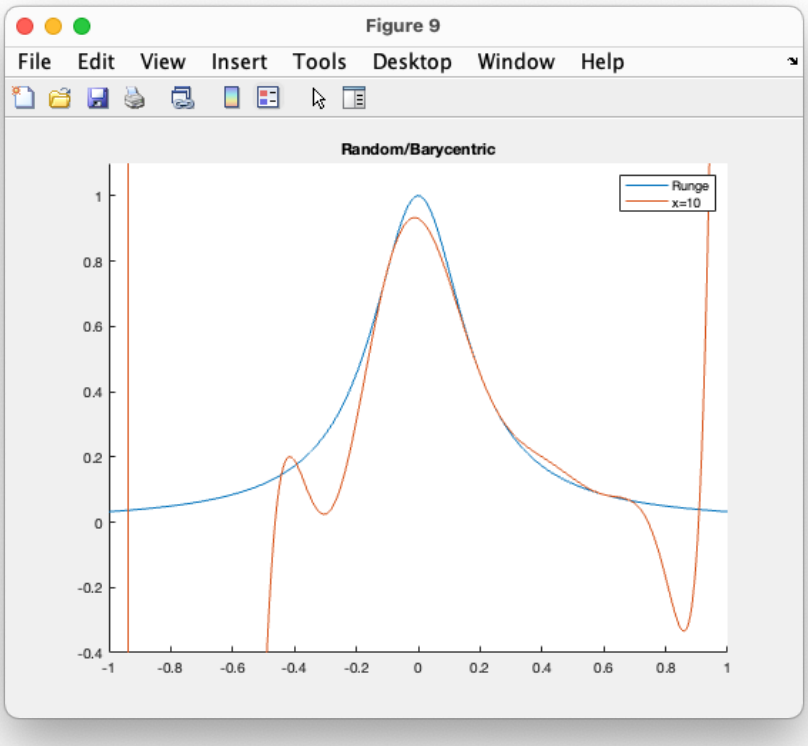


Figure 3
File Edit View Insert Tools Desktop Window Help
Chebyshev/Barycentric
Runge
x=25
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1



c. I tried randomized points on the interval [-1,1], and they were a disaster. Inconsistent and often not close to the Runge function. Overall, I think Chebyshev chooses the best points.



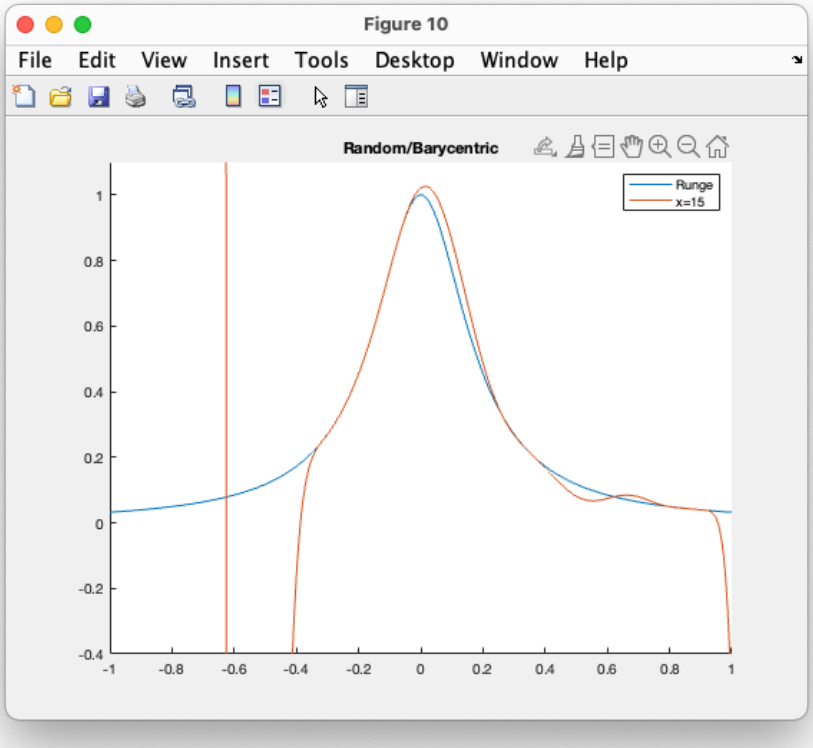


Figure 10
File Edit View Insert Tools Desktop Window Help
Random/Barycentric
Runge
x=15
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

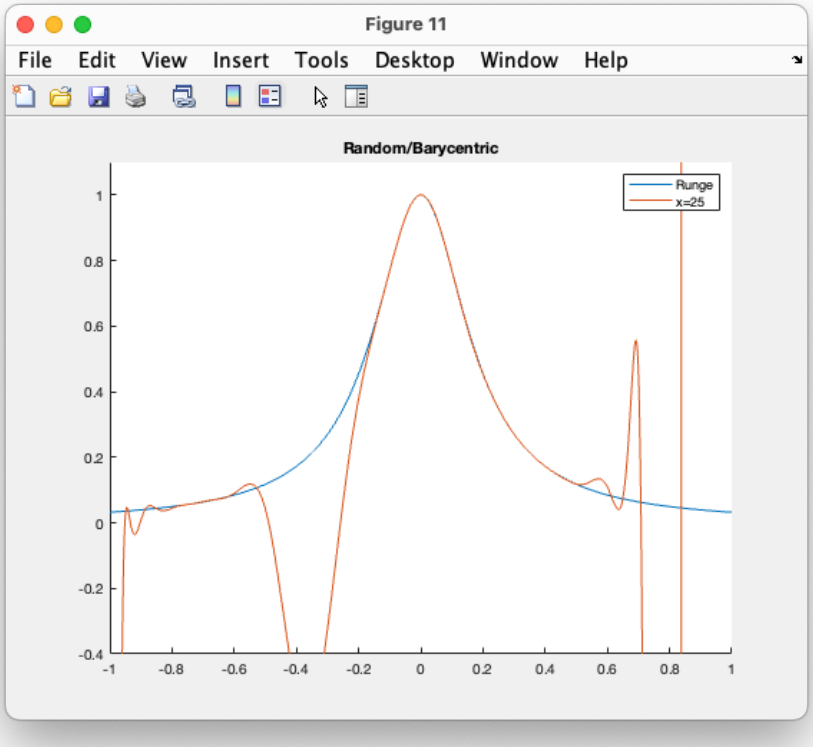


Figure 11
File Edit View Insert Tools Desktop Window Help
Random/Barycentric
Runge
x=25
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

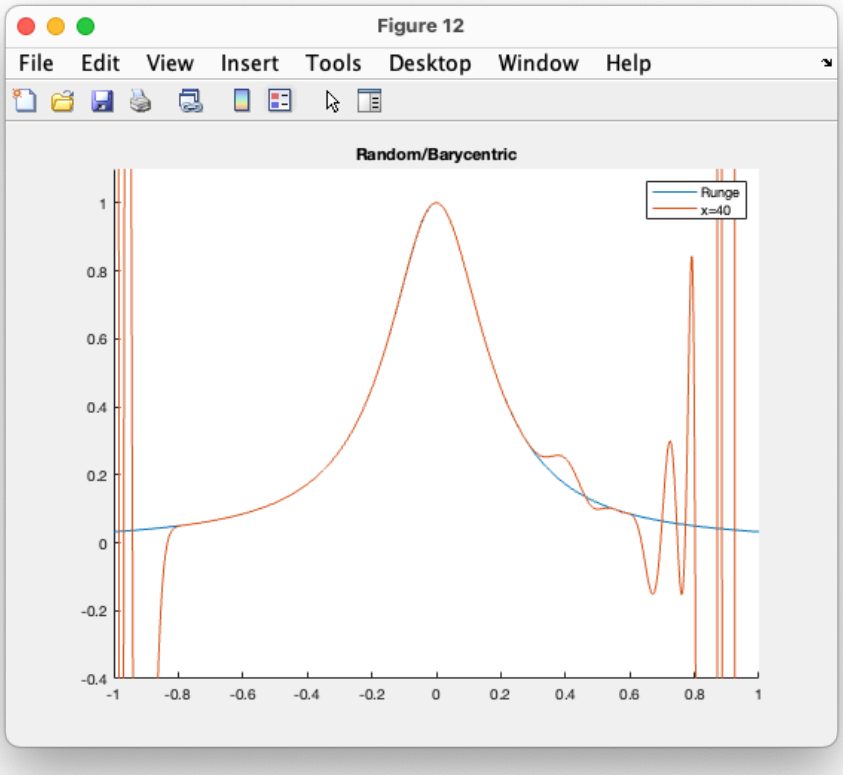
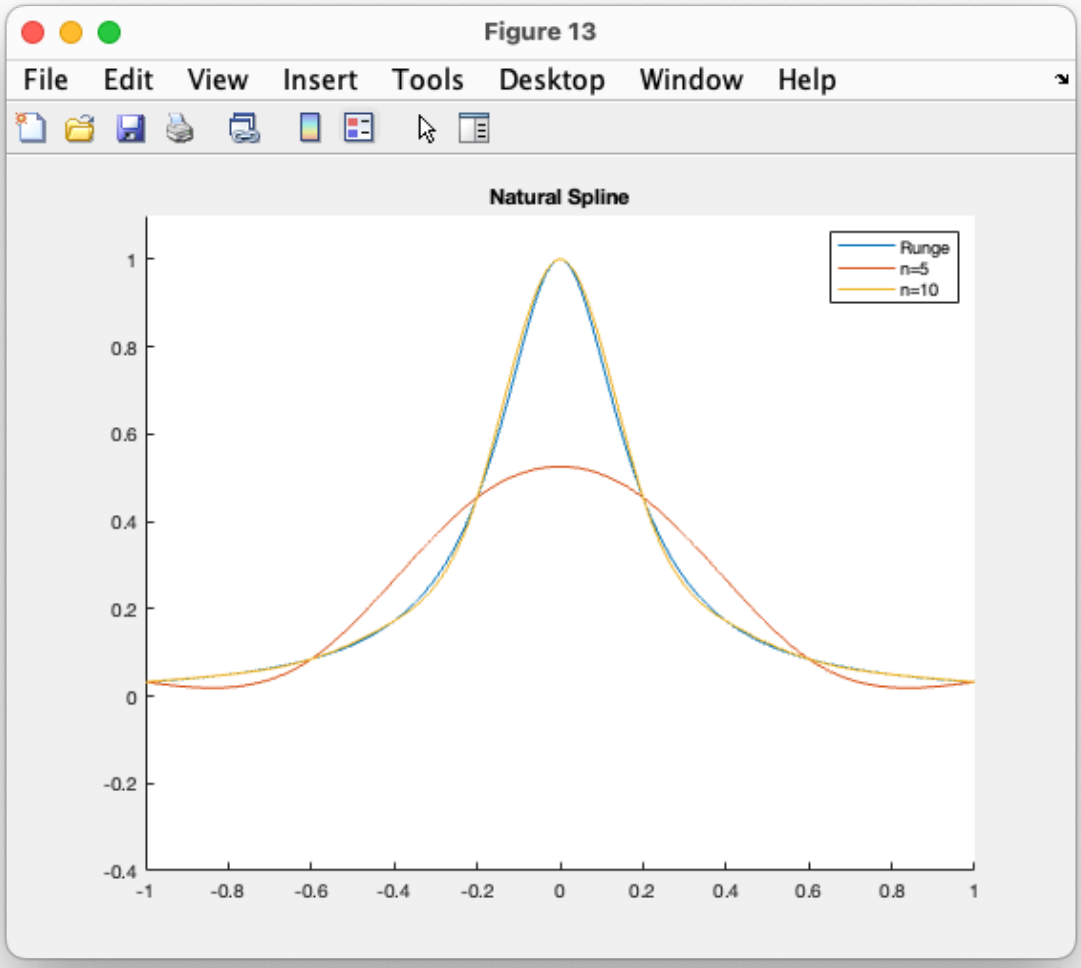


Figure 12
File Edit View Insert Tools Desktop Window Help
Random/Barycentric
Runge
x=40
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

4. Below is the plot for the natural spline. Created using csape() and fnval(). When I increased the n values, I noticed the plot getting more accurate, much like when using Chebyshev points.



```
%% Assignment 5

%for plot
m = 9999;
x = linspace(-1,1,m);

%function
y = 1./(1+30.*x.^2);

% %barycentric interpolation with equally spaced nodes
n = 10+1;
ips = linspace(-1,1,n)';
f = 1./(1+30.*ips.^2);
Eq1 = barycentric_form(ips,f,x);

n = 15+1;
ips = linspace(-1,1,n)';
f = 1./(1+30.*ips.^2);
Eq2 = barycentric_form(ips,f,x);

n = 25+1;
ips = linspace(-1,1,n)';
f = 1./(1+30.*ips.^2);
Eq3 = barycentric_form(ips,f,x);

n = 40+1;
ips = linspace(-1,1,n)';
f = 1./(1+30.*ips.^2);
Eq4 = barycentric_form(ips,f,x);

%barycentric interpolation with Chebyshev nodes
n = 10+1 ;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Cheby1 = barycentric_form(ips,f,x);

n = 15+1 ;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Cheby2 = barycentric_form(ips,f,x);

n = 25+1 ;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Cheby3 = barycentric_form(ips,f,x);

n = 40+1 ;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Cheby4 = barycentric_form(ips,f,x);

%barycentric interpolation with Random nodes

n = 10+1;
ips = -1 + (1+1)*rand(n,1);
f = 1./(1+30.*ips.^2);
bary_random1 = barycentric_form(ips,f,x);
```

```
n = 15+1;
ips = -1 + (1+1)*rand(n,1);
f = 1./(1+30.*ips.^2);
bary_random2 = barycentric_form(ips,f,x);

n = 25+1;
ips = -1 + (1+1)*rand(n,1);
f = 1./(1+30.*ips.^2);
bary_random3 = barycentric_form(ips,f,x);

n = 40+1;
ips = -1 + (1+1)*rand(n,1);
f = 1./(1+30.*ips.^2);
bary_random4 = barycentric_form(ips,f,x);

% Spline

n = 5+1;
ips = linspace(-1,1,n);
f = 1./(1+30.*ips.^2);
spline1 = natural_spline(ips,f,x);

n = 10+1;
ips = linspace(-1,1,n);
f = 1./(1+30.*ips.^2);
spline2 = natural_spline(ips,f,x);

% Timed Interpolation Tests
% Uncomment Code to make it work

%vandermonde interpolation with Chebyshev nodes
n = 20+1;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Vand_test = vandermont_form(ips,f,x);

%barycentric interpolation with Chebyshev nodes
n = 20+1;
angles = linspace(0,pi,n)';
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Bary_test = barycentric_form(ips,f,x);

%newton interpolation with Chebyshev nodes
n = 20+1;
angles = linspace(0,pi,n);
ips = cos(angles);
f = 1./(1+30.*ips.^2);
Newton_test = newton_form(ips,f,x);

% ------------------------ PLOTS --------------------------------

% Plot Chebyshev Nodes/Barycentric

figure(1);
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Cheby1,'LineWidth',0.8)
ylim([-0.4 1.1])
title('Chebyshev/Barycentric')
legend('Runge','x=10')
```

```
figure(2)
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Cheby2,'LineWidth',0.8)
ylim([-0.4 1.1])
title('Chebyshev/Barycentric')
legend('Runge','x=15')

figure(3);
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Cheby3,'LineWidth',0.8)
ylim([-0.4 1.1])
title('Chebyshev/Barycentric')
legend('Runge','x=25')

figure(4);
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Cheby4,'LineWidth',0.8)
ylim([-0.4 1.1])
title('Chebyshev/Barycentric')
legend('Runge','x=40')
%legend('Runge','x=10','x=15','x=25','x=40')

% Plot Equidistant Nodes/Barycentric

figure(5);
hold on;
plot(x,y)
plot(x,Eq1)
title('Equally Spaced/Barycentric')
legend('Runge','x=10')
ylim([-0.4 1.1])

figure(6)
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Eq2)
title('Equally Spaced/Barycentric')
legend('Runge','x=15')
ylim([-0.4 1.1])

figure(7)
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Eq3)
title('Equally Spaced/Barycentric')
legend('Runge','x=25')
ylim([-0.4 1.1])

figure(8)
hold on;
plot(x,y,'LineWidth',0.8)
plot(x,Eq4)
ylim([-0.4 1.1])
title('Equally Spaced/Barycentric')
legend('Runge','x=40')
ylim([-0.4 1.1])
%legend('Runge','x=10','x=15','x=25','x=40')

% Plot Randomized Nodes/Barycentric
```

```
figure(9)
hold on;
plot(x,y);
plot(x,bary_random1);
legend('Runge','x=10')
ylim([-0.4 1.1])
title('Random/Barycentric')

figure(10)
hold on;
plot(x,y)
plot(x,bary_random2);
legend('Runge','x=15')
ylim([-0.4 1.1])
title('Random/Barycentric')

figure(11)
hold on;
plot(x,y)
plot(x,bary_random3);
legend('Runge','x=25')
ylim([-0.4 1.1])
title('Random/Barycentric')

figure(12)
hold on;
plot(x,y)
plot(x,bary_random4);
ylim([-0.6 1.2])
title('Random/Barycentric')
legend('Runge','x=40')
ylim([-0.4 1.1])
%legend('Runge','x=10','x=15','x=25','x=40')

% Plot Natural Spline

figure(13);
hold on;
plot(x,y);
plot(x,spline1);
plot(x,spline2);
ylim([-0.4 1.1])
title('Natural Spline');
legend('Runge','n=5','n=10');

% Plot timed tests

figure(14)
plot(x,Vand_test)
title('Vand Test')
figure(15)
plot(x,Bary_test)
title('Bary Test')
figure(16)
plot(x,Newton_test)
title('Newton Test')

% Plot Runge

figure(17)
plot(x,y)
title('Runge Function')
```

```
%% Natural Spline

function p = natural_spline(ips,f,x)

p = csape(ips,f,'second');
p = fnval(p,x);

end

%% Newton

function p = newton_form(ips,f,x)

n = (size(f,2)-1);
F(:, 1) = f;
for i=1:n
    for j=1:i
        F(i+1,j+1) = (F(i+1,j)-F(i,j)) / (ips(i+1)-ips(i-j+1));
    end
end
a = diag(F);
p = a(n+1);
for i=1:n
    p = a(n+1-i) + p.*(x-ips(n+1-i));
end
end

%% Vandermont Form

function p = vandermont_form(ips,f,x)

A = fliplr(vander(ips));
result = A\f;
result = flip(result);
result = transpose(result);
p=polyval(result,x);

end

%% Barycentric Form
function p = barycentric_form(ips,f,x)

n = size(ips,1);
m = size(x,1);
w = ones(n,1);
for i = 1:n
    for k = 1:n
        if i ~= k
            w(i) = w(i)*(ips(i)-ips(k));
        end
    end
    w(i) = 1/w(i);
end
numer = zeros(m,1);
denom = zeros(m,1);
for i = 1:n
    numer = numer + f(i)*w(i)./(x-ips(i));
    denom = denom + w(i)./(x-ips(i));
end
p = numer./denom;
```

```
end
```