```
clc
close all
clear all
% Q2b test case
% Example from https://www.mathworks.com/help/matlab/ref/polyval.html
p = [3 2 1];
x = [5 7 9];
y = polyval(p,x)';
a = interp_monomials(x,y);

% Q2c
sym x;
f = @(x) 1 / (1 + 30 * (x^2));
x_vals = {3};
y_vals = {3};
n_s = [3 6 12 43];

for i = 1:length(n_s)
    n = n_s(i);
    x_s = zeros(1,n);
    y_s = zeros(1,n);

    for j = 0:length(x_s)
        % j here is our iterator
        x_i = -1 + j * (2/n);
        x_s(j+1) = x_i;
        y_s(j+1) = f(x_i);
    end

    x_vals{i} = x_s;
    y_vals{i} = y_s;

end

% Plot the original function f(x) on 1000 points
full_vals = linspace(-1,1,1000);

for i = 1:1000
    full_vals_evaluated(i) = f(full_vals(i));
end

figure
plot(full_vals, full_vals_evaluated, 'LineWidth', 2);
hold on

evals = {};

for i = 1:length(n_s)
   a = interp_monomials(x_vals{i}, y_vals{i}');
   a_s{i} = a;
   evals{i} = polyval(a, full_vals);
   if i < 4
```

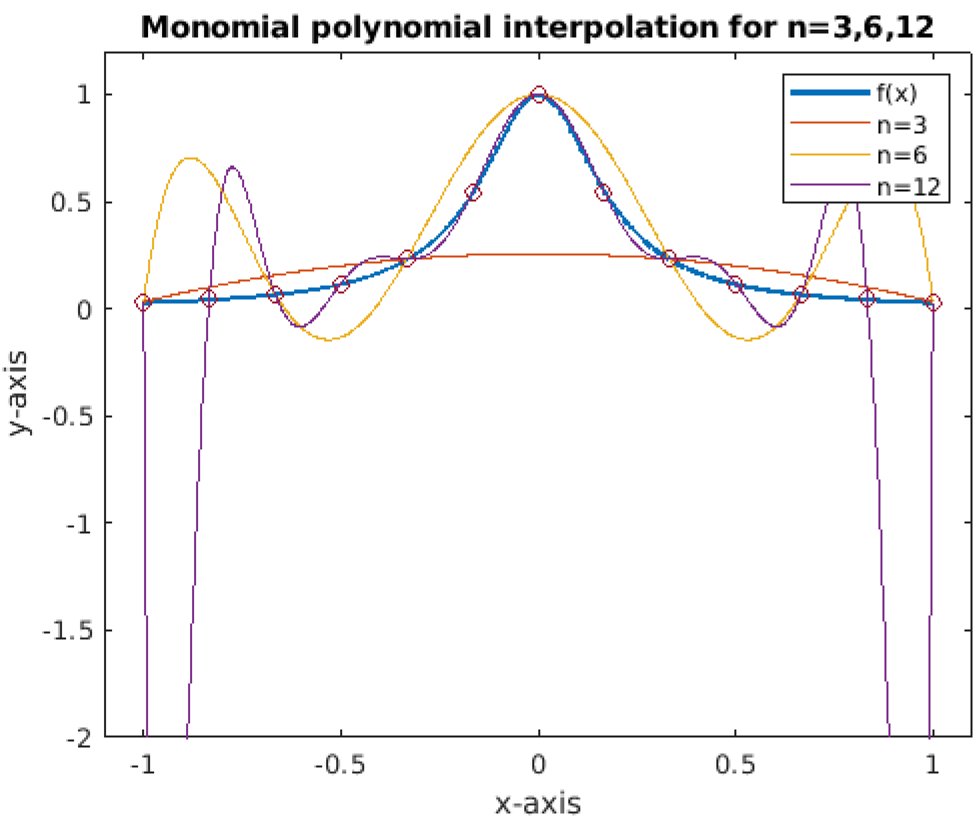
```
        plot(full_vals, evals{i});
    end
end

% Plotting interpolatory points
plot(x_vals{1}, y_vals{1}, 'o');
plot(x_vals{2}, y_vals{2}, 'o');
plot(x_vals{3}, y_vals{3}, 'o');

xlim([-1.1 1.1]);
ylim([-2 1.2]);
title("Monomial polynomial interpolation for n=3,6,12");
xlabel("x-axis");
ylabel("y-axis");
legend("f(x)","n=3","n=6","n=12");

% Q2d
% Getting error for our 4th n, i.e. the ill-conditioned case
polyEval = polyval(fliplr(a_s{4}'), x_vals{4});
error = norm((polyEval - y_vals{4}) / norm(y_vals{4}));
fprintf("Error for n=43: %1.8f", error);

hold off

figure
plot(full_vals, full_vals_evaluated);
hold on
plot(full_vals, evals{4});
title(["Ill-conditioned case: n = " num2str(n)]);
xlabel("x-axis");
ylabel("y-axis");


% Q4 - Interpolate with Chebyshev interpolation
cheby_x_vals = {};
cheby_y_vals = {};
cheby_n_s = [3 6 12 20 43];

for i = 1:length(cheby_n_s)
    x_s = zeros(1,cheby_n_s(i));
    y_s = zeros(1,cheby_n_s(i));

    for j = 0:cheby_n_s(i)
        % j here is our iterator
        x_i = cos(((2 * j + 1) * pi)/ ((2 * cheby_n_s(i)) + 2));
        x_s(j+1) = x_i;
        y_s(j+1) = f(x_i);
    end

    cheby_x_vals{i} = x_s;
    cheby_y_vals{i} = y_s;

end
```

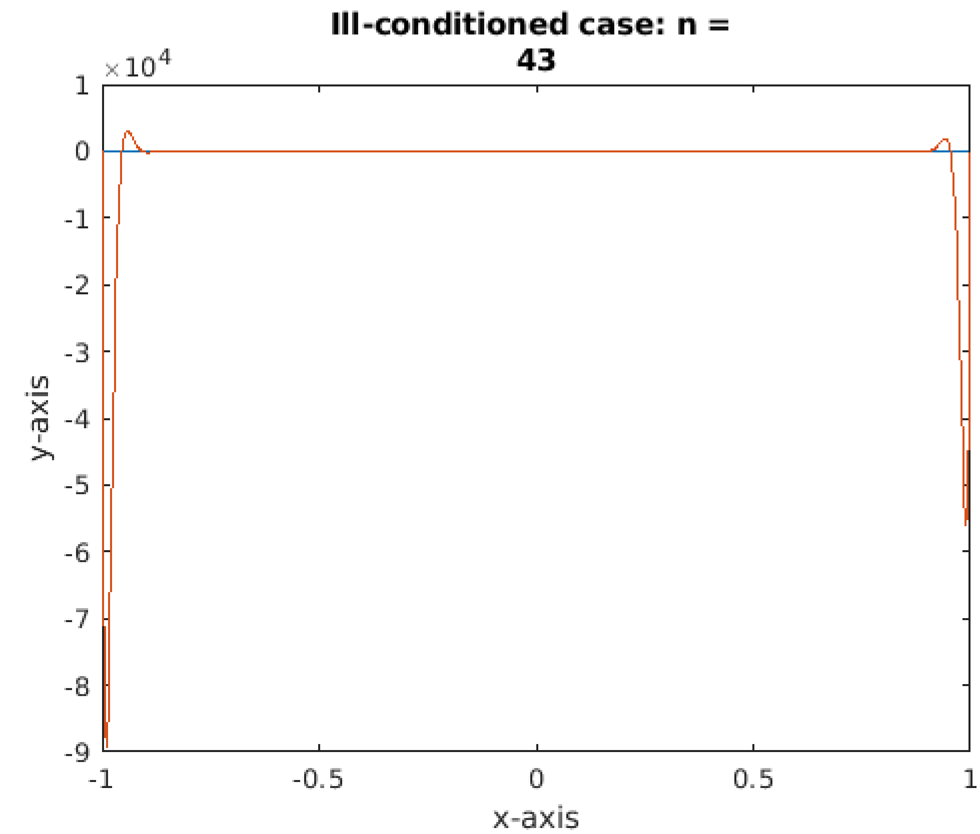
```
% Plots for Q4
figure
plot(full_vals, full_vals_evaluated, 'LineWidth', 2);

hold on
cheby_evals = {};

for i = 1:length(cheby_n_s)
   cheby_evals{i} = lagrange(full_vals, cheby_x_vals{i},
 cheby_y_vals{i});
   plot(full_vals, cheby_evals{i});
end

title(["Chebyshev points with Lagrange interpolation",...
    "including ill-conditioned case."]);
xlabel("x-axis");
ylabel("y-axis");
legend("n=3","n=6","n=12","n=20","n=43");
hold off

% Q5
cheby_x_vals_q5 = {};
cheby_y1_vals = {};
cheby_y2_vals = {};

n_q5 = 1:16;

f1_maxs = zeros(1,16);
f2_maxs = zeros(1,16);
exact_y1 = cos(full_vals);
exact_y2 = abs(full_vals);

for i = 1:length(n_q5)

    x_s = zeros(1,n_q5(i));
    y_1s = zeros(1,n_q5(i));
    y_2s = zeros(1,n_q5(i));

    for j = 0:(n_q5(i))
       x_i = cos(((2 * j + 1) * pi)/ ((2 * n_q5(i)) + 2));

       x_s(j+1) = x_i;
       y_1s(j+1) = cos(x_i);
       y_2s(j+1) = abs(x_i);

    end

    cheby_x_vals_q5{i} = x_s;
    cheby_y1_vals{i} = y_1s;
    cheby_y2_vals{i} = y_2s;

    lagrange_eval1 = lagrange(full_vals, cheby_x_vals_q5{i},
 cheby_y1_vals{i});
```

```
    lagrange_eval2 = lagrange(full_vals, cheby_x_vals_q5{i},
 cheby_y2_vals{i});

    f1_maxs(i) = max(abs(exact_y1 - lagrange_eval1));
    f2_maxs(i) = max(abs(exact_y2 - lagrange_eval2));

end

% Plot for Q5
figure
semilogy(1:16, f1_maxs);
hold on
semilogy(1:16, f2_maxs);
hold off
title(["Error for Chebyshev & Lagrange interpolation",...
    "for f(x)=cos(x) and f(x)=abs(x)"]);
xlabel("x-axis");
ylabel("y-axis (log)");
legend(["f(x)=cos(x)","f(x)=abs(x)"],'Location',"southwest");


% Q2b function
function[a] = interp_monomials(x,y)
    n = length(x);
    V = zeros(n);

    for i = 1:n
        x_val = x(i);
        for j = 1:n
            % Taking powers of x value to fill in row.
            V(i,j) = x_val ^ (j-1);
        end
    end

    % Our coefficients are column-wise and upside down.
    % The below fixes that and makes a row vector.
    a = fliplr((V\y)');
    % disp(cond(V))
end


% https://www.mathworks.com/matlabcentral/fileexchange/899-lagrange-
polynomial-interpolation
% Comments given to explain each step. Inspiration from Lecture 14,
 slide 4
function y0 = lagrange(x0, x, y)

n = length(x);
L = ones(n, length(x0));

% Calculates the coefficients.
% This loop corresponds to the product loop l_k
for i = 1:n
    for j = 1:n
```

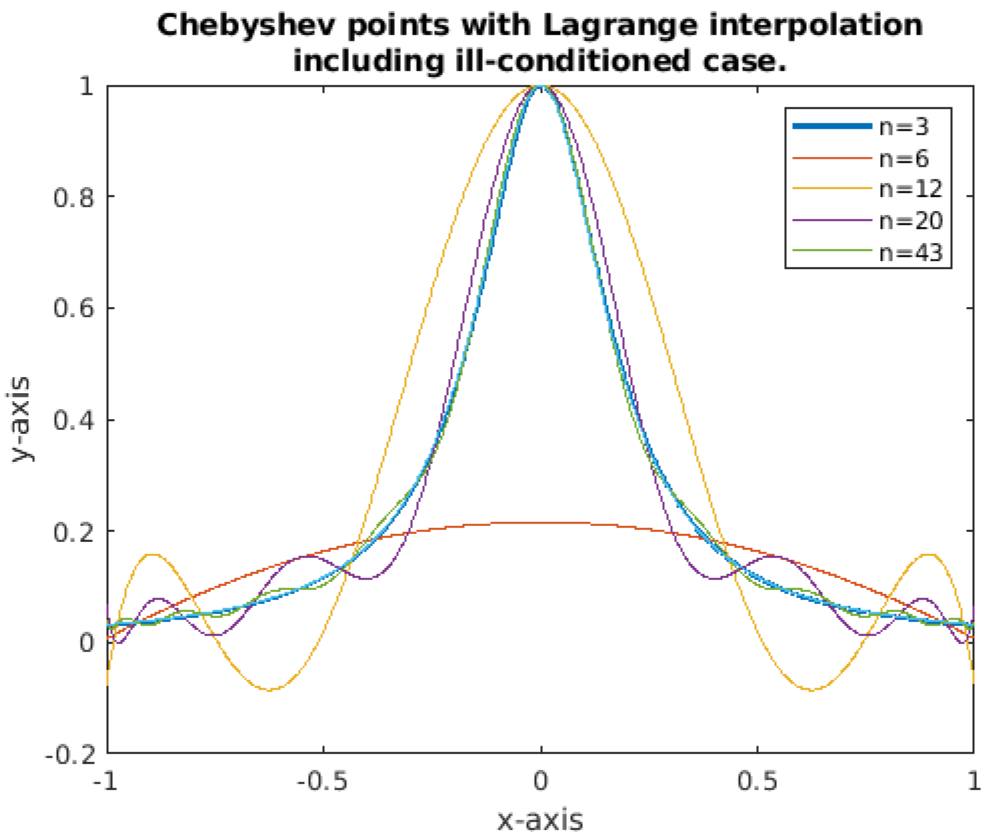
```
        % If i equals j then don't put that difference
        % into the lagrange form.
        if (i ~= j)
            % Calculate the lagrange form. l_k
            L(i,:) = L(i,:).*(x0-x(j))/(x(i)-x(j));
        end
    end
end

% Multiply our coefficients from the Lagrange form by our
% evaluated x's. In this case it is our Chebyshev points.
% This returns our 1000 points interpolated by lagrange
% with x,y inputs.
% The below loop corresponds to p(x) = sum(l_k * y_k)
y0= 0;
for i = 1:n
    y0 = y0+y(i)*L(i,:);
end

end
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 5.557481e-20.
Error for n=43: 0.00221929

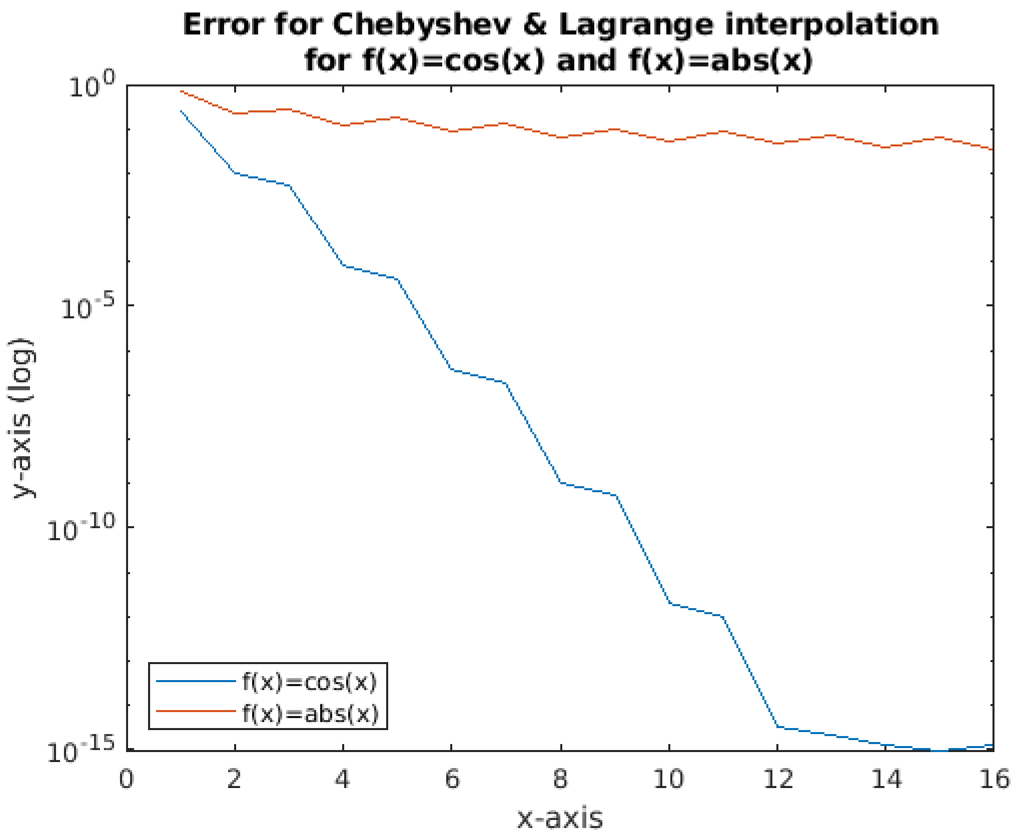




Ill-conditioned case: n =
43
$\times10^4$
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
y-axis
-1
-0.5
0
0.5
1
x-axis



Chebyshev points with Lagrange interpolation
including ill-conditioned case.
1
0.8
0.6
0.4
0.2
0
-0.2
y-axis
n=3
n=6
n=12
n=20
n=43
-1
-0.5
0
0.5
1
x-axis



Published with MATLAB® R2018a