AE 370 — Recitation on polynomial interpolation

1. Here's a remarkable example of when polynomial interpolation can go very well. *Buyer beware*: this example is *not* representative of polynomial interpolation for general functions! You will learn more about that in Homework 2.

   Let's consider interpolating the function $f(x) = \sin(x)$ over the interval $-5 \leq x \leq 5$ using polynomials of degree n; *i.e.* we want to represent $f(x)$ in the space $\mathcal{P}_n$. What decisions do we need to make in order to perform this interpolation? Well, we need to pick

   - A basis for $\mathcal{P}_n$
   - A set of points to interpolate
   - The degree n of our polynomial interpolant

   We know from class that good choices for the first two bullet points are to use a Lagrange basis and to use Chebyshev points over the interval $[-5, 5]$. (*Aside:* The Chebyshev points we expressed in class, $x_j = \cos(j\pi/n)$, are defined over the interval $[-1, 1]$. How can you shift and scale these points to the correct interval?)

   Let's be wise and use Lagrange basis functions, but be very naive in our choice of points and pick uniformly distributed points over the interval $[-3, 3]$. So we're only requiring that our interpolant match $f(x)$ over a small subdomain of the entire domain over which $f(x)$ is defined! Surely, this approach is doomed to fail. Or is it? Let's see!

   We want $n+1$ uniformly spaced points over $[-3, 3]$, so we write our x_j as $x_j = -3 + 6j/n$ for $j = 0, n$.

   And we express our polynomial interpolant as

   $$p_n(x) = \sum_{i=0}^{n} d_i L_i(x) \tag{1}$$

   where the $L_i(x)$ are the Lagrange basis functions defined as

   $$L_i(x) = \frac{\Pi_{k=0(k\neq i)}^{n}(x - x_k)}{\Pi_{k=0(k\neq i)}^{n}(x_i - x_k)} \tag{2}$$

   and the d_i are coefficients that are yet to be determined.

   To solve for our d_i, we require that our interpolant $p_n(x)$ defined in (1) be equal to $f(x)$ at the x_j. That is,

   $$p_n(x_j) = f(x_j), \quad j = 0, \ldots, n \tag{3}$$

   $$\implies \quad \sum_{i=0}^{n} d_i L_i(x_j) = f(x_j), \quad j = 0, \ldots, n \tag{4}$$

   $$\implies \quad \begin{bmatrix} L_0(x_0) & L_1(x_0) & \cdots & L_{n-1}(x_0) & L_n(x_0) \\ L_0(x_1) & L_1(x_1) & \cdots & L_{n-1}(x_1) & L_n(x_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ L_0(x_{n-1}) & L_1(x_{n-1}) & \cdots & L_{n-1}(x_{n-1}) & L_n(x_{n-1}) \\ L_0(x_n) & L_1(x_n) & \cdots & L_{n-1}(x_n) & L_n(x_n) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix} \tag{5}$$

But remember that we said in class that the Lagrange basis functions have the properties $L_i(x_i) = 1$ and $L_i(x_j) = 0$ for $j \neq i$. So the linear system we wrote in (5) can be written much more cleanly as

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix} \tag{6}$$

How cool is that? These special basis functions result in a linear system that is trivial to solve for the coefficients d_j. The relation is simply $d_j = f(x_j)$.

So we can get our coefficients for any value of n that we desire and express the inerpolant p_n using (1). Let's use Matlab (all code attached at the end) to compute the d_j and plot the resulting interpolant for $n = 2, 4, 12, 24$. The plots for the four different values of n are shown in figure 1. It looks like the interpolant is doing a better and better job of interpolating $f(x)$ even though we are only sampling $f(x)$ over a subdomain of $[-5, 5]$! Let's make this more precise: in figure 2 we see that the maximum error $\max_{x\in[-5,5]} |f(x) - p_n(x)|$ is indeed going down as we increase n!

As I mentioned at the beginning of this document, do not be mislead by this remarkable result. In general, using uniformly spaced points—even over the entire domain that $f(x)$ is defined on—can lead to a catastrophically nonconvergent polynomial interpolant. It is therefore vastly more prudent to use Chebyshev points defined over the entire domain. You will explore this in homework 2.

I have hid one last important fact from you. It turns out that numerically evaluating Lagrange polynomials using (2) is not very numerically stable. It will not be an issue for the problems we consider in this class, but for very large n there is a faster and more stable way of computing these basis functions called *barycentric interpolation*, a beautiful technique we won't dive into here.

One last brainteaser: what would have happened if we had used the monomial basis instead of the Lagrange basis?

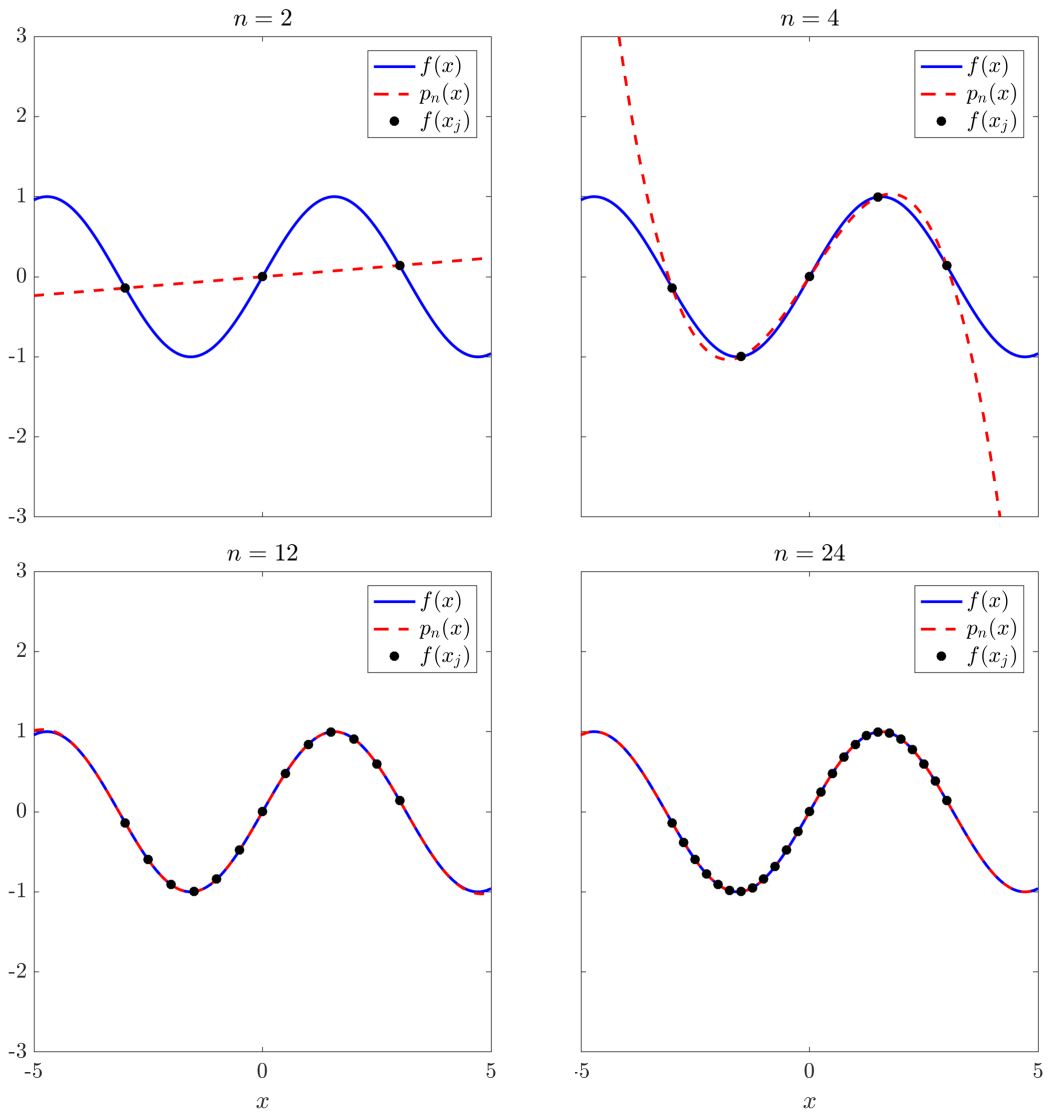

Figure 1: Plots of the polynomial interpolant for different values of n.

```
clear all, close all, clc

%Polynomial degree
nvect = [2; 4; 12; 24];

%function to approx
f = @(x) sin(x);

%error vector:
err = zeros(size(nvect));

for j = 1 : length( nvect )

    %define current n
```

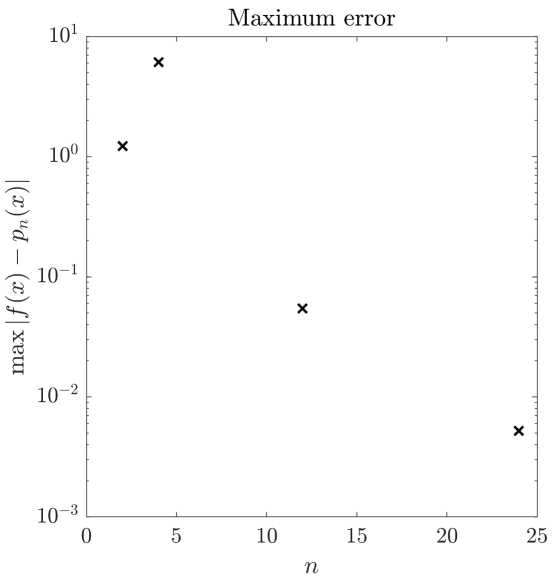

Figure 2: Error of the polynomial interpolant for different values of n.

```
n = nvect( j );

%define interp points
xj = -3 : 6/n : 3;

%transpose to make column vector
xj = xj';

%define values of f at interp points
fj = f(xj);

%define coefficients
dj = fj;

%--plot the polynomial interpolant

    xx = linspace( -5, 5, 1000 );

    pn = 0;
    for i = 1 : n + 1

        %define Lagrange basis vectors
        L_i = @(x) 1;

        %vector indexing can't start at zero, so go from 1 to n+1
        for k = 1 : n + 1
            if k ~= i
                L_i = @(x) ( x - xj(k) )./( xj(i) - xj(k) ) .* L_i(x);
            end
        end
        pn = pn + fj(i) * L_i( xx );
```

```
        end

        figure(j)
        plot( xx, f(xx), 'b-', 'linewidth', 2 ), hold on
        plot( xx, pn, 'r--', 'linewidth', 2 )
        plot( xj, f(xj), 'k.', 'markersize', 24 )

        %make plot pretty
        title( ['$n = ', num2str( n ),'$'] ,'interpreter', 'latex',...
            'fontsize', 16)
        xlabel( '$x$', 'interpreter', 'latex', 'fontsize', 16)
        h = legend( '$f(x)$', '$p_n(x)$', '$f(x_j)$');
        set(h, 'location', 'NorthEast', 'Interpreter', 'Latex', 'fontsize', 16 )
        set(gca, 'TickLabelInterpreter','latex', 'fontsize', 16 )
        axis([-5 5 -3 3])

        set(gcf, 'PaperPositionMode', 'manual')
        set(gcf, 'Color', [1 1 1])
        set(gca, 'Color', [1 1 1])
        set(gcf, 'PaperUnits', 'centimeters')
        set(gcf, 'PaperSize', [15 15])
        set(gcf, 'Units', 'centimeters' )
        set(gcf, 'Position', [0 0 15 15])
        set(gcf, 'PaperPosition', [0 0 15 15])

        svnm = ['pic_', num2str(j)];
        print( '-dpng', svnm, '-r200' )


    %--

    %--compute error
        err(j) = max(abs( f(xx) - pn ) );


end


%plot error
figure(100)
semilogy( nvect, err, 'kx', 'markersize', 8, 'linewidth', 2 )

%make plot pretty
title( 'Maximum error' ,'interpreter', 'latex','fontsize', 16)
xlabel( '$n$', 'interpreter', 'latex', 'fontsize', 16)
ylabel( '$\max|f(x) - p_n(x)|$', 'interpreter', 'latex', 'fontsize', 16)

set(gca, 'TickLabelInterpreter','latex', 'fontsize', 16 )

set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
```

```
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])

svnm = 'error';
print( '-dpng', svnm, '-r200' )
```