The Interpolation Problem

1 Intro

1. **Definition:** The function interpolation problem is, given data points

   $$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

   find a function f so that $y_i = f(x_i)$ for $i = 1..n$.

2. There are an infinite number of possible functions- This is an ill-defined problem. Perhaps the simplest functions to consider would be the set of polynomials.

3. **Polynomial Interpolation Theorem:** Given a set of $n + 1$ points with distinct x values, there is a unique interpolating polynomial of degree n or less that interpolates the points.

   You may have seen this in linear algebra. Given $n + 1$ points, we construct $n + 1$ equations in $n + 1$ unknowns. In the following equation, let a_k be the coefficient for x^k in the unknown polynomial, and let (x_k, y_k) be the k^{th} data point (given in the problem). The j^{th} equation is given by:

   $$a_n(x_j)^n + a_{n-1}(x_j)^{n-1} + \ldots a_1(x_j) + a_0 = y_j$$

   In matrix-vector form (the matrix is $n + 1 \times n + 1$), the equation looks like:

   $$\begin{bmatrix} x_1^n & x_1^{n-1} & \ldots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \ldots & x_2 & 1 \\ \vdots & & & \vdots & \vdots \\ x_{n+1}^n & x_{n+1}^{n-1} & \ldots & x_{n+1} & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \end{bmatrix}$$

   If the points x_i are distinct, then the matrix is invertible, and that gives you the coefficients of your polynomial.

2 The Lagrange Polynomial

First, consider a line through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$. If we define

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \qquad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

then $L_0(x_0) = 1$, $L_0(x_1) = 0$. Similarly, $L_1(x_0) = 0$ and $L_1(x_1) = 1$. With these, we can write the equation of the interpolating line as:

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

(Verify that $P(x_0) = f(x_0)$ and $P(x_1) = f(x_1)$)
Similarly, for a quadratic going through three points,

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \quad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

then

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2)$$

These are the Lagrange Polynomials. To generalize to $n+1$ points with index $k = 0, 1, 2, \ldots, n$ we form

$$L_k(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

with $n+1$ points, L_k is a degree n polynomial, and we see that $L_k(x_j) = 0$ if $j \neq k$ and $L_k(x_k) = 1$. Note the relationship between the index k and the construction of the polynomial $L_k(x)$.
The Lagrange polynomial is then constructed as a linear combination of the L_k:

$$P(x) = L_0(x)f(x_0) + \ldots + L_n(x)f(x_n) \tag{1}$$

Note: We said before that the interpolating polynomial is *unique*. The polynomial that comes from the matrix equation we wrote in the previous section is the **same** as the Lagrange polynomial- The Lagrange polynomial is easier to write down, and we get an immediate function to work with.

2.1 Exercises:

1. Write the Lagrange polynomial for the data:

x	-1	0	1
y	1	-1	2

2. Use the Matlab function `lagrange.m` (download from our class website) to plot the functions L_0, L_1, L_2 for the data whose $x-$coordinates are given by: $1, 2, 3, 4$.

   EXAMPLE: To plot L_1, type:

```
x=linspace(1,4);  %These are x where P(x) is eval'd
y=lagrange(x,[1,2,3,4],[0,1,0,0]);
plot(x,y);
grid on;    %Makes it easier to see the axes values
```

Before continuing with our analysis, we need some theory from Calculus.

1. **Generalized Rolle's Theorem**

   (I) If f is continuous on $[a, b]$ and differentiable on (a, b), and $f(a) = f(b) = 0$, then there exists c in (a, b) such that $f'(c) = 0$.

   (II) If f' is continuous on $[a, b]$ and differentiable on (a, b) and there are points a_0, a_1, a_2 in $[a, b]$ where $f(a_0) = f(a_1) = f(a_2) = 0$, then there exists c in (a, b) such that $f''(c) = 0$.

   Exercise: How does (II) follow from (I)?

   To generalize our result, let $f, f', \ldots, f^{(n)}$ all be continuous on $[a, b]$, and suppose we have $n + 1$ roots of f in $[a, b]$. Then there exists $c \in (a, b)$ so that $f^{(n)}(c) = 0$.

2. Let $x_0, x_1, \ldots, x_n$ be $n + 1$ real numbers, and let x be fixed (so its also constant). Consider the function of t:

$$\Psi(t) = \frac{(t - x_0)(t - x_1) \cdots (t - x_n)}{(x - x_0)(x - x_1) \cdots (x - x_n)}$$

   (a) $\Psi(t)$ is a polynomial of what degree?

   (b) What is $\Psi(x_j)$, $j = 0, 1, 2, \ldots, n$? What is $\Psi(x)$?

   (c) What is the $n + 1$st derivative of Ψ, with respect to t? (HINT below)
   It is possible to write $\Psi(t)$ as:

$$\Psi(t) = \left(\frac{1}{(x - x_0)(x - x_1) \cdots (x - x_n)} \right) t^{n+1} + \text{ terms with lower powers of } t$$

3. Now suppose that we have $n + 1$ points,

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$$

   and suppose that $P(x)$ is the (n^{th} degree) Lagrange polynomial interpolating those points (see Equation 1). The question: Is there a way to estimate how far $f(x)$ is away from $P(x)$? Consider the following function of t:

$$g(t) = f(t) - P(t) - (f(x) - P(x))\Psi(t) \tag{2}$$

   From the previous exercises, the roots of g are when

$$t = x, x_0, x_1, x_2, \ldots, x_n$$

   so that $g(t) = 0$ has $n + 2$ solutions. Now, from the generalized Rolle's Theorem, we know there exists c in the interval from the smallest to the largest of the numbers $x, x_0, x_1, \ldots, x_n$ so that

$$g^{(n+1)}(c) = 0$$

Now, differentiate both sides of Equation 2 $(n+1)-$times with respect to t, and evaluate at c:

$$0 = f^{(n+1)}(c) - 0 - (f(x) - P(x))\Psi^{(n+1)}(c)$$

$$0 = f^{(n+1)}(c) - (f(x) - P(x))\frac{(n+1)!}{(x-x_0)\cdots(x-x_n)}$$

so that:

$$f(x) = P(x) + \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)\cdots(x-x_n)$$

We have proven the following theorem, which gives the error between a function $f(x)$ and the interpolating polynomial, $P(x)$:

Interpolation Error Theorem: Let $x_0, x_1, \ldots, x_n$ be $(n+1)$ distinct numbers in $[a,b]$, and let f be $n+1$ times continuously differentiable on $[a,b]$. Then, for each x in $[a,b]$, there exists a number c in (a,b) such that:

$$f(x) = P(x) + \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)\cdots(x-x_n)$$

Notes about the theorem:

- Compare this error term to the error term in the Taylor Series.
- The error we have here *depends on* x. If we know that the derivatives are all bounded, then we can make an estimate on the error- That is, if $|f^{(n+1)}(x)| \leq M$ for all x in $[a,b]$, then:

$$|f(x) - P(x)| = \left|\frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)\cdots(x-x_n)\right| \leq \frac{M}{(n+1)!}|(x-x_0)\cdots(x-x_n)|$$

- EXAMPLE 1: Let $f(x) = \sin(x)$, $x = 0, \pi/6, \pi/3, \pi/2$. Assess the accuracy of the interpolating polynomial $P(x)$ on $[0, \pi/2]$. In particular, consider the error at the midpoint of each subinterval ($x = \frac{\pi}{12}, \frac{\pi}{4}, \frac{5\pi}{12}$)

  SOLUTION: From the error formula,

$$\sin(x) - P(x) = \frac{\sin(c)}{4!} \cdot x\left(x - \frac{\pi}{6}\right)\left(x - \frac{\pi}{3}\right)\left(x - \frac{\pi}{2}\right)$$

$$|\sin(x) - P(x)| \leq \frac{1}{24} \cdot \left|x\left(x - \frac{\pi}{6}\right)\left(x - \frac{\pi}{3}\right)\left(x - \frac{\pi}{2}\right)\right|$$

  At $\pi/12$, we get:

$$|\sin(\pi/12) - P(\pi/12)| \leq \frac{1}{24} \cdot \frac{\pi}{12} \cdot \frac{\pi}{12} \cdot \frac{3\pi}{12} \cdot \frac{5\pi}{12} = \frac{5}{8}\left(\frac{\pi}{12}\right)^4 \approx 0.002936$$

  The actual error I computed in Matlab as:

```
%% Individual Error:
x=[pi/12,pi/4,5*pi/12];
xp=[0,pi/6,pi/3,pi/2];
y=lagrange(x,xp,sin(xp));  %Download this from our website
Error=abs(y-sin(x))        %Leave off the semicolon to output

%% To plot all, actual, error:
xx=linspace(0,pi/2);
yy=lagrange(xx,xp,sin(xp));
ActualError=abs(yy-sin(xx));
plot(xx,ActualError);

%% To plot the actual error vs. error bound:
ErrorBd=(1/24).*xx.*(xx-pi/6).*(xx-pi/3).*(xx-pi/2);
plot(xx,ActualError,xx,abs(ErrorBd))
```

In this case, the actual error was 0.001798, about half of the upper bound.

We could repeat this for the other points (we leave this as an exercise (see below)). One thing to note about the error plot given in Matlab: Notice that the error goes down in the interior of $[0, \pi/2]$, and we get more error on the two extremes, close to 0 and $\pi/2$. This is generally the case.

3 Exercises

1. Type the previous Matlab commands in a script file and name it, for example, `Error1.m`. Some things to note:

   - The double percentage signs splits the work into three pieces (in Matlab, they call this "Cell Mode").
   - To run the script, go back to the command window and type `Error1` (or whatever the filename is). The second plot function overwrites the first, so you'll only see one figure.
   - To output the results into a nice format, go back to the editor, go to `File`, then `Publish to HTML`. Matlab will create a webpage that has all the output nicely arranged, and even creates a table of contents based on the cell titles! This is a great way to turn in Matlab-based homework problems!

2. Let $f(x) = \mathrm{e}^x$, let $P(x)$ be its interpolating polynomial at $x = -1, -1/2, 1/2, 1$.

   - By hand, construct the Lagrange polynomial, $P(x)$.
   - By hand, find the upper bound for the error, $|f(x) - P(x)|$ at the point $x = \frac{1}{4}$. HINT: For $f'(c)$, consider $f'(x)$ on the interval $[0, 1]$.

3. We want to find the solution to $e^x+\sin(x)-4=0$ using Inverse Quadratic Interpolation. Start the algorithm with $x_0=1, x_1=2$ and $x_2=0$ and update by retaining the three most recent iterates. Continue until $|f(x)|<10^{-10}$. Here's a "skeleton" script file to fill in:

```
f=inline('exp(x)+sin(x)-4');
x=[1, 2, 0];  %Our three points
y=f(x);
Tol=1e-10;

while abs(y(3))>Tol
  %  Compute the new estimate, x3:
  %  Use the function from the website, lagrange

  %Update x and y:

end
```

4 Newton's Divided Differences

Newton's divided differences give another way to construct the interpolating polynomial- One that is computationally more attractive, since it is in a form that is ready for Horner's method of evaluating polynomials.
First, some notation needs to be defined. View the data as coming from some function f, and list the points in a table:

$$\begin{array}{c|c} x_1 & f(x_1) \\ x_2 & f(x_2) \\ \vdots & \vdots \\ x_n & f(x_n) \end{array}$$

Now define the divided differences, which are defined recursively:

$$f[x_k]=f(x_k)$$

$$f[x_k,x_{k+1}]=\frac{f[x_{k+1}]-f[x_k]}{x_{k+1}-x_k}$$

$$f[x_k,x_{k+1},x_{k+2}]=\frac{f[x_{k+1},x_{k+2}]-f[x_k,x_{k+1}]}{x_{k+2}-x_k}$$

$$f[x_k,x_{k+1},x_{k+2},x_{k+3}]=\frac{f[x_{k+1},x_{k+2},x_{k+3}]-f[x_k,x_{k+1},x_{k+2}]}{x_{k+3}-x_k}$$

These numbers are the coefficients for the interpolating polynomial for the data, where:

$$P(x)=f[x_1]+f[x_1,x_2](x-x_1)+f[x_1,x_2,x_3](x-x_1)(x-x_2)+$$

$$f[x_1, x_2, x_3, x_4](x-x_1)(x-x_2)(x-x_3)+\ldots+f[x_1, \ldots, x_n](x-x_1)(x-x_2)\cdots(x-x_{n-1})$$

We'll wait for the proof. For now, we want to notice that the formula gives an efficient way for evaluating the polynomial- It's already nested (we'll see this more in depth shortly).
For now, consider the interpolating polynomial on three points, $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. To compute the coefficients, build a table:

$$\begin{array}{c|l} x_1 & y_1 = f[x_1] \\ & \qquad f[x_1x_2] = \frac{y_2-y_1}{x_2-x_1} \\ x_2 & y_2 = f[x_2] \qquad\qquad f[x_1x_2x_3] = \frac{f[x_2x_3]-f[x_1x_2]}{x_3-x_1} \\ & \qquad f[x_2x_3] = \frac{y_3-y_2}{x_3-x_2} \\ x_3 & y_3 = f[x_3] \end{array}$$

The polynomial coefficients are along the top edge of the triangle.
For example, build the interpolating polynomial through $(0, 1), (2, 2), (3, 4)$:

$$\begin{array}{c|l} 0 & 1 \\ & \qquad \frac{2-1}{2-0} = \frac{1}{2} \\ 2 & 2 \qquad\qquad \frac{2-(1/2)}{3-0} = \frac{1}{2} \\ & \qquad \frac{4-2}{3-2} = 2 \\ 3 & 4 \end{array}$$

Therefore, the interpolating polynomial is:

$$P(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2) = 1 + \frac{1}{2}x + \frac{1}{2}x(x-2)$$

Which can be written in nested form directly, in anticipation of using Horner's Method for evaluation:

$$P(x) = 1 + (x-0)\left(\frac{1}{2} + (x-2)\cdot\frac{1}{2}\right)$$

The table also makes it easy to add a point.
Example: Add the point $(1, 0)$ to the previous list. We simply put the necessary values in at the bottom of the triangle- Note that the $x-$values did not need to be in order:

$$\begin{array}{c|l} 0 & 1 \\ & \qquad\quad \frac{1}{2} \\ 2 & 2 \qquad\qquad \frac{1}{2} \\ & \qquad\quad 2 \qquad\qquad \frac{0-1/2}{1-0} = -\frac{1}{2} \\ 3 & 4 \qquad\qquad \frac{2-2}{1-2} = 0 \\ & \qquad \frac{0-4}{1-3} = 2 \\ 1 & 0 \end{array}$$

Now the new polynomial is (first in normal, then in nested form):

$$P(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2) = 1 + \frac{1}{2}x + \frac{1}{2}x(x-2) - \frac{1}{2}x(x-2)(x-3)$$

$$P(x) = 1 + x\left(\frac{1}{2} + (x-2)\left(\frac{1}{2} + (x-3)\cdot\frac{-1}{2}\right)\right)$$

Newton's Divided Differences, continued from "The Interpolation Problem"

Example: Find the interpolating polynomial passing through the following points:

x	0	1	2	3
y	2	1	0	-1

Construct the divided difference triangle:

$$\begin{array}{c|cccc} 0 & 2 & & & \\ & & -1 & & \\ 1 & 1 & & 0 & \\ & & -1 & & 0 \\ 2 & 0 & & 0 & \\ & & -1 & & \\ 3 & -1 & & & \end{array}$$

Reading off the coefficients, $P(x) = 2 + (-1)(x-0) = 2 - x$. Did you notice that the $y-$values were decreasing linearly?

4.1 Code for Newton's Interpolation

Here is a short piece of Matlab code that computes the polynomial coefficients using Newton's Divided Differences.
Here we also see something slightly different about the function call- Here we have three inputs and two outputs. The output variable `v` contains all of the divided differences, and the vector `c` contains only the top part of the triangle (those are the polynomial coefficients).

```
function [v,c]=newtdd(x,y,n)
for j=1:n
  v(j,1)=y(j);        % Fill in y column of Newton triangle
end
for i=2:n             % For column i,
  for j=1:n+1-i       % fill in column from top to bottom
    v(j,i)=(v(j+1,i-1)-v(j,i-1))/(x(j+i-1)-x(j));
  end
end
for i=1:n
  c(i)=v(1,i);        % Read along top of triangle
end                   % for output coefficients
```

Example: How many polynomials of degree less than 5 pass through the following points?

x	-1	0	2	3
y	-5	-1	1	11

Here is the Matlab version of the divided difference table:

```
xp=[-1 0 2 3];
yp=[-5 -1 1 11];
[v,c]=newtdd(xp,yp,4);
c =
    -5     4    -1     1
v =
    -5     4    -1     1
    -1     1     3     0
     1    10     0     0
    11     0     0     0
%Plot the polynomial over more points:
xx=linspace(-1,3);
yy=nest(3,c,xx,xp);
plot(xx,yy,xp,yp,'r*')  %Plot curve and data points
```

Here we see how our "triangle" was stored, and we can read off the necessary information. Because the polynomial coefficients were not all zero as in our previous example, we know that there are no lines or quadratics that pass through our data points, and only one cubic:

$$P_3(x) = -5 + 4(x+1) - (x+1)x + (x+1)x(x-2)$$

There are infinitely many degree 4 polynomials. Here is one way to get an infinite number of them:

$$P_4(x) = P_3(x) + k(x+1)x(x-2)(x-3)$$

Similarly, there are infinitely many degree 5 polynomials, constructed in a similar way. Here are some examples:

$$P_5(x) = P_3(x) + k(x+1)^2x(x-2)(x-3)$$

or

$$P_5(x) = P_3(x) + k(x+1)x^2(x-2)(x-3)$$

4.2 Exercises:

1. Use Lagrange interpolation to find a polynomial that goes through the following points. Use Newton's Divided Differences to also find that polynomial, and verify that they are the same.

x	-1	2	3	5
y	0	1	1	2

2. How many degree d polynomials go through the four points $(-1,3)$, $(1,1)$, $(2,3)$, $(3,7)$? Write down one, if possible: (a) $d=2$, (b) $d=3$, (c) $d=6$

3. Give an example, or explain why it doesn't exist:

   (a) A polynomial of degree 6 that is zero at $x = 1,2,3,4,5,6$ and is 10 at $x = 7$.

(b) A polynomial of degree 6 that is zero at $x = 1, 2, 3, 4, 5, 6$ and is 10 at $x = 7$, and is 70 at $x = 8$.

4. Use `newtdd.m` and `nest.m` to see how well an interpolating polynomial approximates the function $f(x)$ on the interval $[-1, 1]$, and using evenly spaced points.

$$f(x) = \frac{1}{1 + 12x^2}$$

First, use 7 points and plot the points, $f(x)$ and the interpolating polynomial. Next, generate a new graph using 15 points, plot the points, $f(x)$ and the interpolating polynomial. Repeat with 25 points.

What do you see in terms of the error?

HINT: Below is the beginning of a script file. To turn in the solution to this problem, have Matlab export this as an HTML page, and print that.

```
f=inline('1./(1+12*x.^2)');
%% Example 1:  Seven points
%First define the points for the polynomial:
xp=linspace(-1,1,7);
yp=f(xp);

%Find the coefficients:
[v,c]=newtdd(xp,yp);
xx=linspace(-1,1,200);  %Pts for f, P
yy1=f(xx);
yy2=nest(c,xx,xp);
plot(xp,yp,'r*',xx,yy1,xx,yy2);
legend('Data','f(x)','Polynomial')

%% Example 2:  15 Points


%% Example 3:  25 Points
```