**APPM 4650 — Newton Divided differences**

Recall the theorem from last class: Suppose $x_0, x_1, \ldots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each $x \in [a, b]$, there exists an $\eta$ (not generally known) between $x_0, x_1, \ldots, x_n$ in $[a, b]$ such that

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

where $p_n$) is the Lagrange approximation of degree $n$ through $\{x_j\}_{j=0}^n$.

This theorem has two implications. First, polynomial interpolation is exact for polynomials of degree $n$ with data at $n + 1$ nodes. Second, polynomial interpolation is unique.

**Proof of uniqueness of polynomial interpolation**

This is a proof by contradiction. We assume that there are two polynomials that interpolate with the given data. Let $p_n(x)$ and $q_n(x)$ denote two different interpolating polynomials of order $n$. We know by construction that $p_n(x_j) = q_n(x) = f(x_j)$ for each interpolation node $x_j$ $j = 0, \ldots, n$.

Define $w(x) = p_n(x) - q_n(x)$ which is polynomial of degree at most $n$. We know $w(x_j) = p_n(x_j) - q_n(x_j) = 0$ for each $j = 0, \ldots, n$. This means $w$ has $n + 1$ roots. The only way that this can be true is if $w(x) = 0$. Thus $p_n = q_n$

Now that we know that the polynomial interpolation is unique, we can write our interpolating polynomials in different ways.

For example, let's write

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \cdots (x - x_{n-1})$$

When we plug in $x_0$, we find $p_n(x_0) = a_0 = f(x_0)$.

When we plug in $x_1$, we find $p_n(x_1) = a_0 + a_1(x_1 - x_0) = f(x_1)$.
Solving for $a_1$ we find $a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} := f[x_0, x_1]$

When we plug in $x_2$, we find $p_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f(x_2)$.
Solving for $a_1$ we find $a_2 = \frac{f(x_1) - f[x_0, x_1](x_1 - x_0)}{(x_2 - x_0)(x_2 - x_1)} := f[x_0, x_1, x_2]$

We can continue this or we can get the coefficients from an easy to evaluate table. Then you can simply read the coefficients of the polynomial off the diagonal entries of the table. (In the text, this is Algorithm 3.2. The coefficients are stored in the matrix F.) The entries of the table are defined by the following recursive formula:

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

**Table needs to be inserted.**

To evaluate the Lagrange interpolation formula via the sum of Lagrange polynomials has a cost of $O(n^2)$. To use Newton Divided differences reduces the cost to $O(n)$ but the evaluation is often not stable.

Let's see if we can reduce the cost of the Lagrange evaluation while maintaining stability.

For $x \neq x_j$ the Lagrange polynomial $L_j(x)$ can be written as follows:

$$\begin{aligned} L_j(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\ &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_j)(x-x_{j+1})\cdots(x-x_n)}{(x-x_j)(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \end{aligned}$$

Now let

$$w_j = \frac{1}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$$

If we let $l(x) = (x-x_0)\cdots(x-x_n)$, then $w_j = \frac{1}{l'(x_j)}$. This means that we can write our Lagrange polynomials as

$$L_j(x) = \frac{l(x)w_j}{x-x_j}$$

and our interpolating polynomial is given by

$$p_n(x) = \sum_{j=0}^{n} L_j(x)f(x_j) = \sum_{j=0}^{n} \frac{l(x)f(x_j)w_j}{x-x_j} = l(x)\sum_{j=0}^{n} \frac{w_j f(x_j)}{x-x_j}$$

The cost of evaluating this expression is $O(n)$. It does cost $O(n^2)$ to build the weights $w_j$. This form of writing the interpolating polynomial is called *Barycentric interpolation*.