Numerical Analysis and Computing

Lecture Notes #5 — Interpolation and Polynomial Approximation
Divided Differences, and Hermite Interpolatory Polynomials

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Fall 2014

Outline

Recap and Lookahead

Previously:
Neville's Method to successively generate higher degree polynomial approximations **at a specific point.** — If we need to compute the polynomial at many points, we have to re-run Neville's method for each point. $\mathcal{O}(n^2)$ operations/point.

Algorithm: Neville's Method

To evaluate the polynomial that interpolates the $n+1$ points $(x_i, f(x_i))$, $i = 0, \dots, n$ at the point x:

1. Initialize $Q_{i,0} = f(x_i)$.
2. FOR $i = 1 : n$
   FOR $j = 1 : i$
   $$Q_{i,j} = \frac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$$
   END
   END
3. Output the Q-table.

Recap and Lookahead

Next:
Use divided differences to *generate the polynomials* themselves.*
* The coefficients of the polynomials. Once we have those, we can quickly (remember Horner's method?) compute the polynomial in any desired points. $\mathcal{O}(n)$ operations/point.

Algorithm: Horner's Method

Input: Degree n; coefficients $a_0, a_1, \dots, a_n$; x_0

Output: $y = P(x_0)$, $z = P'(x_0)$.

1. Set $y = a_n$, $z = a_n$
2. For $j = (n-1), (n-2), \dots, 1$
   Set $y = x_0 y + a_j$, $z = x_0 z + y$
3. Set $y = x_0 y + a_0$
4. Output (y, z)
5. End program

Representing Polynomials

If $P_n(x)$ is the n^{th} degree polynomial that agrees with $f(x)$ at the points $\{x_0, x_1, \ldots, x_n\}$, then we can (for the appropriate constants $\{a_0, a_1, \ldots, a_n\}$) write:

$$\begin{aligned} P_n(x) \quad = \quad & a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ & \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned}$$

Note that we can evaluate this "Horner-style," by writing

$$\begin{aligned} P_n(x) \quad = \quad & a_0 + (x - x_0)\,(a_1 + (x - x_1)\,(a_2 + \cdots \\ & \cdots + (x - x_{n-2})\,(a_{n-1} + a_n(x - x_{n-1})))) , \end{aligned}$$

so that each step in the Horner-evaluation consists of a subtraction, a multiplication, and an addition.

Finding the Constants $\{a_0, a_1, \ldots, a_n\}$ "Just Algebra"

Given the relation

$$\begin{aligned} P_n(x) \quad = \quad & a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ & \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned}$$

at $\mathbf{x_0}$: $a_0 = P_n(x_0) = f(x_0)$.

at $\mathbf{x_1}$: $f(x_0) + a_1(x_1 - x_0) = P_n(x_1) = f(x_1)$

$$\Rightarrow a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

at $\mathbf{x_2}$: $a_2 = \dfrac{f(x_2) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)} - \dfrac{f(x_1) - f(x_0)}{(x_2 - x_0)(x_1 - x_0)}$.

This gets massively ugly fast! — We need some nice clean notation!

Sir Isaac Newton to the Rescue: Divided Differences

Zeroth Divided Difference:

$$f[x_i] = f(x_i).$$

First Divided Difference:

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}.$$

Second Divided Difference:

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}.$$

k**th Divided Difference**:

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}.$$

The Constants $\{a_0, a_1, \ldots, a_n\}$ — Revisited

We had

at $\mathbf{x_0}$: $a_0 = P_n(x_0) = f(x_0)$.

at $\mathbf{x_1}$: $f(x_0) + a_1(x_1 - x_0) = P_n(x_1) = f(x_1)$

$$\Rightarrow a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

at $\mathbf{x_2}$: $a_2 = \dfrac{f(x_2) - f(x_0)}{(x_2 - x_0)(x_2 - x_1)} - \dfrac{f(x_1) - f(x_0)}{(x_2 - x_0)(x_1 - x_0)}$.

Clearly:

$$a_0 = f[x_0], \quad a_1 = f[x_0, x_1].$$

We may suspect that $a_2 = f[x_0, x_1, x_2]$, that is indeed so (a "little bit" of careful algebra will show it), and in general

$$\mathbf{a_k = f[x_0, x_1, \ldots, x_k]}.$$

Algebra: Chasing down $a_2 = f[x_0, x_1, x_2]$

$$\begin{aligned}
a_2 &= \frac{f(x_2)-f(x_0)}{(x_2-x_0)(x_2-x_1)} - \frac{f(x_1)-f(x_0)}{(x_2-x_1)(x_1-x_0)} \\
&= \frac{(f(x_2)-f(x_0))(x_1-x_0) - (f(x_1)-f(x_0))(x_2-x_0)}{(x_2-x_0)(x_2-x_1)(x_1-x_0)} \\
&= \frac{(x_1-x_0)f(x_2) - (x_2-x_0)f(x_1) + (x_2-x_0-x_1+x_0)f(x_0)}{(x_2-x_0)(x_2-x_1)(x_1-x_0)} \\
&= \frac{(x_1-x_0)f(x_2) - (\mathbf{x_1}-x_0+x_2-\mathbf{x_1})f(x_1) + (x_2-x_1)f(x_0)}{(x_2-x_0)(x_2-x_1)(x_1-x_0)} \\
&= \frac{(x_1-x_0)(f(x_2)-f(x_1)) - (x_2-x_1)(f(x_1)-f(x_0))}{(x_2-x_0)(x_2-x_1)(x_1-x_0)} \\
&= \frac{(f(x_2)-f(x_1))}{(x_2-x_0)(x_2-x_1)} - \frac{(f(x_1)-f(x_0))}{(x_2-x_0)(x_1-x_0)} \\
&= \frac{f[x_1,x_2]}{x_2-x_0} - \frac{f[x_0,x_1]}{x_2-x_0} = \mathbf{f[x_0,x_1,x_2]} \quad (!!!)
\end{aligned}$$

Newton's Interpolatory Divided Difference Formula

Hence, we can write

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} \left[f[x_0,\ldots,x_k] \prod_{m=0}^{k-1} (x-x_m) \right].$$

$$\begin{aligned}
P_n(x) \;=\; & f[x_0] + \\
& f[x_0,x_1](x-x_0) + \\
& f[x_0,x_1,x_2](x-x_0)(x-x_1) + \\
& f[x_0,x_1,x_2,x_3](x-x_0)(x-x_1)(x-x_2) + \cdots
\end{aligned}$$

This expression is known as **Newton's Interpolatory Divided Difference Formula.**

Computing the Divided Differences (by table)

x	f(x)	1st Div. Diff.	2nd Div. Diff.
x_0	$f[x_0]$		
		$f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0}$	
x_1	$f[x_1]$		$f[x_0,x_1,x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$
		$f[x_1,x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1}$	
x_2	$f[x_2]$		$f[x_1,x_2,x_3] = \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$
		$f[x_2,x_3] = \frac{f[x_3]-f[x_2]}{x_3-x_2}$	
x_3	$f[x_3]$		$f[x_2,x_3,x_4] = \frac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$
		$f[x_3,x_4] = \frac{f[x_4]-f[x_3]}{x_4-x_3}$	
x_4	$f[x_4]$		$f[x_3,x_4,x_5] = \frac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}$
		$f[x_4,x_5] = \frac{f[x_5]-f[x_4]}{x_5-x_4}$	
x_5	$f[x_5]$		

Note: The table can be extended with three *3rd* divided differences, two *4th* divided differences, and one *5th* divided difference.

Algorithm: Computing the Divided Differences

Algorithm: Newton's Divided Differences

Given the points $(x_i, f(x_i)),\ i = 0, \ldots, n$.

`Step 1:` Initialize $F_{i,0} = f(x_i),\ i = 0, \ldots, n$

`Step 2:`

`FOR` $i = 1 : n$

`FOR` $j = 1 : i$

$$F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$$

`END`

`END`

`Result:` The diagonal, $F_{i,i}$ now contains $f[x_0, \ldots, x_i]$.

A Theoretical Result: Generalization of the Mean Value Theorem

Theorem (Generalized Mean Value Theorem)

Suppose that $f \in C^n[a,b]$ and $\{x_0, \ldots, x_n\}$ are distinct number in $[a,b]$. Then $\exists\, \xi \in (a,b)$:

$$f[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

For $n = 1$ this is exactly the *Mean Value Theorem*...

So we have extended to MVT to higher order derivatives!

What is the theorem telling us?

— *Newton's n^{th} divided difference is in some sense an approximation to the n^{th} derivative of f.*

Newton vs. Taylor...

Using Newton's Divided Differences...

$$\begin{aligned} P_n^N(x) &= f[x_0] + f[x_0, x_1](x - x_0) + \\ & \quad f[x_0, x_1, x_2](x - x_0)(x - x_1) + \\ & \quad f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) + \cdots \end{aligned}$$

Using Taylor expansion

$$\begin{aligned} P_n^T(x) &= f(x_0) + f'(x_0)(x - x_0) + \\ & \quad \tfrac{1}{2!} f''(x_0)(x - x_0)^2 + \\ & \quad \tfrac{1}{3!} f'''(x_0)(x - x_0)^3 + \cdots \end{aligned}$$

It makes sense that the divided differences are approximating the derivatives in some sense!

Simplification: Equally Spaced Points

When the points $\{x_0, \ldots, x_n\}$ are equally spaced, *i.e.*

$$h = x_{i+1} - x_i, \; i = 0, \ldots, n-1,$$

we can write $x = x_0 + sh$, $x - x_k = (s - k)h$ so that

$$P_n(x) = P_n(x_0 + sh) = \sum_{k=0}^{n} s(s-1)\cdots(s-k+1)h^k f[x_0, \ldots, x_k].$$

Using the binomial coefficients, $\binom{s}{k} = \dfrac{s(s-1)\cdots(s-k+1)}{k!}$ —

$$\mathbf{P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} k!\, h^k\, f[x_0, \ldots, x_k].}$$

This is **Newton's Forward Divided Difference Formula.**

Notation, Notation, Notation...

Another form, **Newton's Forward Difference Formula** is constructed by using the forward difference operator Δ:

$$\Delta f(x_n) = f(x_{n+1}) - f(x_n)$$

using this notation:

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}\Delta f(x_0).$$

$$f[x_0, x_1, x_2] = \frac{1}{2h}\left[\frac{\Delta f(x_1) - \Delta f(x_0)}{h}\right] = \frac{1}{2h^2}\Delta^2 f(x_0).$$

$$f[x_0, \ldots, x_k] = \frac{1}{k!\, h^k}\Delta^k f(x_0).$$

Thus we can write **Newton's Forward Difference Formula**

$$\mathbf{P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f(x_0).}$$

Notation, Notation, Notation... Backward Formulas

If we reorder $\{x_0, x_1, \ldots, x_n\} \to \{x_n, \ldots, x_1, x_0\}$, and define the backward difference operator ∇:

$$\nabla f(x_n) = f(x_n) - f(x_{n-1}),$$

we can define the backward divided differences:

$$f[x_n, \ldots, x_{n-k}] = \frac{1}{k!\,h^k}\nabla^k f(x_n).$$

We write down **Newton's Backward Difference Formula**

$$\mathbf{P_n(x) = f[x_n] + \sum_{k=1}^{n}(-1)^k \binom{-s}{k} \nabla^k f(x_n)},$$

where

$$\binom{-s}{k} = (-1)^k \, \frac{s(s+1)\cdots(s+k-1)}{k!}.$$

Forward? Backward? I'm Confused!!!

x	$f(x)$	1st Div. Diff.	2nd Div. Diff.
x_0	$f[x_0]$		
		$\mathbf{f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0}}$	
			$\mathbf{f[x_0, x_1, x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}}$
x_1	$f[x_1]$		
		$f[x_1, x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1}$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3] = \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$
		$f[x_2, x_3] = \frac{f[x_3]-f[x_2]}{x_3-x_2}$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4] = \frac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$
		$f[x_3, x_4] = \frac{f[x_4]-f[x_3]}{x_4-x_3}$	
x_4	$f[x_4]$		$\mathbf{f[x_3, x_4, x_5] = \frac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}}$
		$\mathbf{f[x_4, x_5] = \frac{f[x_5]-f[x_4]}{x_5-x_4}}$	
x_5	$f[x_5]$		

Forward: The fwd div. diff. are the top entries in the table.

Backward: The bwd div. diff. are the bottom entries in the table.

Forward? Backward? — Straight Down the Center!

The Newton formulas works best for points close to the edge of the table; if we want to approximate $f(x)$ close to the center, we have to work some more...

x	$f(x)$	1st Div. Diff.	2nd Div. Diff.	3rd Div. Diff.	4th Div. Diff.
x_{-2}	$f[x_{-2}]$				
		$f[x_{-2}, x_{-1}]$			
x_{-1}	$f[x_{-1}]$		$f[x_{-2}, x_{-1}, x_0]$		
		$\mathbf{f[x_{-1}, x_0]}$		$\mathbf{f[x_{-2}, x_{-1}, x_0, x_1]}$	
x_0	$\mathbf{f[x_0]}$		$\mathbf{f[x_{-1}, x_0, x_1]}$		$\mathbf{f[x_{-2}, x_{-1}, x_0, x_1, x_2]}$
		$\mathbf{f[x_0, x_1]}$		$\mathbf{f[x_{-1}, x_0, x_1, x_2]}$	
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		$f[x_{-1}, x_0, x_1, x_2, x_3]$
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		
		$f[x_2, x_3]$			
x_3	$f[x_3]$				

We are going to construct **Stirling's Formula** — a scheme using **centered differences**. In particular we are going to use the **blue** (centered at x_0) entries, and averages of the **red** (straddling the x_0 point) entries.

Stirling's Formula — Approximating at Interior Points

Assume we are trying to approximate $f(x)$ close to the interior point x_0:

$$\begin{aligned}
P_n(x) \;=\; & P_{2m+1}(x) = f[x_0] + sh\,\frac{f[x_{-1}, x_0] + f[x_0, x_1]}{2} \\
& + s^2h^2\; f[x_{-1}, x_0, x_1] \\
& + s(s^2-1)h^3\, \frac{f[x_{-2}, x_{-1}, x_0, x_1] + f[x_{-1}, x_0, x_1, x_2]}{2} \\
& + s^2(s^2-1)h^4\; f[x_{-2}, x_{-1}, x_0, x_1, x_2] \\
& + \;\ldots \\
& + s^2(s^2-1)\cdots(s^2-(m-1)^2)h^{2m}\; f[x_{-m}, \ldots, x_m] \\
& + s(s^2-1)\cdots(s^2-m^2)h^{2m+1} \\
& \cdot \frac{f[x_{-m-1}, \ldots, x_m] + f[x_{-m}, \ldots, x_{m+1}]}{2}
\end{aligned}$$

If n is odd (can be written as $2m+1$), otherwise delete the last two lines.

Summary: Divided Difference Formulas

Newton's Interpolatory Divided Difference Formula

$$P_n(x) = f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) + f[x_0,x_1,x_2,x_3](x-x_0)(x-x_1)(x-x_2) + \cdots$$

Newton's Forward Divided Difference Formula

$$P_n(x_0+sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} k!\, h^k\, f[x_0,\ldots,x_k]$$

Newton's Backward Difference Formula

$$P_n(x) = f[x_n] + \sum_{k=1}^{n} (-1)^k \binom{-s}{k} \nabla^k f(x_n)$$

Reference: Binomial Coefficients

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}, \quad \binom{-s}{k} = (-1)^k\, \frac{s(s+1)\cdots(s+k-1)}{k!}$$

Homework #4 http://webwork.sdsu.edu

- Will open on 09/24/2014 at 09:30am PDT.
- Will close no earlier than 10/3/2014 at 09:00pm PDT.

Combining Taylor and Lagrange Polynomials

A **Taylor polynomial of degree** n matches the function and its first n derivatives at one point.

A **Lagrange polynomial of degree** n matches the function values at $n+1$ points.

Question: Can we combine the ideas of Taylor and Lagrange to get an interpolating polynomial that matches both the function values and some number of derivatives at multiple points?

Answer: To our euphoric joy, such polynomials exist! They are called **Osculating Polynomials.**

The Concise Oxford Dictionary:

Osculate 1. (arch. or joc.) kiss. **2.** (Biol., of species, etc.) be related through intermediate species etc., have common characteristics *with* another or with each other. **3.** (Math., of curve or surface) have contact of higher than first order with, meet at three or more coincident points.

Osculating Polynomials In Painful Generality

Given $(n+1)$ distinct points $\{x_0, x_1, \ldots, x_n\} \in [a,b]$, and non-negative integers $\{m_0, m_1, \ldots, m_n\}$.

Notation: Let $m = \max\{m_0, m_1, \ldots, m_n\}$.

The **osculating polynomial approximation** of a function $f \in C^m[a,b]$ at x_i, $i = 0,1,\ldots,n$ is the polynomial (of lowest possible order) that agrees with

$$\{f(x_i), f'(x_i), \ldots, f^{(m_i)}(x_i)\} \text{ at } x_i \in [a,b],\ \forall i.$$

The degree of the osculating polynomial is **at most**

$$M = n + \sum_{i=0}^{n} m_i.$$

In the case where $m_i = 1$, $\forall i$ the polynomial is called a **Hermite Interpolatory Polynomial**.

If $f \in C^1[a,b]$ and $\{x_0, x_1, \ldots, x_n\} \in [a,b]$ are distinct, the unique polynomial of least degree ($\leq 2n+1$) agreeing with $f(x)$ and $f'(x)$ at $\{x_0, x_1, \ldots, x_n\}$ is

$$\mathbf{H_{2n+1}(x)} = \sum_{\mathbf{j=0}}^{\mathbf{n}} \mathbf{f(x_j) H_{n,j}(x)} + \sum_{\mathbf{j=0}}^{\mathbf{n}} \mathbf{f'(x_j) \hat{H}_{n,j}(x)},$$

where

$$H_{n,j}(x) = \left[1 - 2(x - x_j) L'_{n,j}(x_j)\right] L^2_{n,j}(x)$$
$$\hat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x),$$

and $L_{n,j}(x)$ are our old friends, the **Lagrange coefficients**:

$$L_{n,j}(x) = \prod_{i=0,\, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}.$$

Further, if $f \in C^{2n+2}[a,b]$, then for some $\xi(x) \in [a,b]$

$$f(x) = H_{2n+1}(x) + \frac{\prod_{i=0}^{n}(x - x_i)^2}{(2n+2)!} f^{(2n+2)}(\xi(x)).$$

Recall: $L_{n,j}(x_i) = \delta_{i,j} = \begin{cases} 0, & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$ ($\delta_{i,j}$ is Kronecker's delta).

If follows that when $i \neq j$: $H_{n,j}(x_i) = \hat{H}_{n,j}(x_i) = 0$.

When $i = j$: $\begin{cases} H_{n,j}(x_j) = \left[1 - 2(x_j - x_j) L'_{n,j}(x_j)\right] \cdot 1 = 1 \\ \hat{H}_{n,j}(x_j) = (x_j - x_j) L^2_{n,j}(x_j) = 0. \end{cases}$

Thus, $\mathbf{H_{2n+1}(x_j) = f(x_j)}$.

$$\begin{aligned} H'_{n,j}(x) &= [-2L'_{n,j}(x_j)] L^2_{n,j}(x) + [1 - 2(x - x_j) L'_{n,j}(x_j)] \cdot 2 L_{n,j}(x) L'_{n,j}(x) \\ &= L_{n,j}(x) \left[-2L'_{n,j}(x_j) L_{n,j}(x) + [1 - 2(x - x_j) L'_{n,j}(x_j)] \cdot 2(x) L'_{n,j}\right] \end{aligned}$$

Since $L_{n,j}(x)$ is a factor in $H'_{n,j}(x)$: $H'_{n,j}(x_i) = 0$ when $i \neq j$.

$$\begin{aligned} H'_{n,j}(x_j) &= [-2L'_{n,j}(x_j)] \underbrace{L^2_{n,j}(x_j)}_{1} \\ &+ [1 - 2\underbrace{(x_j - x_j)}_{0} L'_{n,j}(x_j)] \cdot 2 \underbrace{L_{n,j}(x_j)}_{1} L'_{n,j}(x_j) \\ &= -2L'_{n,j}(x_j) + 1 \cdot 2 \cdot L'_{n,j}(x_j) = 0 \end{aligned}$$

i.e. $\mathbf{H'_{n,j}(x_i) = 0}$, $\forall \mathbf{i}$.

$$\begin{aligned} \hat{H}'_{n,j}(x) &= L^2_{n,j}(x) + 2(x - x_j) L_{n,j}(x) L'_{n,j}(x) \\ &= L_{n,j}(x) \left[L_{n,j}(x) + 2(x - x_j) L'_{n,j}(x)\right] \end{aligned}$$

If $i \neq j$: $\hat{H}'_{n,j}(x_i) = 0$, since $L_{n,j}(x_i) = \delta_{i,j}$.
If $i = j$: $\hat{H}'_{n,j}(x_j) = 1 \cdot \left[1 + 2(x_j - x_j) L'_{n,j}(x_j)\right] = 1$.

Hence, $\mathbf{H'_{2n+1}(x_i) = f'(x_i)}$, $\forall \mathbf{i}$. □

- Assume there is a second polynomial $G(x)$ (of degree $\leq 2n+1$) interpolating the same data.
- Define $R(x) = H_{2n+1}(x) - G(x)$.
- Then by construction $R(x_i) = R'(x_i) = 0$, *i.e.* all the x_i's are zeros of multiplicity at least 2.
- This can only be true if $R(x) = q(x) \prod_{i=0}^{n}(x - x_i)^2$, for some $q(x)$.
- If $q(x) \not\equiv 0$ then the degree of $R(x)$ is $\geq 2n+2$, which is a contradiction.
- Hence $q(x) \equiv 0 \Rightarrow R(x) \equiv 0 \Rightarrow H_{2n+1}(x)$ is unique. □

Main Use of Hermite Interpolatory Polynomials

One of the primary applications of Hermite Interpolatory Polynomials is the development of **Gaussian quadrature** for numerical integration. (To be revisited later this semester.)

The most commonly seen Hermite interpolatory polynomial is the cubic one, which satisfies

$$\begin{array}{ll} H_3(x_0) = f(x_0), & H_3'(x_0) = f'(x_0) \\ H_3(x_1) = f(x_1), & H_3'(x_1) = f'(x_1). \end{array}$$

it can be written explicitly as

$$\begin{aligned} H_3(x) \;=\; & \left[1 + 2\tfrac{x-x_0}{x_1-x_0}\right]\left[\tfrac{x_1-x}{x_1-x_0}\right]^2 f(x_0) + (x-x_0)\left[\tfrac{x_1-x}{x_1-x_0}\right]^2 f'(x_0) \\ + & \left[1 + 2\tfrac{x_1-x}{x_1-x_0}\right]\left[\tfrac{x-x_0}{x_1-x_0}\right]^2 f(x_1) + (x-x_1)\left[\tfrac{x-x_0}{x_1-x_0}\right]^2 f'(x_1). \end{aligned}$$

It appears in some optimization algorithms (see `Math 693a`, *linesearch algorithms*.)

Computing from the Definition is Tedious!

However, there is good news: we can re-use the algorithm for **Newton's Interpolatory Divided Difference Formula** with some modifications in the initialization.

We "double" the number of points, *i.e.* let

$$\{y_0, y_1, \ldots, y_{2n+1}\} = \{x_0, x_0+\epsilon, x_1, x_1+\epsilon, \ldots, x_n, x_n+\epsilon\}$$

Set up the divided difference table (up to the first divided differences), and let $\epsilon \to 0$ (formally), and identify:

$$f'(x_i) = \lim_{\epsilon \to 0} \frac{f[x_i+\epsilon] - f[x_i]}{\epsilon},$$

to get the table *[next slide]*...

Hermite Interpolatory Polynomial using Modified Newton Divided Differences

y	f(x)	1st Div. Diff.	2nd Div. Diff.	3rd Div. Diff.
$y_0 = x_0$	$f[y_0]$			
		$f[y_0, y_1] = f'(y_0)$		
$y_1 = x_0$	$f[y_1]$		$f[y_0, y_1, y_2]$	
		$f[y_1, y_2]$		$f[y_0, y_1, y_2, y_3]$
$y_2 = x_1$	$f[y_2]$		$f[y_1, y_2, y_3]$	
		$f[y_2, y_3] = f'(y_2)$		$f[y_1, y_2, y_3, y_4]$
$y_3 = x_1$	$f[y_3]$		$f[y_2, y_3, y_4]$	
		$f[y_3, y_4]$		$f[y_2, y_3, y_4, y_5]$
$y_4 = x_2$	$f[y_4]$		$f[y_3, y_4, y_5]$	
		$f[y_4, y_5] = f'(y_4)$		$f[y_3, y_4, y_5, y_6]$
$y_5 = x_2$	$f[y_5]$		$f[y_4, y_5, y_6]$	
		$f[y_5, y_6]$		$f[y_4, y_5, y_6, y_7]$
$y_6 = x_3$	$f[y_6]$		$f[y_5, y_6, y_7]$	
		$f[y_6, y_7] = f'(y_6)$		$f[y_5, y_6, y_7, y_8]$
$y_7 = x_3$	$f[y_7]$		$f[y_6, y_7, y_8]$	
		$f[y_7, y_8]$		$f[y_6, y_7, y_8, y_9]$
$y_8 = x_4$	$f[y_8]$		$f[y_7, y_8, y_9]$	
		$f[y_8, y_9] = f'(y_8)$		
$y_9 = x_4$	$f[y_9]$			

$H_3(x)$ revisited...

Old notation

$$\begin{aligned} H_3(x) \;=\; & \left[1 + 2\tfrac{x-x_0}{x_1-x_0}\right]\left[\tfrac{x_1-x}{x_1-x_0}\right]^2 f(x_0) + \left[1 + 2\tfrac{x_1-x}{x_1-x_0}\right]\left[\tfrac{x-x_0}{x_1-x_0}\right]^2 f(x_1) \\ + & (x-x_0)\left[\tfrac{x_1-x}{x_1-x_0}\right]^2 f'(x_0) + (x-x_1)\left[\tfrac{x-x_0}{x_1-x_0}\right]^2 f'(x_1). \end{aligned}$$

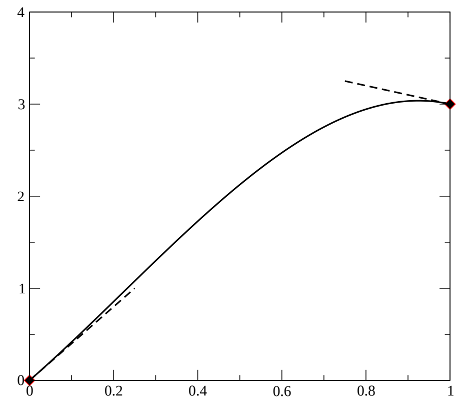
Divided difference notation

$$\begin{aligned} H_3(x) \;=\; & f(x_0) + f'(x_0)(x-x_0) + f[x_0, x_0, x_1](x-x_0)^2 \\ + & f[x_0, x_0, x_1, x_1](x-x_0)^2(x-x_1). \end{aligned}$$

Or with the y's...

$$\begin{aligned} H_3(x) \;=\; & f(y_0) + f'(y_0)(x-y_0) + f[y_0, y_1, y_2](x-y_0)(x-y_1) \\ + & f[y_0, y_1, y_2, y_3](x-y_0)(x-y_1)(x-y_2). \end{aligned}$$

$H_3(x)$ Example



$$x_0 = 0, \quad x_1 = 1$$

$$f(x_0) = 0, \quad f'(x_0) = 4, \quad f(x_1) = 3, \quad f'(x_1) = -1$$

$$H_3(x) = 4x - x^2 - 3x^2(x-1)$$

$H_3(x)$ Example — Not Very Pretty Computations

Example

```
x0  = 0; x1 = 1;                % This is the data
fv0 = 0; fpv0 = 4;
fv1 = 3; fpv1 = -1;

y0 = x0; f0=fv0;                % Initializing the table
y1 = x0; f1=fv0;
y2 = x1; f2=fv1;
y3 = x1; f3=fv1;

f01 = fpv0;                     % First divided differences
f12 = (f2-f1)/(y2-y1);
f23 = fpv1;

f012 = (f12-f01)/(y2-y0);       % Second divided differences
f123 = (f23-f12)/(y3-y1);

f0123 = (f123-f012)/(y3-y0);    % Third divided difference

x=(0:0.01:1)';

H3 = f0 + f01*(x-y0) + f012*(x-y0).*(x-y1) + ...
     f0123*(x-y0).*(x-y1).* (x-y2);
```


Algorithm: Hermite Interpolation

Algorithm: Hermite Interpolation, Part #1

Given the data points $(x_i, f(x_i), f'(x_i))$, $i = 0, \ldots, n$.

Step 1: FOR i=0:n

$y_{2i} = x_i$, $Q_{2i,0} = f(x_i)$, $y_{2i+1} = x_i$, $Q_{2i+1,0} = f(x_i)$

$Q_{2i+1,1} = f'(x_i)$

IF $i > 0$

$$Q_{2i,1} = \frac{Q_{i,0} - Q_{i-1,0}}{y_{2i} - y_{2i-1}}$$

END

END

Algorithm: Hermite Interpolation

Algorithm: Hermite Interpolation, Part #2

Step 2: FOR $i = 2 : (2n+1)$

FOR $j = 2 : i$

$$Q_{i,j} = \frac{Q_{i,j-1} - Q_{i-1,j-1}}{y_i - y_{i-j}}.$$

END

END

Result: $q_i = Q_{i,i}$, $i = 0, \ldots, 2n+1$ now contains the coefficients for

$$H_{2n+1}(x) = q_0 + \sum_{k=1}^{2n+1} \left[q_k \prod_{j=0}^{k-1} (x - y_j) \right].$$

So far we have seen the osculating polynomials of order 0 — the Lagrange polynomial, and of order 1 — the Hermite interpolatory polynomial.

It turns out that generating osculating polynomials of higher order is fairly straight-forward; — and we use Newton's divided differences to generate those as well.

Given a set of points $\{x_k\}_{k=0}^{n}$, and $\{f^{(\ell)}(x_k)\}_{k=0,\ell=0}^{n,\ell_k}$; *i.e.* the function values, as well as the first ℓ_k derivatives of f in x_k. (Note that we can specify a different number of derivatives in each point.)

> Set up the Newton-divided-difference table, and put in $(\ell_k + 1)$ duplicate entries of each point x_k, as well as its function value $f(x_k)$.

Run the computation of Newton's divided differences as usual; with the following exception:

> Whenever a zero-denominator is encountered — *i.e.* the divided difference for that entry cannot be computed due to duplication of a point — use a derivative instead. For m^{th} divided differences, use $\frac{1}{m!}f^{(m)}(x_k)$.

On the next slide we see the setup for two point in which two derivatives are prescribed.

y	**f(x)**	**1st Div. Diff.**	**2nd Div. Diff.**	**3rd Div. Diff.**
$y_0 = x_0$	$f[y_0]$			
		$f[y_0, y_1] = f'(x_0)$		
$y_1 = x_0$	$f[y_1]$		$f[y_0, y_1, y_2] = \frac{1}{2}f''(x_0)$	
		$f[y_1, y_2] = f'(x_0)$		$f[y_0, y_1, y_2, y_3]$
$y_2 = x_0$	$f[y_2]$		$f[y_1, y_2, y_3]$	
		$f[y_2, y_3]$		$f[y_1, y_2, y_3, y_4]$
$y_3 = x_1$	$f[y_3]$		$f[y_2, y_3, y_4]$	
		$f[y_3, y_4] = f'(x_1)$		$f[y_2, y_3, y_4, y_5]$
$y_4 = x_1$	$f[y_4]$		$f[y_3, y_4, y_5] = \frac{1}{2}f''(x_1)$	
		$f[y_4, y_5] = f'(x_1)$		
$y_5 = x_1$	$f[y_5]$			

3rd and higher order divided differences are computed "as usual" in this case.

On the next slide we see four examples of 2nd order osculating polynomials.

Examples...