

Lecture 09 - B

§4.1 Newton Interpolation

T. Takahashi

Department of Mathematics and Statistics
University of New Mexico



We will answer these questions:

- What is the benefit of nesting polynomials?
- What are the Newton polynomial basis functions?
- What type of linear system is involved in Newton interpolation?
- What are the advantages of Newton interpolation?
- How do you use the Tabular method for calculating interpolation by hand?



Nesting Polynomials



Polynomial Operation Count

- Consider the polynomial

$$p_3(x) = 5 + 4x + 7x^2 + 2x^3$$

- When implemented in Matlab, it is

$$p_3(x) = 5 + 4*x + 7*x*x + 2*x*x*x$$

- 3 additions AND (1+2+3) multiplications = 9 FLOPs



In General

- A general polynomial has the form

$$p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

- There are exactly n additions
- Zeroth term has zero multiplications
- The first term has one multiplication...
- The second term has two multiplication...

Number of FLOPs

In general, evaluating $p_n(x)$ costs n additions and $\sum_{i=1}^n i$ multiplications or

$$\text{Total} = n + \left(\frac{n(n+1)}{2} \right) \text{ FLOPs}$$

- We can rewrite this polynomial

$$\begin{aligned}p_3(x) &= 5 + 4x + 7x^2 + 2x^3 \\&= 5 + x(4 + 7x + 2x^2) \\&= 5 + x(4 + x(7 + 2x))\end{aligned}$$

- When implemented in Matlab, it is

$$p_3(x) = 5 + \mathbf{x}*(4 + \mathbf{x}*(7 + 2*\mathbf{x}))$$

- 3 additions AND 3 multiplications = 6 FLOPs



Optimal polynomial representation

- Given a polynomial

$$p(x) = -5 + 4x - 7x^2 + 2x^3 + 3x^4, \quad (14 \text{ FLOPs})$$

we can write this equivalently as

$$p(x) = -5 + x(4 + x(-7 + x(2 + 3x))), \quad (8 \text{ FLOPs})$$

- This technique is called **nesting polynomials** or **Horner's Method**

Method: Nesting

- ▶ **Given:** Standard polynomial
- ▶ **Output:** Nested polynomial
- ▶ **Idea:** Factor out x to minimize the number of multiplications.



Nesting

- The standard polynomial costs $\mathcal{O}(n^2)$ FLOPs to evaluate
- The fully nested polynomial costs exactly $2n$ FLOPs
- Compare

$$p(x) = -5 + 4x - 7x^2 + 2x^3 + 3x^4,$$

versus

$$p(x) = -5 + x(4 + x(-7 + x(2 + 3x)))$$

Key Idea

Nested polynomials are evaluated more optimally.



Summary so far:

- **Monomial basis:** $p(x) = a_0 + a_1x + \cdots + a_nx^n$ results in a poorly conditioned Vandermonde matrix that can lead to inaccurate coefficients.
- **Nesting:** It is trivial to minimize the cost of polynomial evaluation.
- **Lagrange basis:** $p(x) = \ell_0(x)y_0 + \cdots + \ell_n(x)y_n$. There is no error in the coefficients. It's easy to compute by hand.
 - ▶ When adding an additional data point, you cannot re-use previous polynomial easily (for monomials and Lagrange)



Newton Interpolation



Monomial Basis

- Consider interpolating the data

x_0	x_1	x_2
y_0	y_1	y_2

- Our polynomial using a **monomial basis** will be

$$p_2(x) = a + bx + cx^2$$

- We can write out a system of equations for a, b, c

$$\begin{aligned} \text{at } x_0 : \quad p(x_0) &= y_0 = a + bx_0 + cx_0^2, \\ \text{at } x_1 : \quad p(x_1) &= y_1 = a + bx_1 + cx_1^2, \\ \text{at } x_2 : \quad p(x_2) &= y_2 = a + bx_2 + cx_2^2. \end{aligned} \implies \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

- Solve the 3×3 **Vandermonde** system



Another Basis?

Question

What if we use this basis instead?

Function	Order
1	0
$x - x_0$	1
$(x - x_0)(x - x_1)$	2

- Then our interpolation would be

$$p_2(x) = a + b(x - x_0) + c(x - x_0)(x - x_1).$$

- The three unknowns are a , b , c



Conditions

- The polynomial must pass through (x_0, y_0) , (x_1, y_1) , and (x_2, y_2)
- The conditions are

$$y_0 = p_2(x_0)$$

$$y_1 = p_2(x_1)$$

$$y_2 = p_2(x_2)$$

- If $p_2(x) = a + b(x - x_0) + c(x - x_0)(x - x_1)$ then

$$\text{at } x_0 : y_0 = a + 0 + 0,$$

$$\text{at } x_1 : y_1 = a + b(x_1 - x_0) + 0,$$

$$\text{at } x_2 : y_2 = a + b(x_2 - x_0) + c(x_2 - x_0)(x_2 - x_1).$$



Matrix Form

- The matrix form of

$$\text{at } x_0 : y_0 = a + 0 + 0,$$

$$\text{at } x_1 : y_1 = a + b(x_1 - x_0) + 0,$$

$$\text{at } x_2 : y_2 = a + b(x_2 - x_0) + c(x_2 - x_0)(x_2 - x_1).$$

is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Key Idea

This is a **triangular system**! This is great news!



Comparison

Monomial Basis

Fitting a polynomial of the form $p(x) = a + bx + cx^2$ results in a **full matrix**

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Costs $\mathcal{O}(n^3)$ to solve

Alternate Basis

Fitting a polynomial of the form $p = a + b(x - x_0) + c(x - x_0)(x - x_1)$ results in a **triangular matrix**

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Costs $\mathcal{O}(n^2)$ to solve

Newton Polynomials

- **Newton Polynomials** are of the form

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \dots$$

- The **Newton basis functions** have the form:

Function	Order
1	0
$x - x_0$	1
$(x - x_0)(x - x_1)$	2
$(x - x_0)(x - x_1)(x - x_2)$	3

Definition: Newton Basis

The **Newton basis functions** are the family of polynomials

$$\mathcal{F} = \{1, x - x_0, (x - x_0)(x - x_1), \dots\}$$

where x_k are distinct data points.

Method: Newton Interpolation

- **Given:** Data $n + 1$ points (x_i, y_i) for $i = 0, \dots, n$
- **Output:** Interpolating polynomial $p_n(x)$
- **Idea:** The system obtained with the Newton interpolation is triangular.

- Monomial vs. Newton

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \text{ vs. } \begin{bmatrix} 1 & 0 & 0 \\ 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) \end{bmatrix}$$

- Can be solved in $\mathcal{O}(n^2)$ FLOPs



Solving for the Coefficients



Forward Substitution

- We can solve this system with forward substitution

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & x_1 - x_0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

- First row:

$$a_0 = y_0$$

- Second row:

$$a_0 + (x_1 - x_0)a_1 = y_1$$

$$a_1 = \frac{y_1 - a_0}{x_1 - x_0}$$

- Third row:

$$a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2 = y_2$$

$$a_2 = \frac{y_2 - a_0 - (x_2 - x_0)a_1}{(x_2 - x_0)(x_2 - x_1)}$$



Pattern for Coefficients?

Question

Do you see a pattern?

$$a_0 = y_0 = f(x_0)$$

$$a_1 = \frac{y_1 - a_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$\begin{aligned} a_2 &= \frac{y_2 - a_0 - (x_2 - x_0)a_1}{(x_2 - x_1)(x_2 - x_0)} \\ &= \frac{f(x_2) - f(x_0) - (x_2 - x_0)\frac{f(x_1) - f(x_0)}{x_1 - x_0}}{(x_2 - x_1)(x_2 - x_0)} \\ &= \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} \end{aligned}$$

There is a **recursive pattern** of $\frac{\Delta f}{\Delta x}$.



Divided Differences

- We see many instances of the fraction

$$\frac{f(x_j) - f(x_i)}{x_j - x_i}$$

- They're called **divided differences** and are denoted:

$$f[x_i, x_j] := \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

- For example

$$f[a, b] = \frac{f(a) - f(b)}{a - b}$$

- Notice that a and b can be switched

$$f[b, a] = \frac{f(b) - f(a)}{b - a} = \frac{-(f(a) - f(b))}{-(a - b)} = \frac{f(a) - f(b)}{a - b} = f[a, b]$$



Bracket Notation

- The **zeroth order divided difference** is just a base case

$$a_0 = f[x_0] = f(x_0)$$

- The **first order divided difference** is defined to be

$$a_1 = f[x_1, x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

- The **second order divided difference** is

$$a_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

- For higher order terms we use recursion

Recursive Property

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

Recursive Property

Recursive Property

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

$$f \underbrace{[x_0, \dots, x_k]}_{0 \text{ to } k} = \frac{\underbrace{[x_1, \dots, x_k]}_{1 \text{ to } k} - f \underbrace{[x_0, \dots, x_{k-1}]}_{0 \text{ to } k-1}}{x_k - x_0}$$

- Let's call $f[x_0, \dots, x_k]$ the k^{th} level divided difference
- k^{th} level divided difference requires two $(k-1)^{\text{th}}$ level divided differences



The Pattern

- These are the coefficients

$$a_0 = f(x_0)$$

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

$$a_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

- They can be expressed compactly using the **bracket notation**

$$a_0 = f[x_0]$$

$$a_1 = f[x_0, x_1]$$

$$a_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} =: f[x_0, x_1, x_2]$$



Newton Polynomial Coefficients

- The Newton polynomial is

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \text{etc.}$$

- The coefficients are

$$a_0 = f[x_0]$$

$$a_1 = f[x_0, x_1]$$

$$a_2 = f[x_0, x_1, x_2]$$

$$a_3 = f[x_0, x_1, x_2, x_3]$$

$$\vdots$$

$$a_n = f[x_0, x_1, \dots, x_n]$$

Key Idea

Just add one extra point in the bracket. For example

$$a_{n+1} = f[x_0, x_1, \dots, x_n, x_{n+1}]$$

Newton Polynomials using Divided Differences

Example

Find the 2nd order interpolating Newton polynomial for the data points

x	1	-4	0
y	3	13	-23

1. First, write the general 2nd degree Newton polynomial

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

2. Next, compute the coefficients

$$a_0 = f[x_0] = f[1] = f(1) = 3$$

$$a_1 = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{13 - 3}{-4 - 1} = -2$$

$$a_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{\frac{-23 - 13}{0 - -4} - \frac{13 - 3}{-4 - 1}}{0 - 1} = \frac{-9 + 2}{-1} = 7$$

3. Finally, substitute the values

$$p_2(x) = 3 - 2(x - 1) + 7(x - 1)(x + 4)$$



Invariance Theorem

$f[x_0, \dots, x_k]$ is invariant under all permutations of the arguments $x_0, \dots, x_k$

Why is this important?

- The nodes x_i do not have to be arranged from least to greatest
- We can apply the recursive formula easily
- If an additional interpolation point is added (x_{n+1}, y_{n+1}) , we can easily find the next coefficient



Example

Existing Interpolation

x	1	-4	0
y	3	13	-23

$$p_2(x) = 3 - 2(x - 1) + 7(x - 1)(x + 4)$$

Extra Point

If we add an extra point

x	1	-4	0	2
y	3	13	-23	3

then our new interpolation is

$$p_3(x) = 3 - 2(x - 1) + 7(x - 1)(x + 4) + a_3(x - 1)(x + 4)(x)$$

Key Idea

When adding a new point, we may reuse the previously calculated coefficients.

Newton Interpolation

Method: Newton Divided Differences

- **Given:** $n + 1$ data points (x_i, y_i)
- **Output:** Degree n interpolating polynomial $p_n(x)$
- **Idea:** Take advantage of the recursive formula of divided differences

Key Idea

Advantages to using Newton interpolation:

- Newton system can be solved in $\mathcal{O}(n^2)$ FLOPs
- When an additional points, we can build off of the existing interpolation



Newton Interpolation

Method: Newton Divided Differences

- **Given:** $n + 1$ data points (x_i, y_i)
- **Output:** Degree n interpolating polynomial $p_n(x)$
- **Idea:** Take advantage of the recursive formula of divided differences

Key Idea

Advantages to using Newton interpolation:

- Newton system can be solved in $\mathcal{O}(n^2)$ FLOPs
- When an additional points, we can build off of the existing interpolation
- Can be easily calculated by hand using the **tabular method**



Tabular Method



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	
x_2	$f[x_2]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$
x_3	$f[x_3]$	$f[x_2, x_3]$		



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0, x_1]$		
		$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$
		$f[x_2, x_3]$		
x_3	$f[x_3]$			



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0, x_1]$		
x_2	$f[x_2]$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
		$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$
x_3	$f[x_3]$			



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			



Divided Differences with the Table

Recursive Property

Use this to fill out the table

$$f[x_0, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0}$$

We can compute the divided differences much easier using tables. To construct the divided difference table for $f(x)$ for the $x_0, \dots, x_3$

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
3	6			
0	3			
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
3	6	3		
0	3			
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
3	6	3		
0	3	1		
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
		3		
3	6			
		1		
0	3			
		2		
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
		3		
3	6		2	
		1		
0	3			
		2		
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
		3		
3	6		2	
		1		
0	3		-1	
		2		
2	7			



Tabular Method Example

Example

Construct the divided differences table for the data

x	1	3	0	2
y	0	6	3	7

and construct the largest order interpolating polynomial.

Start by filling the first 2 columns with the given data. Then fill each column from left to right.

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
		3		
3	6		2	
		1		-3
0	3		-1	
		2		
2	7			



Writing the polynomial

x	$f[\cdot]$	$f[\cdot, \cdot]$	$f[\cdot, \cdot, \cdot]$	$f[\cdot, \cdot, \cdot, \cdot]$
1	0			
3	6	3	2	
0	3	1	-1	-3
2	7	2		

- After completing the table, write the general form for a cubic

$$p_3(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) - a_3(x - x_0)(x - x_1)(x - x_2)$$

- The coefficients a_0 through a_3 is along the upper diagonal

$$p_2(x) = 0 + 3(x - 1) + 2(x - 1)(x - 3) - 3(x - 1)(x - 3)(x - 0)$$



Math Joke Revisited

What is the next number in the sequence?

1, 2, 3, ____



Math Joke Revisited

What is the next number in the sequence?

1, 2, 3, ____

It's whatever you want it to be!

Create an interpolating polynomial that passes through

$(1, 1), (2, 2), (3, 3), (4, x)$

