Lecture 11

Interpolation

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

April, 2011

Interpolation: Introduction

Objective

Approximate an unknown function $f(x)$ by an easier function $g(x)$, such as a polynomial.

Objective (alt)

Approximate some data by a function $g(x)$.

Types of approximating functions:

1. Polynomials
2. Piecewise polynomials
3. Rational functions
4. Trig functions
5. Others (inverse, exponential, Bessel, etc)

Basis functions

How do we approximate a function $f(x)$ by $g(x)$?

Definition

We define the basis functions as a set of functions $\{\phi_j(x) \mid j = 0, \ldots\}$ and the space spanned by the basis functions as,

$$g(x) = \sum_{j=0}^{n} a_j \phi_j(x)$$

where a_j's can be any real numbers and n is finite.

The function $g(x)$ is said to interpolate the function $f(x)$ at the points $(x_i, y_i) = (x_i, f(x_i)), i = 0, \ldots, n$ if we can find specific values of a_j such that,

$$g(x_i) = \sum_{j=0}^{n} a_j \phi_j(x_i) = f(x_i) = y_i$$

We can solve for the coefficients a_j in,

$$g(x_i) = \sum_{j=0}^{n} a_j \phi_j(x_i) = f(x_i) = y_i$$

by considering the matrix form for the coefficients,

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \dots & \phi_n(x_n) \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Monomials

Obvious attempt: try picking the basis functions as $\phi_j(x) = x^j$.

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

So for each x_i we have

$$p(x_i) = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n = y_i$$

OR

$$\begin{aligned}
a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0 \\
a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1 \\
a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_n x_2^n &= y_2 \\
a_0 + a_1 x_3 + a_2 x_3^2 + \cdots + a_n x_3^n &= y_3 \\
&\vdots \\
a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n
\end{aligned}$$

Monomial: The Vandermonde matrix

so that the matrix form for the coefficients is,

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Question

- Is this a “good” system to solve?

Example(Python)

from Recktenwald

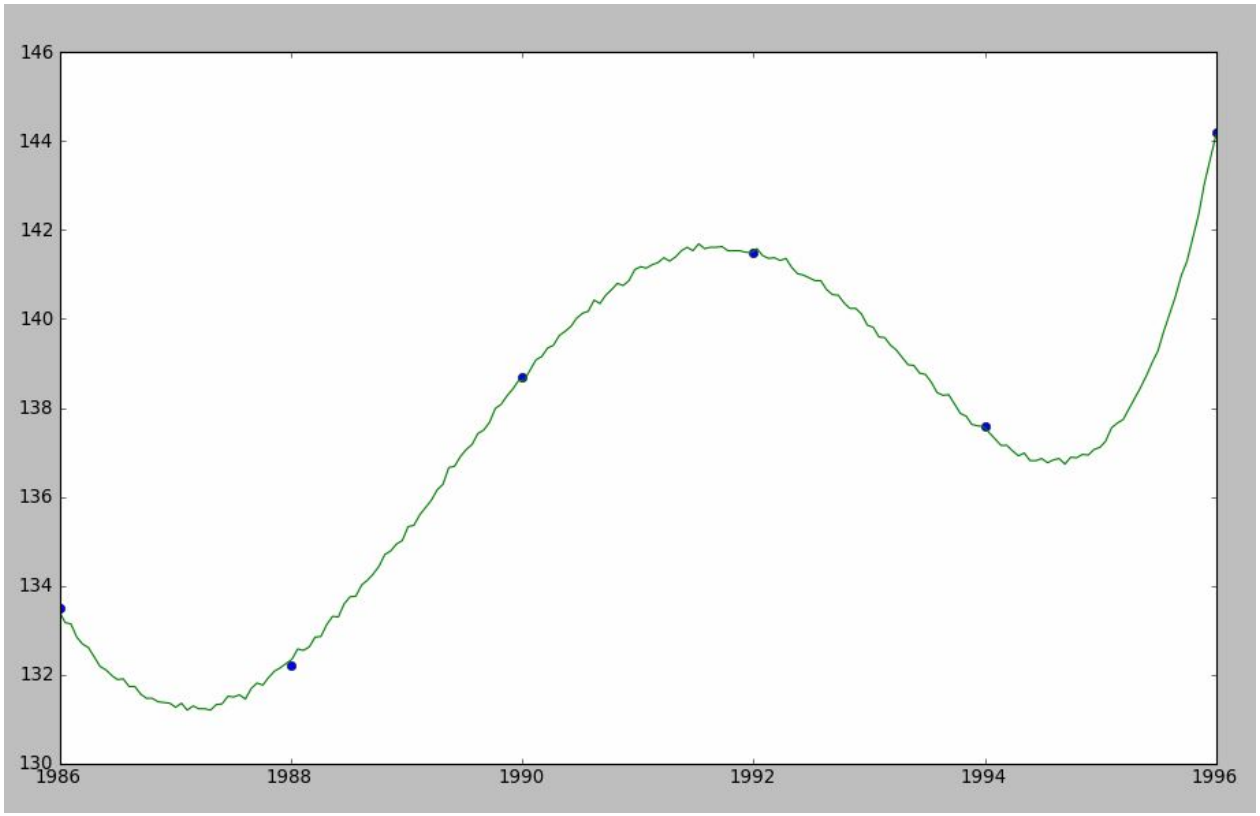
Consider Gas prices (in cents) for the following years:

x	year	1986	1988	1990	1992	1994	1996
y	price	133.5	132.2	138.7	141.5	137.6	144.2

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> array = np.array([1986,1988,1990, 1992, 1994, 1996])
>>> year = np.array([1986,1988,1990, 1992, 1994, 1996])
>>> price = np.array([133.5, 132.2, 138.7, 141.5, 137.6,
    144.2])
>>> M = np.vander(year)
 a = np.linalg.solve(M,price)
>>> a
array([  3.50333862e-03,  -3.48385098e+01,   1.38578605e+05,
        -2.75614447e+08,   2.74079747e+11,  -1.09021229e+14])
>>> x = np.linspace(1986,1996,200)
>>> p = np.polyval(a,x)
>>> plt.plot(year, price,'o',x,p,'-')
```

Example(Python)

from Recktenwald



Example (MATLAB)

from Recktenwald

Consider Gas prices (in cents) for the following years:

x	year	1986	1988	1990	1992	1994	1996
y	price	133.5	132.2	138.7	141.5	137.6	144.2

```
1  year =[1986   1988   1990   1992   1994   1996 ]';
2
3  price=[133.5 132.2  138.7  141.5  137.6  144.2]';
4
5  M = vander(year);
6  a = M\price;
7
8  x=linspace(1986,1996,200);
9  p=polyval(a,x);
10 plot(year,price,'o',x,p,'-');
```

Interpolation error

In what sense is the approximation a good one?

1. Interpolation: $g(x)$ (and/or its derivatives) must have the same values of $f(x)$ (and/or its derivatives) at set of given points.
2. Least-squares: $g(x)$ must deviate as little as possible from $f(x)$ in the sense of a 2-norm: minimize $\|f-g\|_2^2 = \int_a^b |f(t)-g(t)|^2\,dt$
3. Chebyshev: $g(x)$ must deviate as little as possible from $f(x)$ in the sense of the ∞-norm: minimize $\max_{t\in[a,b]} |f(t)-g(t)|$.

Interpolating polynomial is unique!

Given $n+1$ distinct points $x_0, \ldots, x_n$, and values $y_0, \ldots, y_n$, find a polynomial $p(x)$ of degree at most n so that

$$p(x_i) = y_i \quad i = 0, \ldots, n$$

- A polynomial of degree n has $n+1$ degrees-of-freedom:

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$

- $n+1$ constraints determine the polynomial uniquely:

$$p(x_i) = y_i, \quad i = 0, \ldots, n$$

Theorem

If points $x_0, \ldots, x_n$ are distinct, then for arbitrary $y_0, \ldots, y_n$, there is a *unique* polynomial $p(x)$ of degree at most n such that $p(x_i) = y_i$ for $i = 0, \ldots, n$. This unique polynomial is the minimal degree polynomial where $p(x_i) = y_i$ for $i = 0, \ldots, n$.

Fundamental Theorem of Algebra

How can you prove the interpolating polynomial is unique, so that we can speak of the interpolating polynomial? Assume that it isn't, then apply the following form of the Fundamental Theorem of Algebra.

Theorem

Every non-zero polynomial has exactly as many complex roots as its degree, where each root is counted up to its multiplicity.

Interpolation error using the unique interpolating polynomial

Theorem

Given function f *with* $n+1$ *continuous derivatives in the interval formed by* $I = [min(\{x, x_0, \ldots, x_n\}), max(\{x, x_0, \ldots, x_n\})]$. *If* $p(x)$ *is the unique interpolating polynomial of degree* $\leqslant n$ *with,*

$$p(x_i) = f(x_i), \quad i = 0, 1, \ldots, n$$

then the error is computed by the formula,

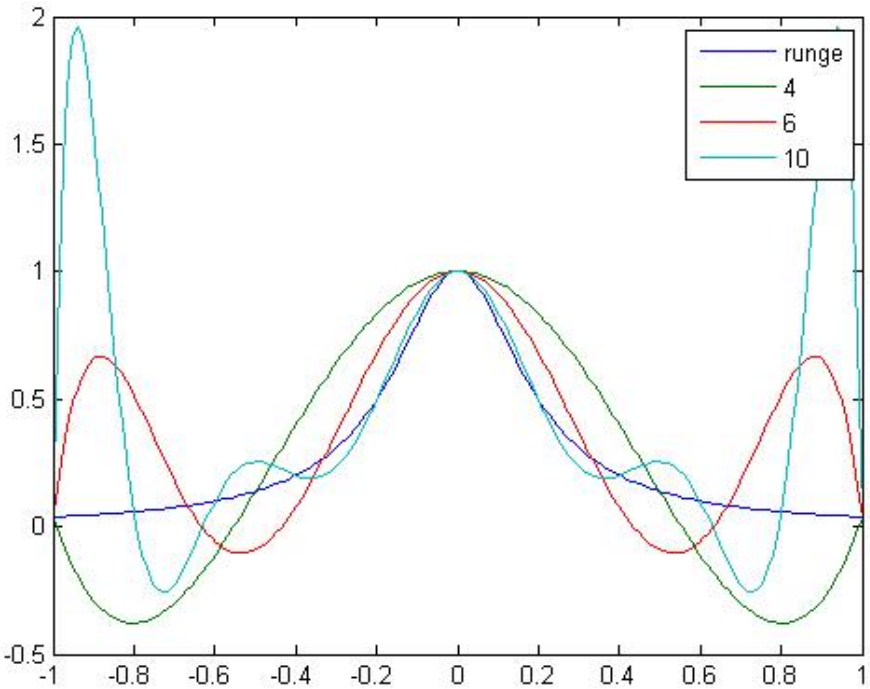
$$p(x) - f(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n), \qquad \text{for some } \xi(x) \in I$$

Interpolating Runge's function

If we interpolate Runge's function,

$$f(x) = \frac{1}{1 + 25x^2}$$

on the interval $[-1, 1]$ with equally spaced points we get the following results:



Higher degree interpolating polynomials can be problematic.

```
function runge()
  close all
% plot runge's function
   x2 = linspace(-1,1,200);
   y2 = 1./(1+25*x2.^2);
   plot(x2,y2);
   hold all
% plot using interp polys
   for i = [5  7  11]
     x = linspace(-1,1,i);
     y = 1./(1+25*x.^2);
     p = polyfit(x,y,i-1);
%
     x2 = linspace(-1,1,200);
     y2 = polyval(p,x2);
     plot(x2,y2);
     pause
   end
   legend('runge','4', '6', '10')
```

Monomial Basis

The Matlab function *polyfit* uses QR factorization to compute the coefficients for the interpolating polynomial, with a cost of $O(n^3)$ for an n-th degree polynomial. To evaluate the interpolating polynomial we can use Horner's method which has a cost of $O(n)$.

Horner's method

The polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

can be efficiently computed as,

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_n x) \ldots))$$

Polynomial Interpolation Strategy

- Lower Degree Polynomial Interpolation

Use piecewise polynomials (Splines)

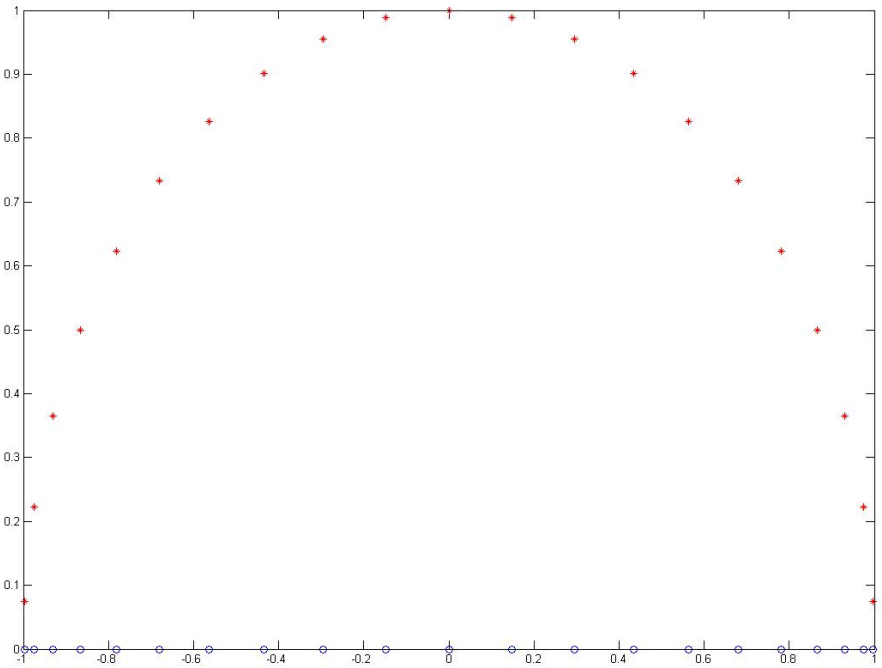
- Higher Degree Polynomial Interpolation

Use non-uniform spacing (Chebyshev)
Use basis functions that yield coefficients that are easier to compute (Lagrange or Newton)

Chebyshev Nodes

Chebyshev nodes in $[-1, 1]$

$$x_i = cos\left(\left(\frac{2i+1}{2}\right)\frac{\pi}{n+1}\right), \quad i = 0, \ldots, n$$



- Can obtain nodes from equidistant points on a circle projected down
- Nodes are non uniform and non nested

Chebyshev nodes

Remember that the interpolating error is given by,

$$p(x) - f(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_n), \quad \text{for some } \xi(x) \in I$$

We would like to choose the x_i values that minimizes

$$\max_{x \in I} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

If $I = [-1, 1]$ then the Chebyshev nodes produce the x_i that minimizes the above product and we have the following result,

$$\max_{x \in [-1,1]} \left| \prod_{i=0}^{n} (x - x_i) \right| = 2^{-n}$$

so that we can bound the interpolation error by the following formula,

$$|p(x) - f(x)| \leqslant \max_{\xi \in [-1,1]} \left| \frac{f^{(n+1)}(\xi)}{2^n (n+1)!} \right|$$

Lagrange polynomials

The general form for the Lagrange basis functions is

$$\ell_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

The resulting interpolating polynomial is

$$p(x) = \sum_{k=0}^{n} y_k \ell_k(x)$$

so the matrix form for the coefficients is,

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \ell_0(x_0) & \ell_1(x_0) & \dots & \ell_n(x_0) \\ \ell_0(x_1) & \ell_1(x_1) & \dots & \ell_n(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ \ell_0(x_n) & \ell_1(x_n) & \dots & \ell_n(x_n) \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Back to the basics...

Example

Find the interpolating polynomial of least degree that interpolates

x	1.4	1.25
y	3.7	3.9

Directly

$$\begin{aligned} p_1(x) &= \left(\frac{x-1.25}{1.4-1.25}\right)3.7 + \left(\frac{x-1.4}{1.25-1.4}\right)3.9 \\ &= 3.7 + \left(\frac{3.9-3.7}{1.25-1.4}\right)(x-1.4) \\ &= 3.7 - \frac{4}{3}(x-1.4) \end{aligned}$$

Lagrange

What have we done? We've written $p(x)$ as

$$p(x) = y_0 \left(\frac{x - x_1}{x_0 - x_1} \right) + y_1 \left(\frac{x - x_0}{x_1 - x_0} \right)$$

- the sum of two linear polynomials
- the first is zero at x_1 and 1 at x_0
- the second is zero at x_0 and 1 at x_1
- these are the two linear Lagrange basis functions:

$$\ell_0(x) = \frac{x - x_1}{x_0 - x_1} \qquad \ell_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Lagrange

Example

Write the Lagrange basis functions for

x	$\frac{1}{3}$	$\frac{1}{4}$	1
y	2	-1	7

Directly

$$\ell_0(x) = \frac{(x-\frac{1}{4})(x-1)}{(\frac{1}{3}-\frac{1}{4})(\frac{1}{3}-1)}$$

$$\ell_1(x) = \frac{(x-\frac{1}{3})(x-1)}{(\frac{1}{4}-\frac{1}{3})(\frac{1}{4}-1)}$$

$$\ell_2(x) = \frac{(x-\frac{1}{3})(x-\frac{1}{4})}{(1-\frac{1}{3})(1-\frac{1}{4})}$$

Example

Find the equation of the parabola passing through the points (1,6), (-1,0), and (2,12)

$x_0 = 1, x_1 = -1, x_2 = 2; \qquad y_0 = 6, y_1 = 0, y_2 = 12;$

$$\begin{aligned}
\ell_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x+1)(x-2)}{(2)(-1)} \\
\ell_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-1)(x-2)}{(-2)(-3)} \\
\ell_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-1)(x+1)}{(1)(3)}
\end{aligned}$$

$$\begin{aligned}
p_2(x) &= y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) \\
&= -3 \times (x+1)(x-2) + 0 \times \frac{1}{6}(x-1)(x-2) \\
&\quad +4 \times (x-1)(x+1) \\
&= (x+1)[4(x-1) - 3(x-2)] \\
&= (x+1)(x+2)
\end{aligned}$$

Summary so far:

- Monomials: $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ results in poor conditioning
- Monomials: but evaluating the Monomial interpolant is cheap (nested iteration)
- Lagrange: $p(x) = y_0 \ell_0(x) + \cdots + y_n \ell_n(x)$ is very well behaved.
- Lagrange: but evaluating the Lagrange interpolant can be expensive (each basis function is of the same order and the interpolant is not easily reduced to nested form). However we can use the Barycentric form of Lagrange interpolation to bound the evaluations to $O(n)$ for a polynomial of degree n.

Improving evaluation of Lagrange polynomials

If we denote,

$$\ell(x) = (x - x_0)(x - x_1)\dots(x - x_n)$$

then we can write the Lagrange basis functions as,

$$\ell_k(x) = \frac{\ell(x)}{(x - x_k)} \frac{1}{\prod_{i=0, i\neq k}^{n}(x_k - x_i)}$$

and thus if we compute,

$$w_k = \frac{1}{\prod_{i=0, i\neq k}^{n}(x_k - x_i)}$$

we can write the Lagrange interpolating polynomial as,

$$p(x) = \ell(x) \sum_{k=0}^{n} \frac{w_k}{x - x_k} y_k$$

If we pre-compute and store the w_k (for a cost of $O(n^2)$) then computing $p(x)$ is reduced to a cost of $O(n)$.

Newton Polynomials

- Newton Polynomials are of the form

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

- The basis used is thus

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

- More stable than monomials
- As computationally efficient (nested iteration) as Barycentric Lagrange interpolation

Newton Polynomials using Divided Differences

Consider the data

x_0	x_1	x_2
y_0	y_1	y_2

We want to find a_0, a_1, and a_2 in the following polynomial so that it fits the data:

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

Matching the data gives three equations to determine our three unknowns a_i:

$$\begin{aligned}
&\text{at } x_0\text{: } y_0 = a_0 + 0 + 0 \\
&\text{at } x_1\text{: } y_1 = a_0 + a_1(x_1 - x_0) + 0 \\
&\text{at } x_2\text{: } y_2 = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1)
\end{aligned}$$

Newton Polynomials using Divided Differences

Or in matrix form:

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

$\Rightarrow$ lower triangular
$\Rightarrow$ only $\mathcal{O}(n^2)$ operations

Question

How many operations are needed to find the coefficients in the monomial basis?

Newton Polynomials using Divided Differences

Using Forward Substitution to solve this lower triangular system yields:

$$
\begin{aligned}
a_0 &= y_0 = f(x_0) \\
a_1 &= \frac{y_1 - a_0}{x_1 - x_0} \\
&= \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\
a_2 &= \frac{y_2 - a_0 - (x_2 - x_0)a_1}{(x_2 - x_1)(x_2 - x_0)} \\
&= \ldots \text{ next slide}
\end{aligned}
$$

Newton Polynomials using Divided Differences

From the previous slide . . .

$$
\begin{aligned}
a_2 &= \frac{f(x_2) - f(x_0) - (x_2 - x_0)\frac{f(x_1)-f(x_0)}{x_1-x_0}}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{f(x_2) - f(x_1) + f(x_1) - f(x_0) - (x_2 - x_0)\frac{f(x_1)-f(x_0)}{x_1-x_0}}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{f(x_2) - f(x_1) + (f(x_1) - f(x_0))\left(1 - \frac{x_2-x_0}{x_1-x_0}\right)}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{f(x_2) - f(x_1) + (f(x_1) - f(x_0))\left(\frac{x_1-x_2}{x_1-x_0}\right)}{(x_2 - x_1)(x_2 - x_0)} \\
&= \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0}
\end{aligned}
$$

Newton Polynomials using Divided Differences

From this we see a pattern. There are many terms of the form

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

These are called *divided differences* and are denoted with square brackets:

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

Applying this to our results:

$$\begin{aligned} a_0 &= f[x_0] \\ a_1 &= f[x_0, x_1] \\ a_2 &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ &= f[x_0, x_1, x_2] \end{aligned}$$

Newton Polynomials using Divided Differences

We can now write the interpolating polynomial as,

$$p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n](x - x_0) \ldots (x - x_{n-1})$$

Newton Polynomials using Divided Differences

example: long way

Example

For the data

x	1	-4	0
y	3	13	-23

Find the 2nd order interpolating polynomial using Newton.

We know

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

And that

$$\begin{aligned}
a_0 &= f[x_0] = f[1] = f(1) = 3 \\
a_1 &= f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{13 - 3}{-4 - 1} = -2 \\
a_2 &= f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= \frac{\frac{-23-13}{0--4} - \frac{13-3}{-4-1}}{0 - 1} \\
&= \frac{-9 + 2}{-1} = 7
\end{aligned}$$

So

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

Divided Differences

Recursive Property

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

With the first two defined by

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

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

Divided Differences

Invariance Theorem

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

Simple "proof": $f[x_0, x_1, ..., x_k]$ is the coefficient of the x^k term in the polynomial interpolating f at $x_0, \ldots, x_k$. But any permutation of the x_i still gives the same polynomial.

Divided Differences

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: tables

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

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

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

Construct the divided differences table for the data

x	1	$\frac{3}{2}$	0	2
y	3	$\frac{13}{4}$	3	$\frac{5}{3}$

and construct the largest order interpolating polynomial.
We can compute the divided differences much more easily using tables. To construct the divided difference table for $f(x)$ for the $x_0, \ldots, x_3$

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

Divided Differences

the easy way: example

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

The coefficients are readily available and we arrive at

$$p_3(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)(x-\frac{3}{2}) - 2(x-1)(x-\frac{3}{2})x$$