Polynomial Interpolation

Jiwen He

Department of Mathematics, University of Houston

jiwenhe@math.uh.edu
math.uh.edu/∼jiwenhe/math6371

Monomial interpolation: Data (1, 1), (2, 3), (4, 3)

Monomial interpolation:

$$p_2(x) = c_0 + c_1 x + c_2 x^2$$

The interpolating conditions in matrix form read

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 4 & 16 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}$$

The MATLAB commands:

$$A = [1\,1\,1; 1\,2\,4; 1\,4\,16]; y = [1; 3; 3]; c = A\backslash y;$$

yields

$$c_0 = -7/3, c_1 = 4, c_2 = -2/3.$$

Lagrange interpolation: Data (1, 1), (2, 3), (4, 3)

Lagrange interpolation:

$$p_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x)$$

where $y_0 = 1$, $y_1 = 3$, $y_2 = 3$ and

$$L_0(x) = \frac{1}{3}(x-2)(x-4), L_1(x) = -\frac{1}{2}(x-1)(x-4), L_2(x) = \frac{1}{6}(x-1)(x-2)$$

Despite the different form, this is precisely the same quadratic interpolant as the one we found before

$$p_2(x) = (-2x^2 + 12x - 7)/3.$$

Newton interpolation:

$$p_2(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

where $x_0 = 1$, $x_1 = 2$, $x_2 = 4$, $f[x_0] = 1$, $f[x_1] = 3$, $f[x_2] = 3$, and

$$f[x_0, x_1] = \frac{3-1}{2-1} = 2, f[x_1, x_2] = \frac{3-3}{4-2} = 0, f[x_0, x_1, x_2] = \frac{0-2}{4-1} = -\frac{2}{3},$$

yielding

i	x_i	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$
0	1	1		
1	2	3	2	
2	4	3	0	$-\frac{2}{3}$

$$\begin{aligned} p_2(x) &= 1 + 2(x-1) - \frac{2}{3}(x-1)(x-2) \\ &= 1 + (x-1)\left(2 - \frac{2}{3}(x-2)\right) \end{aligned}$$

Newton interpolation: Additional Data Point (5, 4)

Note that we need only add another row to the divided difference table:

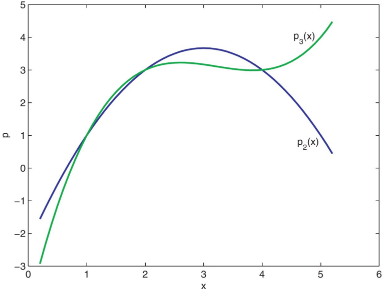
i	x_i	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$
0	1	1		
1	2	3	2	
2	4	3	0	$-\frac{2}{3}$

i	x_i	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
0	1	1			
1	2	3	2		
2	4	3	0	$-\frac{2}{3}$	
3	5	4	1	$\frac{1}{3}$	$\frac{1}{4}$

For p_3 we have the expression

$$
\begin{aligned}
p_3(x) &= p_2(x) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\
&= 1 + (x-1)\left(2 - \frac{2}{3}(x-2)\right) + \frac{1}{4}(x-1)(x-2)(x-4) \\
&= 1 + (x-1)\left(2 + (x-2)\left(-\frac{2}{3} + \frac{1}{4}(x-4)\right)\right).
\end{aligned}
$$

Newton interpolation: Additional Data Point



- Obtaining a higher degree approximation is simply a matter of adding a term.

$$p_3(x) = p_2(x) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2)$$

- Note that p_2 predicts a rather different value for $x = 5$ than the additional datum $f(x_3)$ later imposes, which explains the significant difference between the two interpolating curves.

Interpolating also Derivative Values

Give the five data values

t_i	$f(t_i)$	$f'(t_i)$	$f''(t_i)$
8.3	17.564921	3.116256	0.120482
8.6	18.505155	3.151762	

Set up the divided difference table

$$(x_0,x_1,x_2,x_3,x_4) = \left(\underbrace{8.3,8.3,8.3}_{m_0=2},\underbrace{8.6,8.6}_{m_1=1}\right),$$

$$f[x_0,x_1] = \frac{f'(t_0)}{1!} = f'(8.3), \quad f[x_1,x_2] = \frac{f'(t_0)}{1!},$$

$$f[x_0,x_1,x_2] = \frac{f''(t_0)}{2!} = \frac{f'(8.3)}{2}, \quad f[x_3,x_4] = \frac{f'(t_1)}{1!} = f'(8.6),$$

x_i	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot,\cdot]$
8.3	17.564921				
8.3	17.564921	3.116256			
8.3	17.564921	3.116256	0.060241		
8.6	18.505155	3.130780	0.048413	-0.039426	
8.6	18.505155	3.151762	0.069400	0.071756	0.370604

The resulting quartic interpolant is

$$p_4(x) = \sum_{k=0}^{4} f[x_0,\ldots,x_k]\prod_{j=0}^{k-1}(x-x_j)$$
$$= 17.564921 + 3.116256(x-8.3) + 0.060241(x-8.3)^2$$
$$- 0.039426(x-8.3)^3 + 0.370604(x-8.3)^3(x-8.6).$$

Algorithm: Lagrange Interpolation

Algorithm: Lagrange Polynomial Interpolation.

1. *Construction*: Given data $\{(x_i, y_i)\}_{i=0}^{n}$, compute barycentric weights $w_j = 1/\prod_{i \neq j}(x_j - x_i)$, and also the quantities $w_j y_j$, for $j = 0, 1, \ldots n$.
2. *Evaluation*: Given an evaluation point x not equal to one of the data points $\{x_i\}_{i=0}^{n}$, compute

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j y_j}{(x - x_j)}}{\sum_{j=0}^{n} \frac{w_j}{(x - x_j)}}.$$

Algorithm: Newton Interpolation

Algorithm: Polynomial Interpolation in Newton Form.

1. *Construction*: Given data $\{(x_i, y_i)\}_{i=0}^{n}$, where the abscissae are not necessarily distinct,

$$
\begin{aligned}
&\text{for } j = 0, 1, \ldots, n \\
&\quad \text{for } l = 0, 1, \ldots, j \\
&\qquad \gamma_{j,l} = \begin{cases} \dfrac{\gamma_{j,l-1} - \gamma_{j-1,l-1}}{x_j - x_{j-l}} & \text{if } x_j \neq x_{j-l}, \\[2ex] \dfrac{f^{(l)}(x_j)}{l!} & \text{otherwise.} \end{cases}
\end{aligned}
$$

2. *Evaluation*: Given an evaluation point x,

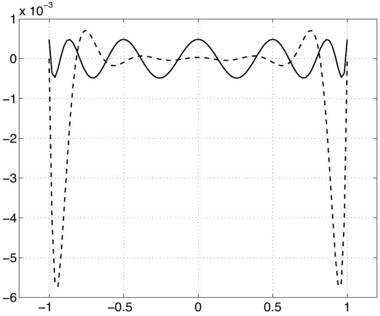
$$
\begin{aligned}
&p = \gamma_{n,n} \\
&\text{for } j = n-1, n-2, \ldots, 0, \\
&\quad p = p\,(x - x_j) + \gamma_{j,j}
\end{aligned}
$$

Algorithms Comparison

Basis name	$\phi_j(x)$	Construction cost	Evaluation cost	Selling feature
Monomial	x^j	$\frac{2}{3}n^3$	$2n$	simple
Lagrange	$L_j(x)$	n^2	$5n$	$c_j = y_j$ most stable
Newton	$\prod_{i=0}^{j-1}(x - x_i)$	$\frac{3}{2}n^2$	$2n$	adaptive

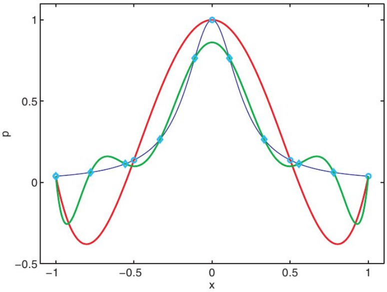
Runge's Phenomenon: Error of Interpolation

$f(x) = x^{n+1}$ on $[-1, 1]$, $n = 11$

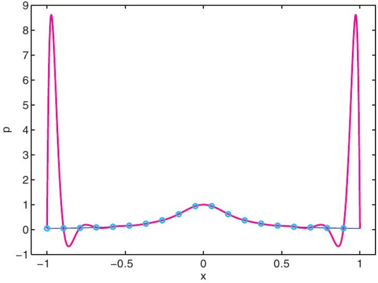


- Using $n + 1 = 12$ Chebyshev points (solid line) and equidistant points (dashed line)
- The interpolation error for $f(x) = x^{n+1}$ becomes $f(x) - p_n(x) = \psi_n(x)$, because $\frac{f^{(n+1)}}{(n+1)!} = 1$
- Equidistant interpolation can give rise to convergence difficulties when the number of interpolation points becomes large.

Runge Function: $f(x) = \frac{1}{1+25x^2}$ with Equidistant Points



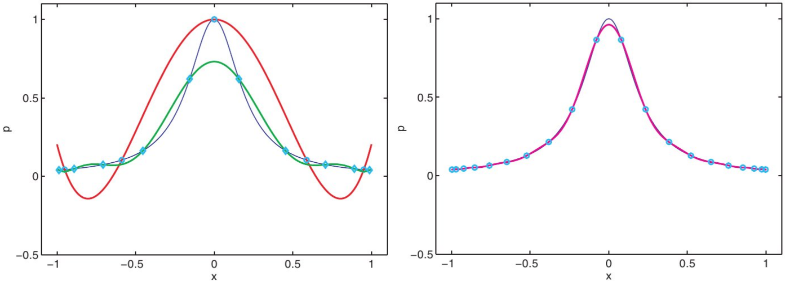
(a) $n = 4, 9$.



(b) $n = 19$.

- Calculating $\frac{f^{(n+1)}}{(n+1)!}$ shows the growth in the error term near the interval ends
- The results do not improve as the degree of the polynomial is increased.

Runge Function: $f(x) = \frac{1}{1+25x^2}$ with Chebyshev Points

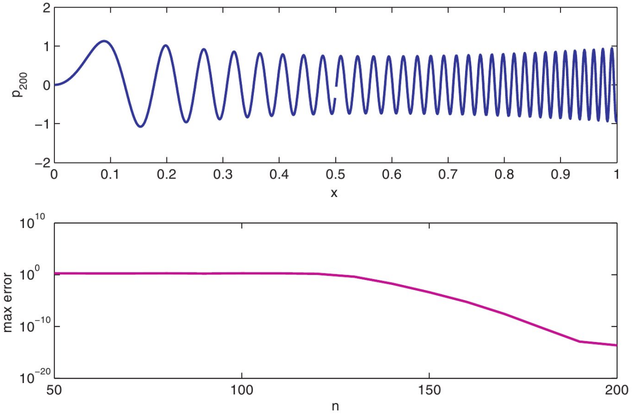


(a) $n = 4, 9$.

(b) $n = 19$.

- The improvement over the interpolation at equidistant points is remarkable!
- The Chebyshev points concentrate more near the interval ends, which is precisely where $\frac{f^{(n+1)}}{(n+1)!}$ gets large.

$f(x) = e^{3x}\sin(200x^2)/(1+20x^2)$ with Chebyshev Points



- As n is increased further the error eventually goes down, and rather fast: the error then looks like $O(q^{-n})$ for some $q > 1$. This is called spectral accuracy.