2021 Spring MAS 365
Chapter 3: Interpolation and Polynomial Approximation

Donghwan Kim

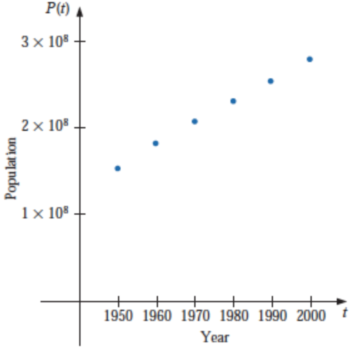
KAIST

Apr/13,15,27,29, May/4, 2021

Introduction

- A census of the population of the US is taken every 10 years.

Year	1950	1960	1970	1980	1990	2000
Population (in thousands)	151,326	179,323	203,302	226,542	249,633	281,422



Q. Can we reasonably estimate the population in 1975 or 2020?

Weierstrass Approximation Theorem

- Consider the algebraic polynomials, the set of functions of the form

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

where n is a nonnegative integer and $a_0, \ldots, a_n$ are real constants.

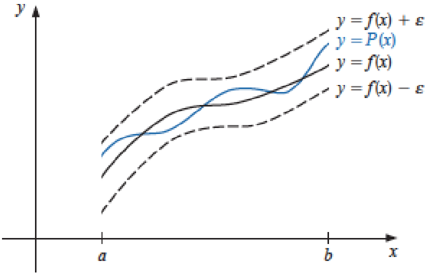
Q. Why polynomials in interpolation?

Weierstrass Approximation Theorem (cont'd)

Theorem 1 (Weierstrass Approximation Theorem)

Suppose that f is defined and continuous on $[a, b]$. For each $\epsilon > 0$, there exists a polynomial $P(x)$, with the property that

$$|f(x) - P(x)| < \epsilon, \quad \text{for all } x \text{ in } [a, b].$$



Q. How can we construct such polynomial?

Taylor Polynomials for Interpolation

Q. How about Taylor polynomials?

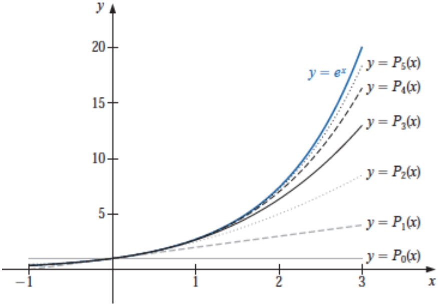
Taylor Polynomials for Interpolation (cont'd)

Ex. Consider Taylor polynomials about $x_0 = 0$ for $f(x) = e^x$

- $P_0(x) = 1$
- $P_1(x) = 1 + x$

$$\vdots$$

- $P_5(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}$



- The error becomes progressively worse as we move away from zero...
- The approximation improves with higher-degree, but not in general?

Taylor Polynomials for Interpolation (cont'd)

Ex. Taylor polynomials about $x_0 = 1$ for $f(x) = 1/x$ to approximate $f(3)$.

- Since $f^{(k)}(x) = (-1)^k k! x^{-k-1}$, the Taylor polynomials are

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(1)}{k!}(x-1)^k = \sum_{k=0}^{n}(-1)^k(x-1)^k.$$

- $P_n(3)$ for larger n provides more inaccurate approximations.

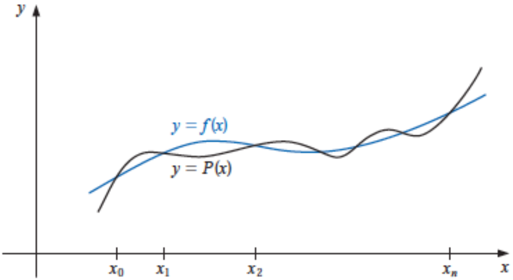
n	0	1	2	3	4	5	6	7
$P_n(3)$	1	−1	3	−5	11	−21	43	−85

- All the information used in the approximation by Taylor polynomials is concentrated at the single number x_0.

Lagrange Interpolating Polynomials

Q. How can we construct a polynomial of degree at most n that passes through the $n+1$ points?

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)).$$



Lagrange Interpolating Polynomials (cont'd)

Q. How can we construct a polynomial of degree one that passes through the distinct points (x_0, y_0) and (x_1, y_1)?

Q. How can we approximate a function f for which $f(x_0) = y_0$ and $f(x_1) = y_1$ by a first-order polynomial?

Lagrange Interpolating Polynomials (cont'd)

- The linear **Lagrange interpolating polynomial** through (x_0, y_0) and (x_1, y_1) is

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

where

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{and} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}.$$

- Note that $L_0(x_0) = 1$, $L_1(x_0) = 0$, and $L_0(x_1) = 0$, $L_1(x_1) = 1$, so $P(x_0) = y_0$ and $P(x_1) = y_1$.

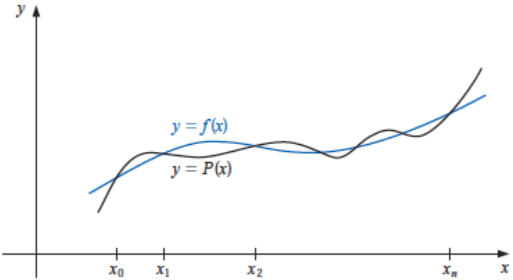
Lagrange Interpolating Polynomials (cont'd)

Ex. Determine the linear Lagrange interpolating polynomial that passes through the points $(x_0, y_0) = (2, 4)$ and $(x_1, y_1) = (5, 1)$.

Lagrange Interpolating Polynomials (cont'd)

- Let's construct a polynomial of degree at most n that passes through the $n+1$ points?

$$(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)).$$



Lagrange Interpolating Polynomials (cont'd)

- Let's first construct a function $L_{n,k}(x)$ for $k = 0, 1, \ldots, n$ that generalizes our previous choice $L_{1,0}(x)$ and $L_{1,1}(x)$.
  (We will omit n in $L_{n,k}$ when there is no confusion on its degree n.)

Q. Which property should we enforce?

The nth Lagrange Interpolating Polynomial

Theorem 2

If $x_0, x_1, \ldots, x_n$ are $n+1$ distinct numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$f(x_k) = P(x_k), \quad \text{for each } k = 0, 1, \ldots, n.$$

This polynomial is given by

$$P(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x),$$

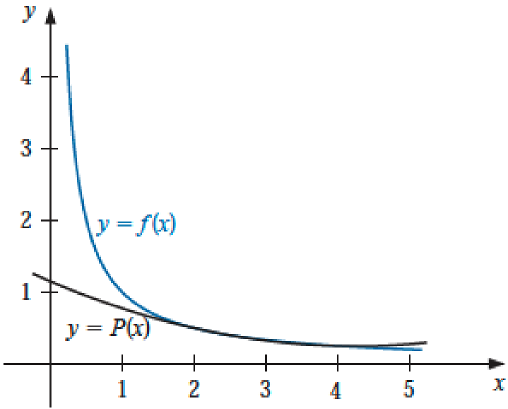
where for each $k = 0, 1, \ldots, n$,

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \\ &= \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)}. \end{aligned}$$

The nth Lagrange Interpolating Polynomial (cont'd)

Ex. Find the second Lagrange interpolating polynomial for $f(x) = 1/x$ given $x_0 = 2$, $x_1 = 2.75$, and $x_2 = 4$. And approximate $f(3) = 1/3$.

The nth Lagrange Interpolating Polynomial (cont'd)



Error Term for Lagrange Interpolating Polynomial

Theorem 3

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 $[a, b]$, a number $\xi(x)$ between the minimum and the maximum of $\{x_0, x_1, \ldots, x_n\}$, and hence in (a, b), exists with

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n)$$

where $P(x)$ is the nth Lagrange interpolating polynomial.

Proof. If $x = x_k$ for any $k = 0, 1, \ldots, n$, then $f(x_k) = P(x_k)$, and choosing $\xi(x_k)$ arbitrarily in (a, b) is enough.

If $x \neq x_k$, for all $k = 0, 1, \ldots, n$, define the function g for $t \in [a, b]$ by

$$g(t) := f(t) - P(t) - [f(x) - P(x))] \prod_{i=0}^{n} \frac{(t - x_i)}{(x - x_i)}.$$

Error Term for Lagrange Interpolating Poly. (cont'd)

Theorem 4 (Generalized Rolle's Theorem)

Suppose $f \in C[a, b]$ *is* n *times differentiable on* (a, b). *If* $f(x) = 0$ *at the* $n + 1$ *distinct numbers* $a \leq x_0 < x_1 < \ldots < x_n \leq b$, *then a number* c *in* (x_0, x_n), *and hence in* (a, b), *exists with* $f^{(n)}(c) = 0$.

Error Term for Lagrange Interpolating Poly. (cont'd)

Proof. Since $f \in C^{n+1}[a,b]$, and $P \in C^{\infty}[a,b]$, it follows that $g \in C^{n+1}[a,b]$. We also have $g(x_k) = 0$ for $k = 0, 1, \ldots, n$ and $g(x) = 0$.

By Generalized Rolle's Theorem, there exists a number ξ in (a,b) for which $g^{(n+1)}(\xi) = 0$, so

$$\begin{aligned} 0 &= g^{(n+1)}(\xi) \\ &= f^{(n+1)}(\xi) - P^{(n+1)}(\xi) - [f(x) - P(x)] \frac{d^{n+1}}{dt^{n+1}} \left[\prod_{i=0}^{n} \frac{(t - x_i)}{(x - x_i)} \right]_{t=\xi} \\ &= f^{(n+1)}(\xi) - 0 - [f(x) - P(x)] \frac{(n+1)!}{\prod_{i=0}^{n}(x - x_i)} \end{aligned}$$

□

Error Term for Lagrange Interpolating Poly. (cont'd)

- Error term of nth Taylor polynomial about x_0:

$$\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)^{n+1}$$

- Error term of nth Lagrange polynomial:

$$\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

Error Term for Lagrange Interpolating Poly. (cont'd)

Ex. Determine the error form and the maximum error (for $x \in [2, 4]$) for the second Lagrange polynomial for $f(x) = 1/x$ on $[2, 4]$ using the nodes $x_0 = 2$, $x_1 = 2.75$, and $x_2 = 4$.

Error Term for Lagrange Interpolating Poly. (cont'd)

Ex. Consider preparing a table for $f(x) = e^x$, for $x \in [0, 1]$. Assume that the difference between adjacent x-values, the step size, is h. Which h will ensure that linear interpolation gives an absolute error of at most 10^{-6} for all x in $[0, 1]$?

Error Term for Lagrange Interpolating Poly. (cont'd)

Q. Will a sequence of nth Lagrange polynomial $\{P_n\}$ converge to a continuous function f as $n \to \infty$?

Error Term for Lagrange Interpolating Poly. (cont'd)

- If the following satisfies

$$\lim_{n\to\infty} \frac{\max_{\xi\in(a,b)} |f^{(n+1)}(\xi)|}{(n+1)!} \max_{x\in[a,b]} |(x-x_0)(x-x_1)\cdots(x-x_n)| = 0,$$

we have

$$\lim_{n\to\infty} \max_{x\in[a,b]} |f(x) - P_n(x)| = 0.$$

Runge Phenomenon

- Consider the sequence of the Lagrange interpolating polynomials $\{P_n(x)\}$ with equally spaced points on $[-5, 5]$

$$x_i = -5 + \frac{10i}{n}, \quad i = 0, \dots, n$$

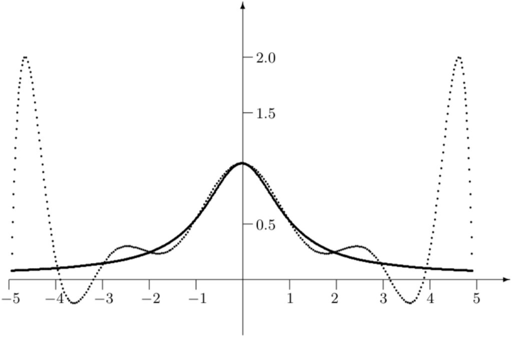
to

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5].$$

Runge Phenomenon (cont'd)

- Table: $\max_{x\in[-5,5]} |f(x) - P_n(x)|$
- Solid line: $f(x) = \frac{1}{1+x^2}$, Dotted line: $P_{10}(x)$

Degree n	Max error
2	0.65
4	0.44
6	0.61
8	1.04
10	1.92
12	3.66
14	7.15
16	14.25
18	28.74
20	58.59
22	121.02
24	252.78



Q. Can we do better?

Data Approximation

- Lagrange polynomials are frequently used when interpolating tabulated data.
- In such case, only the values of the polynomial at specified points are needed (rather than an explicit form of the polynomial).
- Also, the function underlying the data might be unknown, so the explicit form of the error cannot be used.

Data Approximation (cont'd)

Ex. Given values of a function f at various points, provide approximations to $f(1.5)$ by various Lagrange polynomials.

x	$f(x)$
1.0	0.7651977
1.3	0.6200860
1.6	0.4554022
1.9	0.2818186
2.2	0.1103623

Data Approximation (cont'd)

- The most appropriate linear polynomial uses $x_0 = 1.3$ and $x_1 = 1.6$, which gives

$$P_1(1.5) = \frac{1.5 - 1.6}{1.3 - 1.6} f(1.3) + \frac{1.5 - 1.3}{1.6 - 1.3} f(1.6) = 0.5102968.$$

Data Approximation (cont'd)

- Two polynomials of degree 2 can reasonably used, one with $x_0 = 1.3$, $x_1 = 1.6$ and $x_2 = 1.9$:

$$P_2(1.5) = \frac{(1.5-1.6)(1.5-1.9)}{(1.3-1.6)(1.3-1.9)} f(1.3) + \frac{(1.5-1.3)(1.5-1.9)}{(1.6-1.3)(1.6-1.9)} f(1.6) + \frac{(1.5-1.3)(1.5-1.6)}{(1.9-1.3)(1.9-1.6)} f(1.9) = 0.5112857,$$

  and one with $x_0 = 1.0$, $x_1 = 1.3$, $x_2 = 1.6$ yielding $\hat{P}_2(1.5) = 0.5124715$.
- Third-degree case:

  $x_0 = 1.3$, $x_1 = 1.6$, $x_2 = 1.9$, $x_3 = 2.2$ gives $P_3(1.5) = 0.5118302$.

  $x_0 = 1.0$, $x_1 = 1.3$, $x_2 = 1.6$, $x_3 = 1.9$ gives $\hat{P}_3(1.5) = 0.5118127$.
- Fourth-degree case: $x_0 = 1.0$, $x_1 = 1.3$, $x_2 = 1.6$, $x_3 = 1.9$, $x_4 = 2.2$ gives $P_4(1.5) = 0.5118200$.

Data Approximation (cont'd)

- Since $P_3(1.5) = 0.5118302$, $\hat{P}_3(1.5) = 0.5118127$, $P_4(1.5) = 0.5118200$ all agree to within 2×10^{-5}, we expect this degree of accuracy for these approximations.
- We also expect $P_4(1.5)$ to be the most accurate approximations, since it uses more data.

Data Approximation (cont'd)

- The function f we are approximating has $f(1.5) = 0.5118277$, so true accuracies are

$$
\begin{aligned}
|P_1(1.5) - f(1.5)| &\approx 1.53 \times 10^{-3}, \\
|P_2(1.5) - f(1.5)| &\approx 5.42 \times 10^{-4}, \\
|\hat{P}_2(1.5) - f(1.5)| &\approx 6.44 \times 10^{-5}, \\
|P_3(1.5) - f(1.5)| &\approx 2.5 \times 10^{-6}, \\
|\hat{P}_3(1.5) - f(1.5)| &\approx 1.50 \times 10^{-5}, \\
|P_4(1.5) - f(1.5)| &\approx 7.7 \times 10^{-6}.
\end{aligned}
$$

- We have no knowledge of the fourth derivative of f, so the Lagrange error term cannot be computed.

Neville's Method

- Error term is difficult to apply, so the degree of the polynomial needed for the desired accuracy is generally not known.

- A common practice is to compute the results from various polynomials until appropriate agreement is obtained.

Q. Can we do this efficiently?

Neville's Method (cont'd)

Definition 1

Let f be a function defined at $x_0, x_1, x_2, \ldots, x_n$, and suppose that $m_1, m_2, \ldots, m_k$ are k distinct integers, with $0 \leq m_i \leq n$ for each i. The Lagrange polynomial that agrees with $f(x)$ at the k points $x_{m_1}, x_{m_2}, \ldots, x_{m_k}$ is denoted $P_{m_1,m_2,\ldots,m_k}(x)$.

Ex. Suppose that $x_0 = 1$, $x_1 = 2$, $x_2 = 3$, $x_3 = 4$, $x_4 = 6$, and $f(x) = e^x$. Determine the interpolating polynomial denoted $P_{1,2,4}(x)$, and use this polynomial to approximate $f(5)$.

Neville's Method (cont'd)

Theorem 5

Let f be defined at $x_0, x_1, \ldots, x_k$ and let x_j and x_i be two distinct numbers in this set. Then

$$P(x) = \frac{(x - x_j)P_{0,1,\ldots,j-1,j+1,\ldots,k}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,\ldots,k}(x)}{x_i - x_j}$$

is the kth Lagrange polynomial that interpolates f at the $k+1$ points $x_0, x_1, \ldots, x_k$ (i.e., $P(x) = P_{0,1,\ldots,k}(x)$).

Proof. For simplicity, let $Q_i \equiv P_{0,1,\ldots,i-1,i+1,\ldots,k}$ and $Q_j \equiv P_{0,1,\ldots,j-1,j+1,\ldots,k}$. Since $Q_i(x)$ and $Q_j(x)$ are polynomials of degree $k-1$ or less, $P(x)$ is of degree at most k.

Neville's Method (cont'd)

Proof. Using $Q_j(x_i) = f(x_i)$, we have

$$P(x_i) = \frac{(x_i - x_j)Q_j(x_i) - (x_i - x_i)Q_i(x_i)}{x_i - x_j} = f(x_i).$$

Similarly. since $Q_i(x_j) = f(x_j)$, we have $P(x_j) = f(x_j)$.

In addition, if $0 \le r \le k$ and r is neither i nor j, then $Q_i(x_r) = Q_j(x_r) = f(x_r)$. So

$$P(x_r) = \frac{(x_r - x_j)Q_j(x_r) - (x_r - x_i)Q_i(x_r)}{x_i - x_j} = f(x_r).$$

By definition, $P_{0,1,\ldots,k}(x)$ is the unique polynomial of degree at most k that agrees with f at $x_0, x_1, \ldots, x_k$, so $P \equiv P_{0,1,\ldots,k}$. $\square$

Neville's Method (cont'd)

Ex. Interpolating polynomials can be generated recursively, for example

$$P_{0,1} = \frac{(x - x_0)P_1 - (x - x_1)P_0}{x_1 - x_0}$$
$$P_{1,2} = \frac{(x - x_1)P_2 - (x - x_2)P_1}{x_2 - x_1}$$

and

$$P_{0,1,2} = \frac{(x - x_0)P_{1,2} - (x - x_2)P_{0,1}}{x_2 - x_0}$$

Neville's Method (cont'd)

- Neville's method: recursively generate interpolating polynomial approximations

x_0	P_0				
x_1	P_1	$P_{0,1}$			
x_2	P_2	$P_{1,2}$	$P_{0,1,2}$		
x_3	P_3	$P_{2,3}$	$P_{1,2,3}$	$P_{0,1,2,3}$	
x_4	P_4	$P_{3,4}$	$P_{2,3,4}$	$P_{1,2,3,4}$	$P_{0,1,2,3,4}$

Neville's Method (cont'd)

- The notation can be simplified by introducing

$$Q_{i,j} = P_{i-j,i-j+1,\ldots,i-1,i} \quad \text{for } 0 \leq j \leq i,$$

which denotes the interpolating polynomial of degree j on the $(j+1)$ numbers $x_{i-j}, x_{i-j+1}, \ldots, x_i$.

- Then,

$$Q_{i,j} = \frac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{(x_i - x_{i-j})}.$$

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

Neville's Method (cont'd)

Ex. The table lists the values of $f(x) = \ln x$ at given points. Use Neville's method and four-digit rounding arithmetic to approximate $f(2.1)$.

i	x_i	$\ln x_i$
0	2.0	0.6931
1	2.2	0.7885
2	2.3	0.8329

Sol. Using $x - x_0 = 0.1$, $x - x_1 = -0.1$, $x - x_2 = -0.2$ and $Q_{0,0} = 0.6931$, $Q_{1,0} = 0.7885$, $Q_{2,0} = 0.8329$, we have

$$Q_{1,1} = \frac{1}{0.2}[(0.1)0.7885 - (-0.1)0.6931] = \frac{0.1482}{0.2} = 0.7410$$
$$Q_{2,1} = \frac{1}{0.1}[(0.1)0.8329 - (-0.2)0.7885] = \frac{0.07441}{0.1} = 0.7441$$

and

$$Q_{2,2} = \frac{1}{0.3}[(0.1)0.7441 - (-0.2)0.7410] = \frac{0.2276}{0.3} = 0.7420.$$

Neville's Method (cont'd)

- We know $f(2.1) = \ln 2.1 = 0.7419$ to four decimal places, so the absolute error is

$$|f(2.1) - P_2(2.1)| = |0.7419 - 0.7420| = 10^{-4}.$$

- The Lagrange error formula gives

$$|f(2.1) - P_2(2.1)| = \left|\frac{f'''(\xi(2.1))}{3!}(x - x_0)(x - x_1)(x - x_2)\right|$$

Neville's Method vs. Divided Difference Method

- Neville's method: successively generates $P_n(x)$ for a fixed x.
- Divided difference method: successively generates the Lagrange polynomials themselves.

Alternate Representation of Lagrange Polynomial

- Although the Lagrange interpolating polynomial $P_n(x)$ is unique, there are alternate representations that are useful in certain situations, *e.g.*,

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

Q. How can we determine the constants $a_0, a_1, \ldots, a_n$?

Alternate Representation of Lagrange Polynomial (cont'd)

- Evaluating $P_n(x)$ at x_0 yields

$$a_0 = P_n(x_0) = f(x_0).$$

- Evaluating $P_n(x)$ at x_1 yields

$$f(x_0) + a_1(x_1 - x_0) = P_n(x_1) = f(x_1),$$

so

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

Divided Difference

- The zeroth divided difference of f with respect to x_i is defined as

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

- The first divided difference of f with respect to x_i and x_{i+1} is defined as

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

- The second divided difference is defined as

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

- Then kth divided difference relative to $x_i, x_{i+1}, \ldots, x_{i+k}$ is defined as

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

- This ends with the single nth divided difference.

Newton's Divided Difference

- We have that $a_k = f[x_0, x_1, \ldots, x_k]$, and thus $P_n(x)$ can be rewritten in a form called Newton's Divided Difference:

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0) \cdots (x - x_{k-1})$$

$f[x_0, x_1, \ldots, x_k]$ is independent of the order of $x_0, x_1, \ldots, x_k$.

Newton's Divided Difference (cont'd)

x	$f(x)$	First divided differences	Second divided differences	Third divided differences
x_0	$f[x_0]$			
		$f[x_0,x_1]=\dfrac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2]=\dfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$	
		$f[x_1,x_2]=\dfrac{f[x_2]-f[x_1]}{x_2-x_1}$		$f[x_0,x_1,x_2,x_3]=\dfrac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$
x_2	$f[x_2]$		$f[x_1,x_2,x_3]=\dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	
		$f[x_2,x_3]=\dfrac{f[x_3]-f[x_2]}{x_3-x_2}$		$f[x_1,x_2,x_3,x_4]=\dfrac{f[x_2,x_3,x_4]-f[x_1,x_2,x_3]}{x_4-x_1}$
x_3	$f[x_3]$		$f[x_2,x_3,x_4]=\dfrac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$	
		$f[x_3,x_4]=\dfrac{f[x_4]-f[x_3]}{x_4-x_3}$		$f[x_2,x_3,x_4,x_5]=\dfrac{f[x_3,x_4,x_5]-f[x_2,x_3,x_4]}{x_5-x_2}$
x_4	$f[x_4]$		$f[x_3,x_4,x_5]=\dfrac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}$	
		$f[x_4,x_5]=\dfrac{f[x_5]-f[x_4]}{x_5-x_4}$		
x_5	$f[x_5]$			

Generalized Mean Value Theorem

- By the Mean Value Theorem,

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

implies that when f' exists, $f[x_0, x_1] = f'(\xi)$ for some number ξ between x_0 and x_1.

Theorem 6 (Generalized Mean Value Theorem)

Suppose that $f \in C^n[a, b]$ and $x_0, x_1, \ldots, x_n$ are distinct numbers in $[a, b]$. Then a number ξ exists in (a, b) with

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

Q. What do we have here?

Generalized Mean Value Theorem

Proof Let $g(x) := f(x) - P_n(x)$.

Theorem 7 (Generalized Rolle's Theorem)

Suppose $f \in C[a, b]$ *is* n *times differentiable on* (a, b). *If* $f(x) = 0$ *at the* $n + 1$ *distinct numbers* $a \leq x_0 < x_1 < \cdots < x_n \leq b$, *then a number* c *in* (x_0, x_n) *and hence in* (a, b) *exists with* $f^{(n)}(c) = 0$.

Newton's Divided-Difference with Equal Spacing

- Consider the case when the nodes are equally spaced, *i.e.*,

$$h = x_{i+1} - x_i \quad \text{for each } i = 0, 1, \ldots, n-1.$$

- Let $x = x_0 + sh$, then $x - x_i = (s-i)h$, so

$$P_n(x) = P_n(x_0 + sh) = ?$$

Newton Forward-Difference Formula

- Newton forward-difference formula is constructed by using the forward difference notation Δ:

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

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

- In general,

$$f[x_0, x_1, \ldots, x_k] = \frac{1}{k!h^k}\Delta^k f(x_0)$$

and thus

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

Newton Backward-Difference Formula

- Consider the reordered $x_n, x_{n-1}, \ldots, x_0$, then we have

$$\begin{aligned} P_n(x) = {} & f[x_n] + f[x_n, x_{n-1}](x - x_n) \\ & + \cdots + f[x_n, \ldots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_1). \end{aligned}$$

- Let $x = x_n + sh$, then $x - x_i = (s + n - i)h$.
- By defining the backward difference ∇p_n (read *nabla* p_n) by

$$\nabla p_n = p_n - p_{n-1},$$

we have

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

and thus

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

Newton Difference: Forward and Backward

x	$f(x)$	First divided differences	Second divided differences	Third divided differences
x_0	$f[x_0]$			
		$f[x_0,x_1]=\dfrac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2]=\dfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$	
		$f[x_1,x_2]=\dfrac{f[x_2]-f[x_1]}{x_2-x_1}$		$f[x_0,x_1,x_2,x_3]=\dfrac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$
x_2	$f[x_2]$		$f[x_1,x_2,x_3]=\dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	
		$f[x_2,x_3]=\dfrac{f[x_3]-f[x_2]}{x_3-x_2}$		$f[x_1,x_2,x_3,x_4]=\dfrac{f[x_2,x_3,x_4]-f[x_1,x_2,x_3]}{x_4-x_1}$
x_3	$f[x_3]$		$f[x_2,x_3,x_4]=\dfrac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$	
		$f[x_3,x_4]=\dfrac{f[x_4]-f[x_3]}{x_4-x_3}$		$f[x_2,x_3,x_4,x_5]=\dfrac{f[x_3,x_4,x_5]-f[x_2,x_3,x_4]}{x_5-x_2}$
x_4	$f[x_4]$		$f[x_3,x_4,x_5]=\dfrac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}$	
		$f[x_4,x_5]=\dfrac{f[x_5]-f[x_4]}{x_5-x_4}$		
x_5	$f[x_5]$			

Q. Why do we care forward and backward?

Newton Difference: Forward and Backward (cont'd)

		First divided differences	Second divided differences	Third divided differences	Fourth divided differences
1.0	0.7651977				
		−0.4837057			
1.3	0.6200860		−0.1087339		
		−0.5489460		0.0658784	
1.6	0.4554022		−0.0494433		0.0018251
		−0.5786120		0.0680685	
1.9	0.2818186		0.0118183		
		−0.5715210			
2.2	0.1103623				

- Approximate $f(1.1)$: consider the earliest use of points closest to $x = 1.1$. Newton forward divided-difference uses those with a *solid* underline

$$P_4(1.1) = P_4(1.0 + \frac{1}{3}0.3) = 0.7651977 + \frac{1}{3}(0.3)(-0.4837057)$$
$$+ \cdots + \frac{1}{3}\left(-\frac{2}{3}\right)\left(-\frac{5}{3}\right)\left(-\frac{8}{3}\right)(0.3)^4(0.0018251)$$

- Approximate $f(2.0)$: Newton backward uses those with a *wavy* underline.

Centered Differences

- We noticed that Newton forward- and backward-difference formulas may not be appropriate for x near the center of the table.
- Stirling's formula (one of centered-difference formulas):
  If $n = m + 1$ is odd,

$$P_n(x) = P_{2m+1}(x) = f[x_0] + \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1])$$

$$+ \cdots + \frac{s(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2}(f[x_{-m-1}, \ldots, x_m] + f[x_{-m}, \ldots, x_{m+1}])$$

x	$f(x)$	First divided differences	Second divided differences	Third divided differences	Fourth divided differences
x_{-2}	$f[x_{-2}]$				
		$f[x_{-2}, x_{-1}]$			
x_{-1}	$f[x_{-1}]$		$f[x_{-2}, x_{-1}, x_0]$		
		$\underline{f[x_{-1}, x_0]}$		$\underline{f[x_{-2}, x_{-1}, x_0, x_1]}$	
x_0	$\underline{f[x_0]}$		$\underline{f[x_{-1}, x_0, x_1]}$		$\underline{f[x_{-2}, x_{-1}, x_0, x_1, x_2]}$
		$\underline{f[x_0, x_1]}$		$\underline{f[x_{-1}, x_0, x_1, x_2]}$	
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$			
x_2	$f[x_2]$				

Taylor and Lagrange Polynomials

- Taylor polynomials: degree at most m

$$\frac{d^k P(x_0)}{dx^k} = \frac{d^k f(x_0)}{dx^k}, \quad k = 0, \ldots, m$$

- Lagrange polynomials: degree at most n

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

- Osculating polynomials: ?

Osculating Polynomials

Definition 2

Let $x_0, x_1, \ldots, x_n$ *be* $n+1$ *distinct numbers in* $[a,b]$ *and for* $i = 0, 1, \ldots, n$ *let* m_i *be a nonnegative integer. Suppose that* $f \in C^m[a,b]$, *where* $m = \max_{0 \le i \le n} m_i$. *The* ***osculating polynomial*** *approximating* f *is the polynomial* $P(x)$ *of least degree such that*

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}, \quad \text{for each } i = 0, 1, \ldots, n, \quad \text{and} \quad k = 0, 1, \ldots, m_i.$$

- m_0th Taylor polynomials for f at x_0: ?
- nth Lagrange polynomials for f on $x_0, x_1, \ldots, x_n$: ?
- **Hermite polynomials** for f on $x_0, x_1, \ldots, x_n$: ?

Hermite Polynomials

- Hermite Polynomials:

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}, \quad i = 0, \ldots, n,\ k = 0, 1$$

have the "shape" as the function at $(x_i, f(x_i))$ in the sense that the *tangent lines* to the polynomial and the function agree.

- Construction of Hermite polynomials: degree at most ?
- Uniqueness and the error formula of Hermite polynomials

Recall: Uniqueness of Lagrange Polynomials

- Consider $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ that satisfies $P(x_i) = y_i$ for $i = 0, 1, \ldots, n$. Then $a_0, \ldots, a_n$ satisfy

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

Recall: Uniqueness of Lagrange Polynomials (cont'd)

- $a_0, \ldots, a_n$ are uniquely determined when the following Vandermonde matrix

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

is non-singular.

- The Vandermonde determinant

$$\det\{V\} = \prod_{0 \le i < j \le n} (x_j - x_i)$$

is nonzero if all points are distinct.

Hermite Polynomials (cont'd)

Theorem 8

If $f \in C^1[a,b]$ and $x_0, \ldots, x_n \in [a,b]$ are distinct, the unique polynomial of least degree agreeing with f and f' at $x_0, \ldots, x_n$ is the Hermite polynomial of degree at most $2n+1$ given by

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

where, for $L_{n,j}(x)$ denoting the jth Lagrange coefficient polynomial of degree n, we have

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

Moreover, if $f \in C^{2n+2}[a,b]$, then

$$f(x) = H_{2n+1}(x) + \frac{(x - x_0)^2 \cdots (x - x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi(x)),$$

for some (generally unknown) $\xi(x)$ in the interval (a,b).

Hermite Polynomials (cont'd)

Proof.

- $H_{2n+1}(x_i) = f(x_i)$
- $H'_{2n+1}(x_i) = f'(x_i)$
- Uniqueness
- Error formula (Exercise 11)

$H_{2n+1}(x_i) = f(x_i)$ of Hermite Polynomials

Proof. Recall $L_{n,j}(x_i) = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$

$H'_{2n+1}(x_i) = f'(x_i)$ of Hermite Polynomials

Proof. We have

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

Hence, $H'_{n,j}(x_i) = 0$ when $i \neq j$. When $i = j$, we have $L_{n,i}(x_i) = 1$, so

$$H'_{n,i}(x_i) = ?$$

Finally,

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

so $\hat{H}'_{n,j}(x_i) = 0$ if $i \neq j$ and $\hat{H}'_{n,i}(x_i) = 1$, yielding $H'_{2n+1}(x_i) = f'(x_i)$.

Uniqueness of Hermite Polynomials

Proof. Consider $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2n+1} x^{2n+1}$ that satisfies $P(x_i) = y_i$ and $P'(x_i) = z_i$ for $i = 0, 1, \ldots, n$.

- Then $a_0, \ldots, a_n$ satisfy

$$\begin{array}{cc} a_0 + a_1 x_0 + \cdots + a_{2n+1} x_0^{2n+1} = y_0, & a_1 + 2a_2 x_0 + \cdots + (2n+1) a_n x_0^{2n} = z_0, \\ \vdots & \vdots \\ a_0 + a_1 x_n + \cdots + a_{2n+1} x_n^{2n+1} = y_n, & a_1 + 2a_2 x_n + \cdots + (2n+1) a_n x_n^{2n} = z_n. \end{array}$$

- $a_0, \ldots, a_n$ are uniquely determined when the following is non-singular.

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{2n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{2n+1} \\ 0 & 1 & 2x_0 & \cdots & (2n+1) x_0^{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 2x_n & \cdots & (2n+1) x_n^{2n} \end{pmatrix}$$

Example of Hermite Polynomials

Ex. Construct a cubic polynomial H_3 such that

$$H_3(0) = 0,\ H_3(1) = 1,\ H_3'(0) = 1,\ \text{and}\ H_3'(1) = 0.$$

Hermite Polynomials Using Divided Differences

- Recall: Newton's divided-difference formula

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)\cdots(x - x_{k-1}),$$

where

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{n-1}]}{x_k - x_0}.$$

- Define a new sequence $\{z_i\}_{i=0}^{2n+1}$ by

$$z_{2i} = z_{2i+1} = x_i, \quad i = 0, 1, \ldots, n.$$

and construct the Hermite polynomial

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \ldots, z_k](x - z_0)\cdots(x - z_{k-1}).$$

Hermite Polynomials Using Divided Differences (cont'd)

- Let $f[z_{2i}, z_{2i+1}] = f[x_i, x_i] = f'(x_i)$.

z	$f(z)$	First divided differences	Second divided differences
$z_0 = x_0$	$f[z_0] = f(x_0)$		
		$f[z_0, z_1] = f'(x_0)$	
$z_1 = x_0$	$f[z_1] = f(x_0)$		$f[z_0, z_1, z_2] = \dfrac{f[z_1, z_2] - f[z_0, z_1]}{z_2 - z_0}$
		$f[z_1, z_2] = \dfrac{f[z_2] - f[z_1]}{z_2 - z_1}$	
$z_2 = x_1$	$f[z_2] = f(x_1)$		$f[z_1, z_2, z_3] = \dfrac{f[z_2, z_3] - f[z_1, z_2]}{z_3 - z_1}$
		$f[z_2, z_3] = f'(x_1)$	
$z_3 = x_1$	$f[z_3] = f(x_1)$		$f[z_2, z_3, z_4] = \dfrac{f[z_3, z_4] - f[z_2, z_3]}{z_4 - z_2}$
		$f[z_3, z_4] = \dfrac{f[z_4] - f[z_3]}{z_4 - z_3}$	
$z_4 = x_2$	$f[z_4] = f(x_2)$		$f[z_3, z_4, z_5] = \dfrac{f[z_4, z_5] - f[z_3, z_4]}{z_5 - z_3}$
		$f[z_4, z_5] = f'(x_2)$	
$z_5 = x_2$	$f[z_5] = f(x_2)$		

Hermite Polynomials Using Divided Differences (cont'd)

Ex. Use the Hermite polynomial that agrees with the data to find an approximation of $f(1.5)$.

k	x_k	$f(x_k)$	$f'(x_k)$
0	1.3	0.6200860	−0.5220232
1	1.6	0.4554022	−0.5698959
2	1.9	0.2818186	−0.5811571

1.3	0.6200860					
		−0.5220232				
1.3	0.6200860		−0.0897427			
		−0.5489460		0.0663657		
1.6	0.4554022		−0.0698330		0.0026663	
		−0.5698959		0.0679655		−0.0027738
1.6	0.4554022		−0.0290537		0.0010020	
		−0.5786120		0.0685667		
1.9	0.2818186		−0.0084837			
		−0.5811571				
1.9	0.2818186					

Hermite Polynomials Using Divided Differences (cont'd)

Hermite Interpolation

To obtain the coefficients of the Hermite interpolating polynomial $H(x)$ on the $(n+1)$ distinct numbers $x_0, \ldots, x_n$ for the function f:

INPUT numbers $x_0, x_1, \ldots, x_n$; values $f(x_0), \ldots, f(x_n)$ and $f'(x_0), \ldots, f'(x_n)$.

OUTPUT the numbers $Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1}$ where

$$\begin{aligned} H(x) = {} & Q_{0,0} + Q_{1,1}(x-x_0) + Q_{2,2}(x-x_0)^2 + Q_{3,3}(x-x_0)^2(x-x_1) \\ & + Q_{4,4}(x-x_0)^2(x-x_1)^2 + \cdots \\ & + Q_{2n+1,2n+1}(x-x_0)^2(x-x_1)^2 \cdots (x-x_{n-1})^2(x-x_n). \end{aligned}$$

Step 1 For $i = 0, 1, \ldots, n$ do Steps 2 and 3.

Step 2 Set $z_{2i} = x_i$;
$z_{2i+1} = x_i$;
$Q_{2i,0} = f(x_i)$;
$Q_{2i+1,0} = f(x_i)$;
$Q_{2i+1,1} = f'(x_i)$.

Step 3 If $i \neq 0$ then set

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

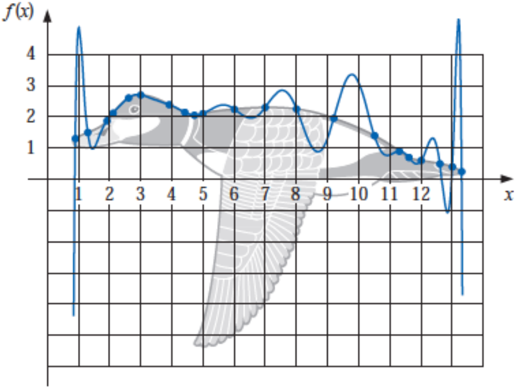
Step 4 For $i = 2, 3, \ldots, 2n+1$

for $j = 2, 3, \ldots, i$ set $Q_{i,j} = \dfrac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}$.

Step 5 OUTPUT $(Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1})$;
STOP

Runge Phenomenon

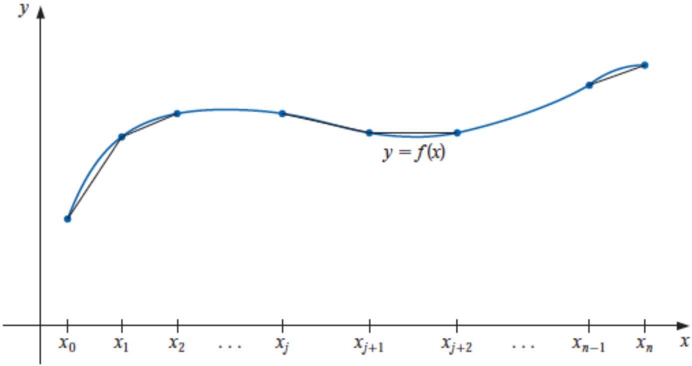
Ex. Runge Phenomenon: Lagrange polynomial of degree 20



Piecewise-Polynomial Approximation

- Piecewise-polynomial approximation: divide the interval into a collection of subintervals and construct a polynomial on each subinterval.

- The simplest version is a **piecewise-linear** interpolation.



Q. Is this good enough?

Piecewise-Hermite Cubic Approximation

- Given f and f' at the points $x_0 < x_1 < \cdots < x_n$, one can use Hermite cubic polynomial on each subinterval $[x_i, x_{i+1}]$ to construct a function that has a continuous derivative on the entire interval $[x_0, x_n]$.

Piecewise-Quadratic Approximation

- Let f be defined on $[a, b]$, and let the nodes $a = x_0 < x_1 < x_2 = b$ be given. A quadratic spline interpolating function S consists of the quadratic polynomial

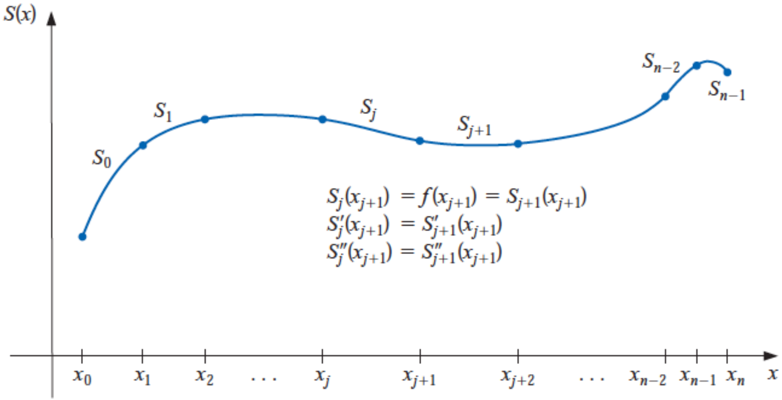
$$S_0(x) = a_0 + b_0(x - x_0) + c_0(x - x_0)^2 \quad \text{on } [x_0, x_1]$$

and

$$S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 \quad \text{on } [x_1, x_2]$$

Q. What are the conditions that we would like to impose?

Cubic Splines



Cubic Splines (cont'd)

Definition 3

Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a **cubic spline interpolant** *S for f is a function that satisfies the following conditions:*

(a) *$S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \ldots, n-1$;*

(b) *$S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \ldots, n-1$;*

(c) *$S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$; (Implied by (b).)*

(d) *$S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;*

(e) *$S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;*

(f) *One of the following sets of boundary conditions is satisfied:*

  (1) $S''(x_0) = S''(x_n) = 0$ (**natural** *(or free)* **boundary**);

  (2) $S'(x_0) = f'(x_0)$ *and* $S'(x_n) = f'(x_n)$ (**clamped boundary**).

- A **spline** is a special function defined piecewise by polynomials.

Cubic Splines (cont'd)

Ex. Construct a natural cubic spline that passes through the points $(1, 2)$, $(2, 3)$, and $(3, 5)$.

Construction of a Cubic Spline

- Apply conditions to the cubic polynomials

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

for each $j = 0, 1, \ldots, n - 1$.

- Since $S_j(x_j) = a_j = f(x_j)$, we have

Construction of a Cubic Spline (cont'd)

- Similarly, let $b_n = S'(x_n)$, and we have

$$b_{j+1} = S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}) = b_j + 2c_j h_j + 3d_j h_j^2,$$

for each $j = 0, 1, \ldots, n-1$.

- Let $c_n = S''(x_n)/2$, and we have

$$c_{j+1} = S''_{j+1}(x_{j+1})/2 = S''_j(x_{j+1})/2 = c_j + 3d_j h_j$$

for each $j = 0, 1, \ldots, n-1$.

Construction of a Cubic Spline (cont'd)

- Removing d_j yields

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}).$$
$$b_{j+1} = b_j + h_j(c_j + c_{j+1}).$$

- Further removing b_j yields

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

for each $j = 1, 2, \ldots, n-1$.

Natural Splines

Theorem 9

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$, then f has a unique natural spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is, a spline interpolant that satisfies the natural boundary conditions $S''(a) = 0$ and $S''(b) = 0$

Natural Splines (cont'd)

Proof. The equations with the boundary conditions $c_n = S''(x_n)/2 = 0$ and $0 = S''(x_0) = 2c_0$ yield a linear system $\boldsymbol{Ax} = \boldsymbol{b}$, where

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Natural Splines (cont'd)

Natural Cubic Spline

To construct the cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

INPUT $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \ldots, n-1$.

(*Note:* $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ *for* $x_j \leq x \leq x_{j+1}$.)

Step 1 For $i = 0, 1, \ldots, n-1$ set $h_i = x_{i+1} - x_i$.

Step 2 For $i = 1, 2, \ldots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

Step 3 Set $l_0 = 1$; (*Steps 3, 4, 5, and part of Step 6 solve a tridiagonal linear system using a method described in Algorithm 6.7.*)
$\mu_0 = 0$;
$z_0 = 0$.

Step 4 For $i = 1, 2, \ldots, n-1$
set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;
$\mu_i = h_i / l_i$;
$z_i = (\alpha_i - h_{i-1} z_{i-1}) / l_i$.

Step 5 Set $l_n = 1$;
$z_n = 0$;
$c_n = 0$.

Step 6 For $j = n-1, n-2, \ldots, 0$
set $c_j = z_j - \mu_j c_{j+1}$;
$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;
$d_j = (c_{j+1} - c_j)/(3h_j)$.

Step 7 OUTPUT $(a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n-1)$;
STOP.

Natural Splines (cont'd)

Ex. Use the data points $(0,1)$,$(1,e)$,$(2,e^2)$, and $(3,e^3)$ to form a natural spline $S(x)$ that approximates $f(x)=e^x$.

Natural Splines (cont'd)

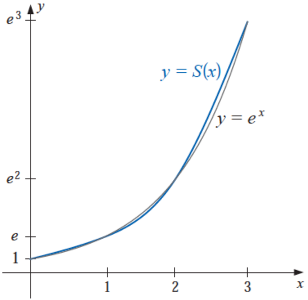
Sol. The system has the solution

$$\begin{aligned} c_0 &= 0, \\ c_1 &= \frac{1}{5}(-e^3 + 6e^2 - 9e + 4) \approx 0.757, \\ c_2 &= \frac{1}{5}(4e^3 - 9e^2 + 6e - 1) \approx 5.830, \\ c_3 &= 0. \end{aligned}$$

After solving for the remaining constants, we have

$$S(x) = \begin{cases} 1 + 1.466x + 0.252x^3, & x \in [0, 1], \\ 2.718 + 2.223(x-1) + 0.757(x-1)^2 + 1.691(x-1)^3, & x \in [1, 2], \\ 7.389 + 8.810(x-2) + 5.830(x-2)^2 - 1.943(x-2)^3, & x \in [2, 3]. \end{cases}$$

Natural Splines (cont'd)



- To approximate the integral of $f(x) = e^x$ on $[0, 3]$, which has the value

$$\int_0^3 e^x dx = e^3 - 1 \approx 19.08553692,$$

we piecewise integrate the splines:

$$\int_0^3 S(x)dx = ?$$

Clamped Splines

Ex. Construct a clamped cubic spline s that passes through the points $(1, 2)$, $(2, 3)$, and $(3, 5)$ that has ? and ?.

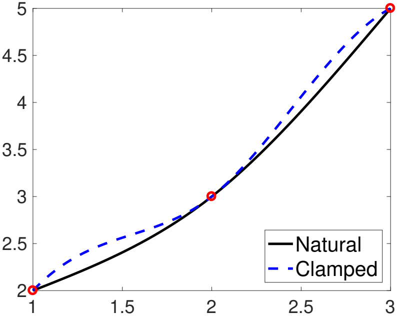
Clamped Splines (cont'd)

Sol. Solving the system of equation gives the clamped cubic spline

$$s(x) = \begin{cases} 2 + 2(x-1) - \frac{5}{2}(x-1)^2 + \frac{3}{2}(x-1)^3, & x \in [1,2], \\ 3 + \frac{3}{2}(x-2) + 2(x-2)^2 - \frac{3}{2}(x-2)^3, & x \in [2,3]. \end{cases}$$

Recall: Natural cubic spline:

$$S(x) = \begin{cases} 2 + \frac{3}{4}(x-1) + \frac{1}{4}(x-1)^3, & x \in [1,2], \\ 3 + \frac{3}{2}(x-2) + \frac{3}{4}(x-2)^2 - \frac{1}{4}(x-2)^3, & x \in [2,3]. \end{cases}$$



Clamped Splines (cont'd)

Theorem 10

If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$ and differentiable at a and b, then f has a unique clamped spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is, a spline interpolant that satisfies the clamped boundary conditions $S'(a) = f'(a)$ and $S'(b) = f'(b)$.

Proof.

$$A = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & & h_{n-1} \\ 0 & \cdots & 0 & & h_{n-1} & 2h_{n-1} \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(a) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}, \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

Clamped Splines (cont'd)

Ex. Use the data points $(0,1)$,$(1,e)$,$(2,e^2)$,$(3,e^3)$ and ? to form a clamped spline $S(x)$ that approximates $f(x)=e^x$.

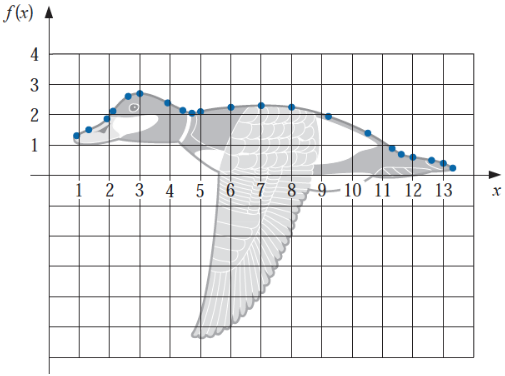
Sol. It is

$$s(x)=\begin{cases}1+x+0.445x^2+0.274x^3, & x\in[0,1],\\ 2.718+2.710(x-1)+1.265(x-1)^2+0.695(x-1)^3, & x\in[1,2],\\ 7.389+7.327(x-2)+3.351(x-2)^2+2.019(x-2)^3, & x\in[2,3].\end{cases}$$

- Recall: Natural cubic spline:

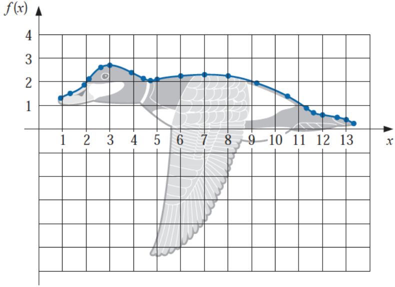
$$S(x)=\begin{cases}1+1.466x+0.252x^3, & x\in[0,1],\\ 2.718+2.223(x-1)+0.757(x-1)^2+1.691(x-1)^3, & x\in[1,2],\\ 7.389+8.810(x-2)+5.830(x-2)^2-1.943(x-2)^3, & x\in[2,3].\end{cases}$$

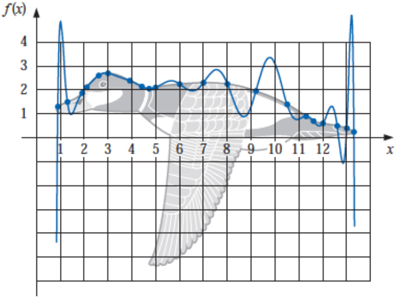
Comparison to Lagrange Polynomial



Comparison to Lagrange Polynomial (cont'd)

- Natural Cubic Spline vs. Lagrange polynomial of degree 20





Error Bound for Clamped Splines

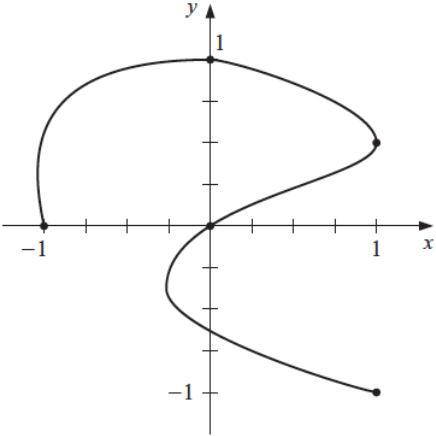
Theorem 11

Let $f \in C^4[a,b]$ with $\max_{a \le x \le b} |f^{(4)}(x)| = M$. If s is the unique clamped cubic spline interpolant to f with respect to the nodes $a = x_0 < x_1 < \cdots < x_n = b$, then for all x in $[a,b]$,

$$|f(x) - s(x)| \le \frac{5M}{384} \max_{0 \le j \le n-1} (x_{j+1} - x_j)^4.$$

- A fourth-order error-bound for the natural boundary conditions exist, but it is more complicated so omitted here.
- In general, the natural boundary conditions will give less accurate results than the clamped conditions near the ends of the interval $[x_0, x_n]$ unless the function f happens to nearly satisfy $f''(x_0) = f''(x_n) = 0$.
- An alternative condition is the *not-a-knot* condition requiring that $S'''(x)$ be continuous at x_1 and x_{n-1}. This is a default option in the MATLAB built-in function “spline”.

Parametric Curves



- Determine a polynomial or piecewise polynomial to connect the points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ in the order given. How?

Parametric Curves (cont'd)

- Use a parameter t on an interval $[t_0, t_n]$, with $t_0 < t_1 < \cdots < t_n$, and construct approximation functions with

$$x_i = x(t_i) \quad \text{and} \quad y_i = y(t_i), \quad \text{for each } i = 0, 1, \ldots, n.$$

Parametric Curves (cont'd)

Ex. Construct a pair of Lagrange polynomials to approximate the previous curve using the following data points shown on the curve

$$(-1,0),\ (0,1),\ (1,0.5),\ (0,0),\ (1,-1).$$

Parametric Curves (cont'd)

- Choose the points $\{t_i\}_{i=0}^{4}$ equally spaced in $[0, 1]$.

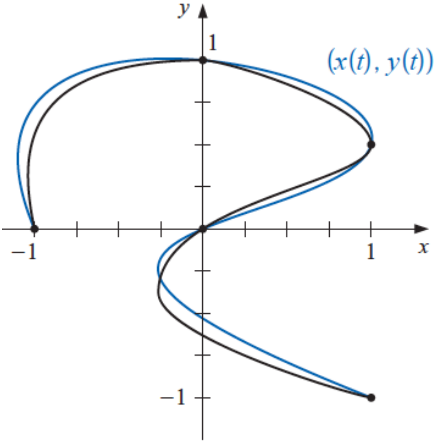
i	0	1	2	3	4
t_i	0	0.25	0.5	0.75	1
x_i	-1	0	1	0	1
y_i	0	1	0.5	0	-1

This then produces the interpolating polynomials

$$x(t) = \left(\left(\left(64t - \frac{352}{3}\right)t + 60\right)t - \frac{14}{3}\right)t - 1,$$

$$y(t) = \left(\left(\left(-\frac{64}{3}t + 48\right)t - \frac{116}{3}\right)t + 11\right)t.$$

Parametric Curves (cont'd)



Q. How can we improve the accuracy?

Piecewise Cubic Hermite Polynomial

Q. In computer graphics, piecewise cubic Hermite polynomial is preferred. Why?

Piecewise Cubic Hermite Polynomial (cont'd)

- Suppose the curve has $n+1$ data points $(x(t_0), y(t_0)), \ldots, (x(t_n), y(t_n))$. Then we must specify $x'(t_i)$ and $y'(t_i)$, for each $i = 0, 1, \ldots, n$.
- Constructing a piecewise cubic Hermite polynomial reduces to determining a pair of cubic Hermite polynomials in t, where $t_0 = 0$ and $t_1 = 1$ given $(x(0)), y(0))$, $(x(1), y(1))$, $\frac{dy}{dx}(t=0)$ and $\frac{dy}{dx}(t=1)$.

Piecewise Cubic Hermite Polynomial (cont'd)

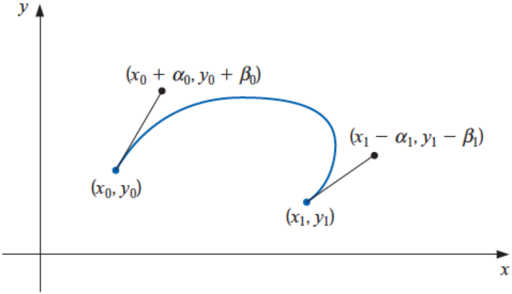
- We have six conditions for eight unknowns, and we need to specify $x'(0), x'(1), y'(0), y'(1)$ considering

$$\frac{dy}{dx}(t=0) = \frac{y'(0)}{x'(0)} \quad \text{and} \quad \frac{dy}{dx}(t=1) = \frac{y'(1)}{x'(1)}.$$

- The choice of $x'(0)$ and $y'(0)$ affects the shape of the curve. The larger the value, the closer the curve comes to approximating the tangent line near $(x(0), y(0))$.

Piecewise Cubic Hermite Polynomial (cont'd)

- To further simply the process in interactive computer graphics (using a mouse or touchpad), the derivative at an endpoint is specified by using a second point, called a *guidepoint*, on the desired tangent line.



- The farther the guidepoint is from the node, the more closely the curve approximates the tangent line near the node.

Piecewise Cubic Hermite Polynomial (cont'd)

- The cubic Hermite polynomial $x(t)$ on $[0,1]$ satisfying the conditions

$$x(0) = x_0, \quad x(1) = x_1, \quad x'(0) = \alpha_0, \quad x'(1) = \alpha_1$$

is

$$\begin{aligned} x(t) = {} & [2(x_0 - x_1) + (\alpha_0 + \alpha_1)]t^3 \\ & + [3(x_1 - x_0) - (\alpha_1 + 2\alpha_0)]t^2 + \alpha_0 t + x_0. \end{aligned}$$

- Similarly for $y(t)$, the conditions

$$y(0) = y_0, \quad y(1) = y_1, \quad y'(0) = \beta_0, \quad y'(1) = \beta_1$$

are satisfied by

$$\begin{aligned} y(t) = {} & [2(y_0 - y_1) + (\beta_0 + \beta_1)]t^3 \\ & + [3(y_1 - y_0) - (\beta_1 + 2\beta_0)]t^2 + \beta_0 t + y_0. \end{aligned}$$

Piecewise Cubic Hermite Polynomial (cont'd)

Ex Determine the graph of the parametric curve when the endpoints are $(x_0, y_0) = (0, 0)$ and $(x_1, y_1) = (1, 0)$ and respective guide points are $(2, 2)$ and $(2, -1)$.

Piecewise Cubic Hermite Polynomial (cont'd)