Chapter 3
Interpolation and Polynomial Approximation

Per-Olof Persson
persson@berkeley.edu

Department of Mathematics
University of California, Berkeley

Math 128A Numerical Analysis

Polynomial Interpolation

Polynomials

- Polynomials $P_n(x) = a_n x^n + \cdots a_1 x + a_0$ are commonly used for interpolation or approximation of functions
- Benefits include efficient methods, simple differentiation, and simple integration
- Also, Weierstrass Approximation Theorem says that for each $\varepsilon > 0$, there is a $P(x)$ such that

$$|f(x) - p(x)| < \varepsilon \qquad \text{for all } x \text{ in } [a, b]$$

  for $f \in C[a, b]$. In other words, polynomials are good at approximating general functions.

The Lagrange Polynomial

Theorem

If $x_0, \ldots, x_n$ distinct and f given at these numbers, a unique polynomial $P(x)$ of degree $\leq n$ exists with

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

The polynomial is

$$P(x) = f(x_0)L_{n,0}(x) + \ldots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x)$$

where

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \\ &= \prod_{i\neq k} \frac{(x-x_i)}{(x_k-x_i)} \end{aligned}$$

Lagrange Polynomial Error Term

Theorem

$x_0, \ldots, x_n$ distinct in $[a, b]$, $f \in C^{n+1}[a, b]$, then for $x \in [a, b]$ there exists $\xi(x)$ in (a, b) 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 interpolating polynomial.

Divided Differences

Divided Differences

- Write the nth Lagrange polynomial in the form

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

- Introduce the *kth divided difference*

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

- The coefficients are then $a_k = f[x_0, x_1, x_2, \dots, x_k]$ and

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

Newton's Divided-Difference

MATLAB Implementation

```
function F = divideddifference(x, f)
% Compute interpolating polynomial using Divided Differences.

n = length(x)-1;
F = zeros(n+1,n+1);
F(:,1) = f(:);
for i = 1:n
    for j = 1:i
        F(i+1,j+1) = (F(i+1,j) - F(i,j)) / (x(i+1) - x(i-j+1));
    end
end
```

Equally Spaced Nodes

Equal Spacing

- Suppose $x_0, \ldots, x_n$ increasing with equal spacing $h = x_{i+1} - x_i$ and $x = x_0 + sh$
- The Newton Forward-Difference Formula then gives

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

where

$$\begin{aligned}
\Delta f(x_0) &= f(x_1) - f(x_0) \\
\Delta^2 f(x_0) &= \Delta f(x_1) - \Delta f(x_0) = f(x_2) - 2f(x_1) + f(x_0) \\
&\ldots
\end{aligned}$$

Backward Differencing

The Newton Backward-Difference Formula

- Reordering the nodes gives

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

- The Newton Backward-Difference Formula is

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

where the *backward difference* ∇p_n is defined by

$$\begin{aligned} \nabla p_n &= p_n - p_{n-1} \\ \nabla^k p_n &= \nabla(\nabla^{k-1} p_n) \end{aligned}$$

Osculating Polynomials

Definition

Let $x_0, \dots, x_n$ be distinct in $[a, b]$, and m_i nonnegative integers. Suppose $f \in C^m[a, b]$, with $m = \max_{0 \le i \le n} m_i$. The *osculating polynomial* approximating f is the $P(x)$ of least degree such that

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}, \qquad \text{for } i = 0, \dots, n \text{ and } k = 0, \dots, m_i$$

Special Cases

- $n = 0$: m_0th Taylor polynomial
- $m_i = 0$: nth Lagrange polynomial
- $m_i = 1$: Hermite polynomial

Hermite Interpolation

Theorem

If $f \in C^1[a,b]$ and $x_0, \dots, x_n \in [a,b]$ distinct, the Hermite polynomial is

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

where

$$\begin{aligned} H_{n,j}(x) &= [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x) \\ \hat{H}_{n,j}(x) &= (x - x_j) L^2_{n,j}(x). \end{aligned}$$

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 $\xi(x) \in (a,b)$.

Hermite Polynomials from Divided Differences

Divided Differences

Suppose $x_0, \dots, x_n$ and f, f' are given at these numbers. Define $z_0, \dots, z_{2n+1}$ by

$$z_{2i} = z_{2i+1} = x_i$$

Construct divided difference table, but use

$$f'(x_0), f'(x_1), \dots, f'(x_n)$$

instead of the undefined divided differences

$$f[z_0, z_1], f[z_2, z_3], \dots, f[z_{2n}, z_{2n+1}]$$

The Hermite polynomial is

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

Cubic Splines

Definition

Given a function f on $[a, b]$ and nodes $a = x_0 < \cdots < x_n = b$, a *cubic spline interpolant* S for f satisfies:

a. $S(x)$ is a cubic polynomial $S_j(x)$ on $[x_j, x_{j+1}]$
b. $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$
c. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$
d. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$
e. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$
f. One of the following boundary conditions:
    i. $S''(x_0) = S''(x_n) = 0$ (*free* or *natural* boundary)
    ii. $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (*clamped* boundary)

Natural Splines

Computing Natural Cubic Splines

Solve for coefficients a_j, b_j, c_j, d_j in

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

by setting $a_j = f(x_j)$, $h_j = x_{j+1} - x_j$, and solving $A\mathbf{x} = \mathbf{b}$:

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

$$\mathbf{b} = (0, 3(a_2 - a_1)/h_1 - 3(a_1 - a_0)/h_0, \dots, \\ 3(a_n - a_{n-1})/h_{n-1} - 3(a_{n-1} - a_{n-2})/h_{n-2}, 0)^T$$

$$\mathbf{x} = (c_0, \dots, c_n)^T$$

Finally,

$$b_j = (a_{j+1} - a_j)/h_j - h_j(2c_j + c_{j+1})/3, \qquad d_j = (c_{j+1} - c_j)/(3h_j)$$

Clamped Splines

Computing Clamped Cubic Splines

Solve for coefficients a_j, b_j, c_j, d_j in

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

using same procedure as for natural cubic splines, but with

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

$$\begin{aligned} \mathbf{b} = (&3(a_1 - a_0)/h_0 - 3f'(a), 3(a_2 - a_1)/h_1 - 3(a_1 - a_0)/h_0, \dots, \\ &3(a_n - a_{n-1})/h_{n-1} - 3(a_{n-1} - a_{n-2})/h_{n-2}, \\ &3f'(b) - 3(a_n - a_{n-1})/h_{n-1})^T \end{aligned}$$