

Interpolation and Polynomial Approximation

1. Weierstrass Approximation Theorem

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

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

2. Polynomial Interpolation

- Problem: Let $f(x_i) = f_i$ for $i = 0, 1, \dots, n$. Find a polynomial $p(x)$ passing the points $(x_i, f(x_i))$ for $i = 0, 1, \dots, n$, i.e., $p(x_i) = f_i$ for $i = 0, 1, \dots, n$.
- Interpolation Theorem:
There is a unique polynomial of degree n which interpolates $f(x)$ at distinct points $x_0, x_1, \dots, x_n \in [a, b]$.

3. Method of undetermined coefficients:

- Given (x_i, f_i) for $i = 0, 1, \dots, n$, choose $1, x, x^2, \dots, x^n$ as basis functions (monomial basis). Then the interpolating polynomial has the form

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n.$$

We want to find coefficients a_1, a_2, \dots, a_n such that $p(x_i) = f_i$ for $i = 0, 1, \dots, n$.

- Vandermonde matrix system

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

4. Lagrange interpolation

- Lagrange formula

$$p(x) = a_0l_0(x) + a_1l_1(x) + \dots + a_nl_n(x)$$

Let $l_i(x)$ be a polynomial of degree n for $i = 0, 1, \dots, n$ and

$$l_i(x) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Then

$$p(x_j) = \sum_{i=0}^n a_i l_i(x_j) = a_j l_j(x_j) = a_j = f_j$$

therefore

$$p(x) = \sum_{i=0}^n f_i l_i(x).$$

- $l_i(x)$ is called a basis function for polynomial of degree n and given by

$$l_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

5. Evaluating a polynomial

- Horner's method: Let $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. $p_n(x)$ can be written in the following nested form

$$\begin{aligned} p_n(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \\ &= a_0 + x(a_1 + a_2x + \cdots + a_nx^{n-1}) \\ &= a_0 + x(a_1 + x(a_2 + \cdots + a_nx^{n-2})) \\ &\quad \vdots \\ &= a_0 + x(a_1 + x(a_2 + \cdots (a_{n-1} + a_nx) \cdots)) \end{aligned}$$

- Note: Matlab has several built-in functions which can be used to aid in polynomial interpolation. *polyfit* determines the coefficients of a polynomial from the given interpolation points. *polyval* evaluates a polynomial at a vector of points.

6. Iterative interpolation (Neville's algorithm)

- Idea: Given (x_i, f_i) for $i = 0, 1, \dots, n$, let $p_{m_1m_2\cdots m_k}(x)$ ($1 \leq m_j \leq n$) be a polynomial of degree less than k and

$$p_{m_1m_2\cdots m_k}(x_{m_j}) = f_{m_j} \quad \text{for } j = 1, \dots, k$$

i.e., $p_{m_1m_2\cdots m_k}(x)$ interpolates $(x_{m_1}, f_{m_1}), (x_{m_2}, f_{m_2}), \dots, (x_{m_k}, f_{m_k})$.

Interpolating polynomials are linked by the following recursion:

$$\begin{aligned} p_m(x_m) &= f_m \quad \text{for } m = 1, \dots, n \\ p_{m_1m_2\cdots m_k}(x) &= \frac{(x-x_{m_1})p_{m_2m_3\cdots m_k}(x) - (x-x_{m_k})p_{m_1m_2\cdots m_{k-1}}(x)}{x_{m_k} - x_{m_1}} \end{aligned} \quad (1)$$

for $k = 2, \dots, n$.

- $p_{m_1m_2\cdots m_k}(x)$ defined in (1) interpolates $(x_{m_1}, f_{m_1}), (x_{m_2}, f_{m_2}), \dots, (x_{m_k}, f_{m_k})$.

If $x = x_{m_1}$,

$$p_{m_1m_2\cdots m_k}(x_{m_1}) = \frac{0 - (x_{m_1} - x_{m_k})p_{m_1m_2\cdots m_{k-1}}(x_{m_1})}{x_{m_k} - x_{m_1}} = p_{m_1m_2\cdots m_{k-1}}(x_{m_1}) = f_{m_1}$$

If $x = x_{m_j}$ for $1 < j < k$,

$$\begin{aligned} p_{m_1m_2\cdots m_k}(x_{m_j}) &= \frac{(x_{m_j} - x_{m_1})p_{m_2m_3\cdots m_k}(x_{m_j}) - (x_{m_j} - x_{m_k})p_{m_1m_2\cdots m_{k-1}}(x_{m_j})}{x_{m_k} - x_{m_1}} \\ &= \frac{(x_{m_j} - x_{m_1})f_{m_j} - (x_{m_j} - x_{m_k})f_{m_j}}{x_{m_k} - x_{m_1}} = f_{m_j} \end{aligned}$$

If $x = x_{m_k}$,

$$p_{m_1m_2\cdots m_k}(x_{m_k}) = \frac{(x_{m_k} - x_{m_1})p_{m_2m_3\cdots m_k}(x_{m_k}) - 0}{x_{m_k} - x_{m_1}} = p_{m_2\cdots m_k}(x_{m_k}) = f_{m_k}$$

- The result in (1) is used in the Neville's algorithm as follows:

$$\begin{array}{cccccc}
 (x_0, f_0) & f_0 = p_0(x) & & & & \\
 (x_1, f_1) & f_1 = p_1(x) & p_{01}(x) & & & \\
 (x_2, f_2) & f_2 = p_2(x) & p_{12}(x) & p_{012}(x) & & \\
 (x_3, f_3) & f_3 = p_3(x) & p_{23}(x) & p_{123}(x) & p_{0123} & \\
 (x_4, f_4) & f_4 = p_4(x) & p_{34}(x) & p_{423}(x) & p_{1234} & p_{01234} \\
 \vdots & \vdots & \vdots & \vdots & &
 \end{array}$$

- Remark: Neville's algorithm aims at evaluating the interpolating polynomial p at a single value of x . It is less suitable for determining the interpolating polynomial itself.

7. Newton interpolation

- Given (x_i, f_i) for $i = 0, 1, \dots, n$, choose $1, x - x_0, (x - x_0)(x - x_1), \dots, (x - x_0)(x - x_1) \dots (x - x_{n-1})$ as basis functions. Then the interpolating polynomial has the form

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}).$$

We want to find coefficients a_0, a_1, \dots, a_n such that $p(x_i) = f_i$ for $i = 0, 1, \dots, n$.

- Undetermined coefficients method

$$\begin{array}{lcl}
 p(x_0) = f_0 & \rightarrow & a_0 = f_0 \\
 p(x_1) = f_1 & \rightarrow & a_0 + a_1(x_1 - x_0) = f_1 \\
 & \vdots & \vdots \\
 p(x_n) = f_n & \rightarrow & a_0 + a_1(x_n - x_0) + a_2(x_n - x_0)(x_n - x_1) + \dots + a_n(x_n - x_0) \dots (x_n - x_{n-1}) = f_n
 \end{array}$$

The above lower triangular system can be solved by the forward substitution.

- Once coefficients are determined, p is evaluated by the Horner's method:

$$p(x) = a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \dots (a_{n-1} + a_n(x - x_{n-1}) \dots)))$$

- Divided difference

Define divided differences as follows:

$$\begin{aligned}
 f[x_i] &= f_i && \text{for } i = 0, 1, \dots, n \quad (\text{zeroth divided difference}) \\
 f[x_i, x_{i+1}] &= \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} && (\text{first divided difference}) \\
 f[x_i, x_{i+1}, \dots, x_{i+k}] &= \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i} && (k\text{th divided difference})
 \end{aligned}$$

Then it can be shown that

$$p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}).$$

Generation of divided differences:

$$\begin{aligned}
 f[x_0] &= f_0 \\
 f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\
 f[x_1] &= f_1 \\
 f[x_1, x_2] &= \frac{f[x_2] - f[x_1]}{x_2 - x_1} \\
 f[x_2] &= f_2 \\
 f[x_2, x_3] &= \frac{f[x_3] - f[x_2]}{x_3 - x_2} \\
 f[x_3] &= f_3 \\
 f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
 f[x_1, x_2, x_3] &= \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} \\
 f[x_0, x_1, x_2, x_3] &= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}
 \end{aligned}$$

8. Error Analysis

- Theorem:

Let $f \in C^{n+1}[a, b]$ and $p(x)$ be the polynomial which interpolates $f(x)$ at the points x_0, x_1, \dots, x_n , where $x_0 = a, x_n = b$ and $x_i \in (a, b)$ for $i = 1, \dots, n - 1$. Then

$$f(x) - p(x) = \frac{f^{(n+1)}(\theta)}{(n+1)!} w(x),$$

where $\theta \in [a, b]$ and $w(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$.

- Using the theorem, we can get a rough bound of the approximation error:

$$\begin{aligned}
 |f(x) - p(x)| &\leq \frac{|f^{(n+1)}(\theta)|}{(n+1)!} |w(x)| \\
 &\leq \frac{\max_{x \in [a, b]} |f^{(n+1)}(\theta)|}{(n+1)!} (b - a)^{n+1} \quad \text{if } x \in [a, b]
 \end{aligned}$$

- Remarks

- $|w(x)|$ grows very fast for \hat{x} outside $[a, b]$. Therefore, $p(\hat{x})$ is usually not a good approximation to $f(\hat{x})$.
- An interpolating polynomial with higher degree (using more data points) does not always result in a better approximation. Hence, if n is too large, sometimes it is better to look for a different approximation such as spline interpolation.

9. Hermite cubic interpolation

- Given (x_i, f_i) and (x_i, f'_i) for $i = 0, 1, \dots, n$, we seek for a piecewise cubic interpolating polynomial $p(x)$ satisfying the following.
 - a. $p(x)$ is cubic in each interval $[x_i, x_{i+1}]$.
 - b. $p(x)$ and $p'(x)$ are continuous such that $p(x_i) = f_i$ and $p'(x_i) = f'_i$ for $i = 0, 1, \dots, n$.
- Undetermined coefficients method
Let $p(x)$ be $p_i(x) = a_i + b_i x + c_i x^2 + d_i x^3$ on $[x_i, x_{i+1}]$. Then from the conditions $p(x_i) = f_i$, $p(x_{i+1}) = f_{i+1}$, $p'(x_i) = f'_i$ and $p'(x_{i+1}) = f'_{i+1}$, $4n$ equations with $4n$ unknowns are derived.

- Basis function method

Let

$$\begin{aligned}\phi_{i0}(x) &= \left(\frac{x_{i+1}-x}{x_{i+1}-x_i}\right)^2 \left[1 + 2\left(\frac{x-x_i}{x_{i+1}-x_i}\right)\right] \\ \phi_{i+1,0}(x) &= \left(\frac{x-x_i}{x_{i+1}-x_i}\right)^2 \left[1 + 2\left(\frac{x_{i+1}-x}{x_{i+1}-x_i}\right)\right] \\ \phi_{i1}(x) &= \left(\frac{x_{i+1}-x}{x_{i+1}-x_i}\right)^2 (x - x_i) \\ \phi_{i+1,1}(x) &= -\left(\frac{x-x_i}{x_{i+1}-x_i}\right)^2 (x_{i+1} - x)\end{aligned}$$

Then, on each $[x_i, x_{i+1}]$,

$$p(x) = f_i \phi_{i0}(x) + f'_i \phi_{i1}(x) + f_{i+1} \phi_{i+1,0}(x) + f'_{i+1} \phi_{i+1,1}(x).$$

10. Cubic spline interpolation

- Given (x_i, f_i) for $i = 0, 1, \dots, n$, we seek for a twice continuously differentiable piecewise polynomial $p(x)$ which is cubic in each subinterval $[x_i, x_{i+1}]$ and interpolates (x_i, f_i) for $i = 0, 1, \dots, n$. Let $p(x) = p_i(x)$ on $[x_i, x_{i+1}]$ for $i = 0, 1, \dots, n - 1$. Then,

a. $p_i(x)$ is cubic in $[x_i, x_{i+1}]$ for $i = 0, 1, \dots, n - 1$

b. $p_i(x_i) = f_i$ for $i = 0, 1, \dots, n - 1$

c. $p_i(x_{i+1}) = f_{i+1}$ for $i = 0, 1, \dots, n - 1$

d. $p'_i(x_{i+1}) = p'_{i+1}(x_{i+1})$ for $i = 0, 1, \dots, n - 2$

e. $p''_i(x_{i+1}) = p''_{i+1}(x_{i+1})$ for $i = 0, 1, \dots, n - 2$

- There are total $4n$ unknowns(coefficients) to be determined. Total $4n - 2$ equations ($2n$ from b, c and $2n - 2$ from d, e) are derived from the conditions b–e.

- In order to determine the coefficients uniquely, two more conditions are needed. The following additional conditions are used in practice.

a. $p''(x_0) = p''(x_n) = 0$ (natural cubic spline or free boundary spline)

b. $p'(x_0) = f'_0$ and $p'(x_n) = f'_n$ (complete cubic spline or clamped boundary spline)

c. $p'(x_0) = p'(x_n)$ and $p''(x_0) = p''(x_n)$ (periodic cubic spline)

- Calculation of cubic spline

Since $p(x)$ is piecewise cubic, $p'(x)$ and $p''(x)$ are piecewise quadratic and linear, respectively, and they are continuous. Let $M_i = p''(x_i)$ for $i = 0, 1, \dots, n$ and $p(x) = p_i(x)$ on $[x_i, x_{i+1}]$ for $i = 0, 1, \dots, n - 1$. Then, $p''_i(x_i) = M_i$ and $p''_{i+1}(x_{i+1}) = M_{i+1}$. Using the interpolation conditions $p_i(x_i) = f_i$, $p_i(x_{i+1}) = f_{i+1}$ and the continuity of $p'(x)$ at x_i , we can have the following 3-moment equations

$$\frac{h_{i-1}}{6}M_{i-1} + \frac{h_{i-1} + h_i}{3}M_i + \frac{h_i}{6}M_{i+1} = \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}$$

for $i = 1, 2, \dots, n - 1$, where $h_i = x_{i+1} - x_i$ for $i = 0, 1, \dots, n - 1$

- Properties of cubic spline interpolating polynomials

- a. Out of all functions in $C^2[x_0, x_n]$ which interpolate (x_i, f_i) for $i = 0, 1, \dots, n$, the natural cubic spline $p(x)$ has the smallest 2nd derivative measured in L^2 -norm, i.e.,

$$\int_{x_0}^{x_n} (p''(x))^2 dx \leq \int_{x_0}^{x_n} (h''(x))^2 dx$$

for any $h \in C^2[x_0, x_n]$ interpolates (x_i, f_i) for $i = 0, 1, \dots, n$.

- b. If $f \in C^4[x_0, x_n]$ and $p(x)$ is a cubic spline which approximates f ,

$$\max_{x_i \leq x \leq x_{i+1}} |f(x) - p(x)| \leq \frac{\max_{x_i \leq x \leq x_{i+1}} |f^{(4)}(x)|}{4!} (x_{i+1} - x_i)^4.$$