6.1 Polynomial interpolation

Question: If $f \in C[a,b]$ is given by $f(x_i) = y_i$, $i = 0, 1, \cdots, n$,

how can we approximate f?

- **Weierstrass approximation Thm**: Let $f \in C[a,b]$. For any $\varepsilon > 0$,

$\exists$ polynomial $p(x)$ s.t. $|f(x) - p(x)| \leq \varepsilon$, $\forall\, x \in [a,b]$

(i.e. $\|f - p\|_{L^\infty([a,b])} \leq \varepsilon$)

- Consider the **polynomial interpolation**: a polynomial $p(x)$ w/ deg $\leq n$

s.t. $p(x_i) = y_i$, $i = 0, 1, \cdots, n$

Let $p(x) = \sum_{j=0}^{n} a_j x^j$. Then we need

$$\sum_{j=0}^{n} a_j x_i^j = y_i, \quad i = 0, 1, \cdots, n$$

$$\begin{pmatrix} x_0^0 & x_0^1 & x_0^2 & \cdots & x_0^n \\ x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ x_n^0 & x_n^1 & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

↳ $(n+1) \times (n+1)$ Vandemonde matrix

invertible if $x_0, x_1, \cdots, x_n$ are distinct.

Thm Let $x_0, \cdots, x_n \in \mathbb{R}$ be distinct, and $y_0, \cdots, y_n \in \mathbb{R}$. Then

$\exists!$ polynomial $p(x)$ w/ deg $\leq n$ s.t. $p(x_i) = y_i$, $i = 0, 1, \cdots, n$.

Lagrange form of polynomial interpolation

Define $l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$ $\quad i = 0, 1, \cdots, n$. Then $l_i(x)$ satisfies

$$l_i(x_k) = \delta_{ik} = \begin{cases} 1 & k = i \\ 0 & k \neq i \end{cases}$$

Then the polynomial interpolation $p(x)$ is

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

(because ① $\deg p \leq n$ ② $p(x_k) = \sum_{i=0}^{n} y_i \, \delta_{ik} = y_k$, $k = 0, \cdots, n$)

Newton form of poly. interp.

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1})$$

$p(x_0) = y_0 \Rightarrow c_0 = y_0$

$p(x_1) = y_1 \Rightarrow c_0 + c_1(x_1 - x_0) = y_1 \Rightarrow c_1 = \dfrac{y_1 - c_0}{x_1 - x_0}$

$p(x_2) = y_2 \Rightarrow c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2$

$$\Rightarrow c_2 = \frac{y_2 - (c_0 + c_1(x_2 - x_0))}{(x_2 - x_0)(x_2 - x_1)}$$

$\cdots$

$p(x_k) = y_k \Rightarrow c_0 + c_1(x_k - x_0) + \cdots + c_k(x_k - x_0)\cdots(x_k - x_{k-1}) = y_k$

$$\Rightarrow c_k = \frac{y_k - p_{k-1}(x_k)}{(x_k - x_0)\cdots(x_k - x_{k-1})}$$

($p_{k-1}(x_k)$ involves $c_0, \cdots, c_{k-1}$)

($p_k(x) := c_0 + c_1(x - x_0) + \cdots + c_k(x - x_0)\cdots(x - x_{k-1})$)

- Newton form is computationally efficient : suppose $c_0, \cdots, c_n$ are already computed. Then, to evaluate $p(x)$,

$$p(x) = c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + \cdots + c_n(x-x_0)\cdots(x-x_{n-1})$$

Horner algorithm:

$$\left.\begin{aligned} u_n &= c_n(x-x_{n-1}) + c_{n-1} \\ u_{n-1} &= u_n(x-x_{n-2}) + c_{n-2} \\ &\vdots \\ u_1 &= u_2(x-x_0) + c_0 = p(x) \end{aligned}\right\} \text{ total : } n \text{ multiplications.}$$

(in Lagrange form, evaluating $p(x)$ costs $O(n^2)$)

- Direct calculation of $c_0, \cdots, c_n$

$$\sum_{k=0}^{n} \Big((k-1) + 1 + k \Big) = 2 \cdot \tfrac{1}{2} n(n+1) \approx n^2 \quad \text{multiplications/divisions.}$$

- Use <u>divided difference</u> (sec. 6.2)

c_k only depends on $x_0, \cdots, x_k, y_0, \cdots, y_n$. We may denote

$$c_k := f[x_0, \cdots, x_k] \quad \text{where } f \text{ denotes a function w/ } f(x_i) = y_i.$$

↳ called <u>divided difference</u>

$$f[x_0] = f(x_0)$$

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

$$f[x_0, x_1, x_2] = \frac{f(x_2) - \left(f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) \right)}{(x_2 - x_0)(x_2 - x_1)}$$

<u>Thm</u> $f[x_0, x_1, \cdots, x_n] = \dfrac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0}.$

Proof $f[x_0, \cdots, x_n]$ is the leading coefficient in the poly. interp for $(x_i, f(x_i))$, $i = 0, \cdots, n$.

Let p_k denote the poly. interp. at $x_0, \cdots, x_k$, q denote the poly. interp. at $x_1, \cdots, x_n$.

Claim: $p_n(x) = \underbrace{q(x) + \frac{x - x_n}{x_n - x_0}(q(x) - p_{n-1}(x))}_{:= P(x)}$

(deg = n-1, deg = 1, deg n-1, deg n-1)

To prove the claim,

① $\deg P \le n$

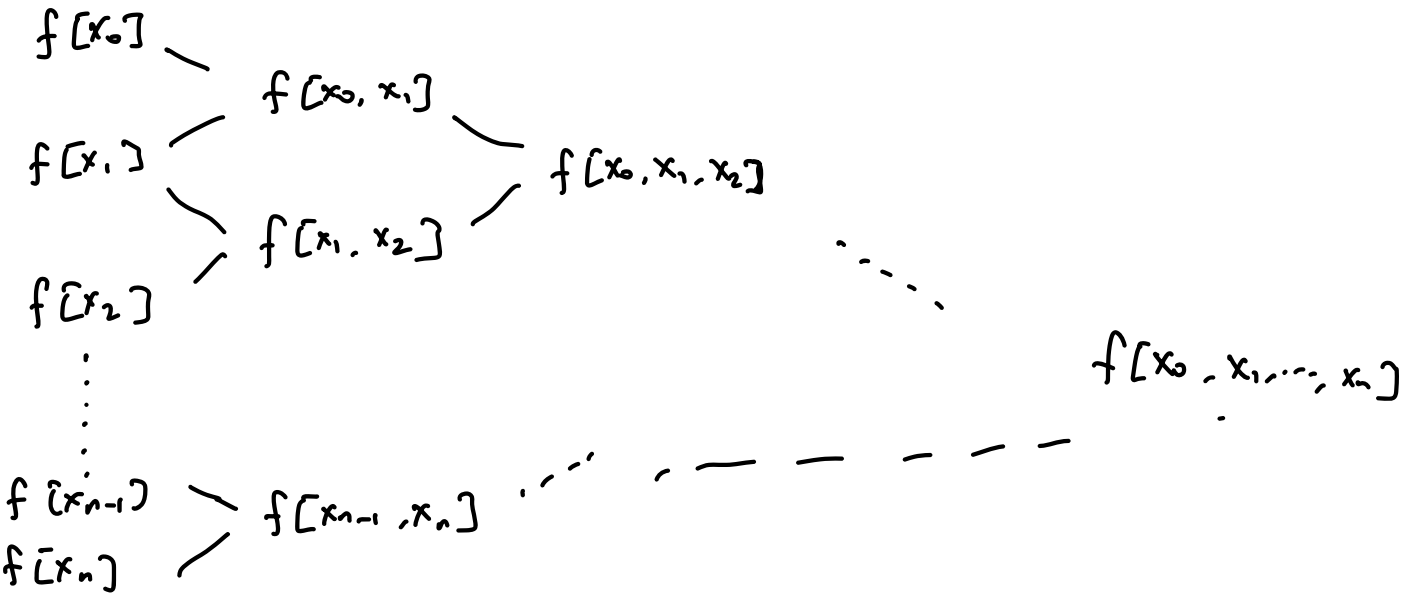
② Check $P(x_i) = f(x_i)$ $\quad i = 0, \cdots, n$

$$\begin{cases} \text{for } i = 1, \cdots, n-1, \quad P(x_i) = f(x_i) + \frac{x_i - x_n}{x_n - x_0}(f(x_i) - f(x_i)) = f(x_i) \\ \text{for } i = 0, \quad P(x_0) = q(x_0) + \frac{x_0 - x_n}{x_n - x_0}(q(x_0) - f(x_0)) = f(x_0) \\ \text{for } i = n, \quad P(x_n) = f(x_n) + \frac{x_n - x_n}{x_n - x_0}(q(x_n) - p_{n-1}(x_n)) = f(x_n) \end{cases}$$

Compare deg-n coeff. in Claim:

$$f[x_0, \cdots, x_n] = \frac{1}{x_n - x_0}(f[x_1, \cdots, x_n] - f[x_0, \cdots, x_{n-1}])$$

Algorithm for divided differences



Total cost (divisions): $n+(n-1)+\cdots+1 = \frac{1}{2}n(n+1) \approx \frac{1}{2}n^2$

Cost of evaluating $p(x)$ w/ Lagrange form:

$$p(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}$$

One can pre-calculate $y_i \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{1}{x_i-x_j} := b_i \quad i=0,\cdots,n$ (costs $O(n^2)$)

Then $p(x) = \sum_{i=0}^{n} b_i \prod_{\substack{j=0 \\ j\neq i}}^{n} (x-x_j) = \prod_{j=0}^{n}(x-x_j)\cdot\sum_{i=0}^{n}\frac{b_i}{x-x_i}$

Using this form, evaluating $p(x)$ costs $\approx 2n$ multiplication/division

(still 2 times as expensive as Newton form)