

## Polynomial Interpolation (PI)

**Problem:**

Given  $n + 1$  points  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ , where  $x_i$  are distinct, seek a polynomial  $p(x)$  with least degree such that  $p(x_i) = y_i$  for  $i = 0, 1, \dots, n$ , i.e., the polynomial curve passes through the given points.

Here  $x_i$  are called *nodes*, and  $p$  is said to *interpolate* the  $n + 1$  points.

**The Vandermonde Approach**

**Theorem.** If nodes  $x_0, x_1, \dots, x_n$  are distinct, then for arbitrary real values  $y_0, y_1, \dots, y_n$ , there is a *unique* polynomial  $p$  of degree  $\leq n$  such that  $p(x_i) = y_i$  for  $i = 0, 1, \dots, n$ .

**Proof.** Let  $p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$ , where  $c_i$  are to be determined. Set  $p(x_i) = y_i$ , then

$$\begin{aligned} c_0 + c_1x_0 + c_2x_0^2 + \dots + c_nx_0^n &= y_0 \\ c_0 + c_1x_1 + c_2x_1^2 + \dots + c_nx_1^n &= y_1 \\ &\dots\dots\dots \\ c_0 + c_1x_n + c_2x_n^2 + \dots + c_nx_n^n &= y_n \end{aligned}$$

Write this as a matrix-vector form  $Ac = y$ :

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdot & x_0^n \\ 1 & x_1 & x_1^2 & \cdot & x_1^n \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_n & x_n^2 & \cdot & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdot \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdot \\ y_n \end{bmatrix},$$

where  $A$  is called the Vandermonde matrix and it be shown (check a linear algebra book) that

$$\det(A) = \prod_{0 \leq i < j \leq n} (x_j - x_i) \neq 0.$$

Thus  $A$  is nonsingular and  $Ac = y$  has a unique solution  $c = A^{-1}y$ . ‡

The proof provides a method to compute the coefficients of the interpolating polynomial:

Step 1: Form the linear system  $Ac = y$ .

Step 2: Solve  $Ac = y$  by GEPP.

**Computational cost:** In step 1,  $A(:, j + 1) = A(:, j) \cdot \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ x_n \end{bmatrix}$  for  $j = 2, \dots, n$ .

It needs  $(n + 1)(n - 1) \approx n^2$  flops. In step 2,  $\frac{2}{3}n^3$  flops are required. Total:  $\frac{2}{3}n^3$  flops. Note:  $A$  has structures and actually faster algorithms are available to solve  $Ac = y$ .

### Evaluating $p(x)$ :

Nested multiplication

$$\begin{aligned} p(x) &= c_0 + c_1x + c_2x^2 + \dots + c_nx^n \\ &= c_0 + x(c_1 + x(c_2 + \dots + x(c_{n-1} + xc_n))) \dots \end{aligned}$$

Procedure for evaluating  $p(x)$  for some  $x$ :

```
p ← cn
for i = n - 1 : -1 : 0
    p ← ci + xp
end
```

**Computational cost:**  $2n$  flops.

### The Lagrange Approach

Lagrange form (LF):

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

where

$$l_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}, \quad l_i(x_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

Obviously  $p(x)$  defined above is a polynomial of degree less than or equal to  $n$  and  $p(x_i) = y_i$  for  $i = 0, 1, \dots, n$ .

We can rewrite  $p(x)$ :

$$p(x) = \sum_{i=0}^n l_i(x)y_i = \sum_{i=0}^n \frac{y_i}{\prod_{j=0, j \neq i}^n (x_i - x_j)} \cdot \frac{\prod_{j=0}^n (x - x_j)}{x - x_i} = q(x) \sum_{i=0}^n \frac{c_i}{x - x_i}$$

where  $c_i \equiv \frac{y_i}{\prod_{j=0, j \neq i}^n (x_i - x_j)}$ ,  $q(x) \equiv \prod_{j=0}^n (x - x_j)$ .

**Computational cost** of finding  $c_0, c_1, \dots, c_n$ :

For each  $i$ , computing  $c_i$  needs 1 division,  $n$  subtractions,  $n - 1$  multiplications, a total of  $2n$  flops. So computing all  $c_i$  needs  $2n * (n + 1) \approx 2n^2$  flops.

**Computational cost** of evaluating  $p(x)$  for some  $x$  (given  $c_i$  for  $i = 0, \dots, n$ ):

Computing  $q(x)$  needs  $2n + 1$  flops. Computing  $\frac{c_i}{x - x_i}$  needs 2 flops for each  $i$ . Thus computing  $p(x)$  needs a total of  $(2n + 1) + (n + 1) * 2 + n \approx 5n$  flops.

Note: In practice we usually do not use the Lagrange approach, since the evaluation of  $p(x)$  is not efficient.

### The Newton Approach

Idea: Suppose a polynomial  $p_k(x)$  of degree  $\leq k$  has been found to interpolate  $(x_0, y_0), (x_1, y_1), \dots, (x_k, y_k)$ . We seek a polynomial  $p_{k+1}(x)$  of degree  $\leq k + 1$  to interpolate  $(x_0, y_0), (x_1, y_1), \dots, (x_k, y_k), (x_{k+1}, y_{k+1})$ .

Let  $p_{k+1}(x) = p_k(x) + a_{k+1}(x - x_0)(x - x_1) \dots (x - x_k)$ , where  $a_{k+1}$  is to be determined. Obviously we have

$$p_{k+1}(x_i) = p_k(x_i) = y_i, \quad 0 \leq i \leq k.$$

Set  $p_{k+1}(x_{k+1}) = y_{k+1}$ , we obtain

$$a_{k+1} = \frac{y_{k+1} - p_k(x_{k+1})}{(x_{k+1} - x_0)(x_{k+1} - x_1) \dots (x_{k+1} - x_k)}.$$

This  $p_{k+1}(x)$  with the above  $a_{k+1}$  interpolates  $(x_0, y_0), \dots, (x_{k+1}, y_{k+1})$ . Also notice  $p_{k+1}(x)$  is a polynomial of degree  $\leq k + 1$ . So it is what we seek.

Finally  $p_n$  has the so-called Newton form (NF):

$$\begin{aligned} p_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots \\ &\quad + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}). \end{aligned}$$

### **Evaluating $p_n(x)$ for some $x$ :**

Nested multiplication

$$\begin{aligned} p_n(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \\ &= (\dots ((a_n(x - x_{n-1}) + a_{n-1})(x - x_{n-2}) + a_{n-2}) \dots)(x - x_0) + a_0 \end{aligned}$$

Procedure for evaluating  $p_n(x)$ :

```
p ← an
for i = n - 1 : -1 : 0
    p ← p * (x - xi) + ai
end
```

**Computational cost of this procedure:**  $3n$  flops.

**Cost of computing  $a_1, a_2, \dots, a_n$  by**

$$a_{k+1} = \frac{y_{k+1} - p_k(x_{k+1})}{(x_{k+1} - x_0)(x_{k+1} - x_1) \dots (x_{k+1} - x_k)} :$$

Cost of computing  $a_{k+1}$ :

$(1 + 3k) + [(k + 1) + k] + 1 = 5k + 3$  flops.

Total cost:  $\sum_{k=0}^{n-1} (5k + 3) = \frac{5}{2}n^2 + \frac{1}{2}n \approx \frac{5}{2}n^2$  flops.



**Remark:** The computation can be performed in a table (an example will be given in class).

**Computational cost:**  $\sum_{k=0}^{n-1} 3(n-k) = \frac{3}{2}n(n+1) \approx \frac{3}{2}n^2$  flops.

### Dangers of High Degree Polynomial Interpolation

Let  $y = f(x)$ . We approximate  $f$  on  $[a, b]$  by an interpolating polynomial  $p$  at  $n + 1$  nodes, i.e.,

$$p(x_i) = y_i = f(x_i).$$

**Q.** Is it true that  $f$  will be well approximate at all intermediate points as the number of nodes increases?

Answer: No. The Runge function:

$$f(x) = 1/(1 + 25x^2), \quad x \in [-1, 1].$$

If  $p_n$  is the polynomial that interpolates the  $f$  at  $n + 1$  equally spaced points on  $[-1, 1]$ , then

$$\lim_{n \rightarrow \infty} \max_{-1 \leq x \leq 1} |f(x) - p_n(x)| = \infty.$$

We could choose better nodes. But in general high-degree polynomial interpolation should be avoided.

**Interpolation error theorem:** If  $p$  is the polynomial of degree at most  $n$  that interpolates  $f$  at the  $n + 1$  distinct nodes  $x_0, x_1, \dots, x_n$  belonging to  $[a, b]$  and if  $f^{(n+1)}$  is continuous. Then for any  $x$  in  $[a, b]$ , there is  $z_x$  in  $(a, b)$  for which

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(z_x) \prod_{i=0}^n (x - x_i).$$