In the previous lecture we addressed the problem of interpolation: We assumed that we only know a certain function $f(x)$ through samples of its values $y_1 = f(x_1)$, $y_2 = f(x_2)$, $\ldots$, $y_N = f(x_N)$ at the N locations $x_1, x_2, \ldots, x_N$.

We want to reconstruct (or guess) a function $f(x)$, defined for arbitrary values of x (beyond the ones at $x_1, \ldots, x_N$). Knowing such a function enables us to:

- Estimate values of f in between the samples $\{x_i\}$.
- Make predictions for the value of even beyond the range spanned by the $\{x_i\}$ (i.e, beyond the maximum x_i, or below the minimum one).
- Make an estimate for the derivative(s) of f at arbitrary locations

Polynomial interpolation performs this task by passing an appropriate polynomial through data points $(x_1, y_1) \ldots (x_N, y_N)$. In particular, we choose to employe a degree-n polynomial

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

when we have $(n+1)$ data points

$$(x_1, y_1), (x_2, y_2), \dots, (x_{n+1}, y_{n+1})$$

to interpolate through. This choice is made in order to ensure existence (such a polynomial exists) and uniqueness (only one such polynomial exists). Let's examine these properties more closely:

Existence: We will later show that, with a degree-n polynomial it is always possible to match $n+1$ data points. For now, we can at least show that such a task would have been impossible (in general) if we were only allowed to use degree-$(n-1)$ polynomials. In fact, consider the points

$$(x_1, y_1=0), (x_2, y_2=0), \dots, (x_n, y_n=0) \text{ and } (x_{n+1}, y_{n+1}=1).$$

Thus, if a $(n-1)$-degree polynomial was able to interpolate these points, we would have:

$$P_{n-1}(x_1) = P_{n-1}(x_2) = \dots = P_{n-1}(x_n) = 0.$$

P_{n-1} can only equal zero at exactly $n-1$ locations (max) unless it is the zero polynomial $P_{n-1}(x) \equiv 0$. This is a contradiction as $P_{n-1}(x_{n+1}) \neq 0$.

<u>Uniqueness</u>: Assume that both $P_n(x)$ & $Q_n(x)$ interpolate all data points $(x_1, y_1), \ldots, (x_{n+1}, y_{n+1})$. Thus

$$P_n(x_1) = Q_n(x_1) \ (= y_1), \ \ldots\ldots, \ P_n(x_{n+1}) = Q_n(x_{n+1}) \ (= y_{n+1})$$

The polynomial $R_n(x) = P_n(x) - Q_n(x)$ thus satisfies

$$R_n(x_i) = 0 \qquad \forall i \in \{1, 2, \ldots, n+1\}.$$

R_n is of degree n, thus it can only be zero at n points <u>unless</u> (which is the case here) $R_n(x) \equiv 0$, i.e. $P_n(x) \equiv Q_n(x)$.

The most basic procedure to determine the coefficients $a_0, a_1, \ldots, a_n$ of the interpolating polynomial $P_n(x)$, is to write a linear system of equations as follows:

$$\left.\begin{aligned} P_n(x_1) &= y_1 \\ P_n(x_2) &= y_2 \\ &\vdots \\ P_n(x_n) &= y_n \\ P_n(x_{n+1}) &= y_{n+1} \end{aligned}\right\} \Rightarrow \begin{aligned} a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1} + a_n x_1^n &= y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_{n-1} x_2^{n-1} + a_n x_2^n &= y_2 \\ &\vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_{n-1} x_n^{n-1} + a_n x_n^n &= y_n \\ a_0 + a_1 x_{n+1} + a_2 x_{n+1}^2 + \cdots + a_{n-1} x_{n+1}^{n-1} + a_n x_{n+1}^n &= y_{n+1} \end{aligned}$$

or, in matrix form:

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} & x_2^n \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & x_n^n \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^{n-1} & x_{n+1}^n \end{bmatrix}}_{\substack{(n+1)\times(n+1) \text{ matrix} \\ V}} \cdot \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix}}_{\substack{(n+1)\text{-vector} \\ \underset{\sim}{a}}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \end{bmatrix}}_{\substack{(n+1)\text{-vector} \\ \underset{\sim}{y}}}$$

The matrix V is called a Vandermonde matrix. We will see that V is non-singular, thus we can solve the system $V\underset{\sim}{a} = \underset{\sim}{y}$ to obtain the coefficients $\underset{\sim}{a} = (a_0, a_1, \ldots, a_n)$. Let's evaluate the merits and drawbacks of this approach:

- Cost to determine the polynomial $P_n(x)$: VERY COSTLY, since a dense $(n+1)\times(n+1)$ linear system has to be solved. This will generally require time proportional to n^3, making large interpolation problems intractable. In addition, the Vandermonde matrix is notorious for being challenging to solve (esp. with Gauss elimination) and prone to large errors in the computed coefficients $\{a_i\}$, when n is large and/or $x_i \approx x_j$.

- Cost to evaluate $f(x)$ (x = arbitrary) if coefficients are known : VERY CHEAP. Using Horner's scheme:

$$a_0 + a_1 x + \cdots + a_n x^n = a_0 + x\left(a_1 + x\left(a_2 + \cdots + x\left(a_{n-1} + x a_n\right)\right)\right)$$

- Availability of derivatives : VERY EASY. e.g.

$$P_n'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \cdots + (n-1)a_{n-1}x^{n-2} + n a_n x^{n-1}$$

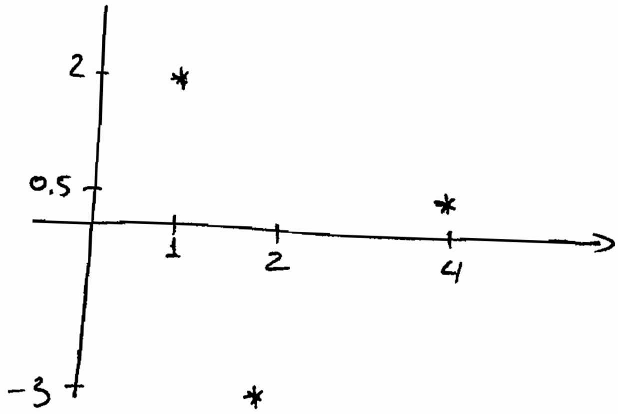
- Allows incremental interpolation : NO!

This property examines if interpolating through $(x_1,y_1)\ldots,(x_{n+1},y_{n+1})$ is <u>easier</u> if we already know a polynomial (of degree $= n-1$) that interpolates through $(x_1,y_1),\ldots,(x_n,y_n)$. In our case the system $V\underline{a} = \underline{y}$ would have to be solved from scratch for the $(n+1)$ data points.

<u>Lagrange interpolation</u> (§4.3) is an alternative way to define $P_n(x)$, without having to solve expensive systems of equations.

We shall explain how Lagrange interpolation works, with an example.

Example: Pass a ~~cubic~~ quadratic polynomial through $(1,2)$, $(2,-3)$, $(4,0.5)$. 2|8|2011 L

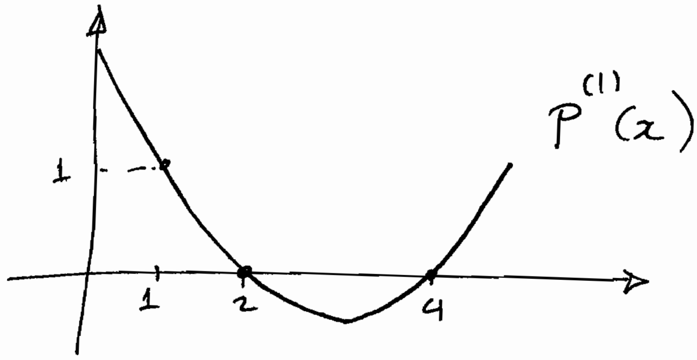


Assume we have somehow constructed 3 ~~cubic~~ quadratic polynomials $P^{(1)}(x)$, $P^{(2)}(x)$, $P^{(3)}(x)$, such that:

$P^{(1)}(1) = 1$

$P^{(1)}(2) = 0$

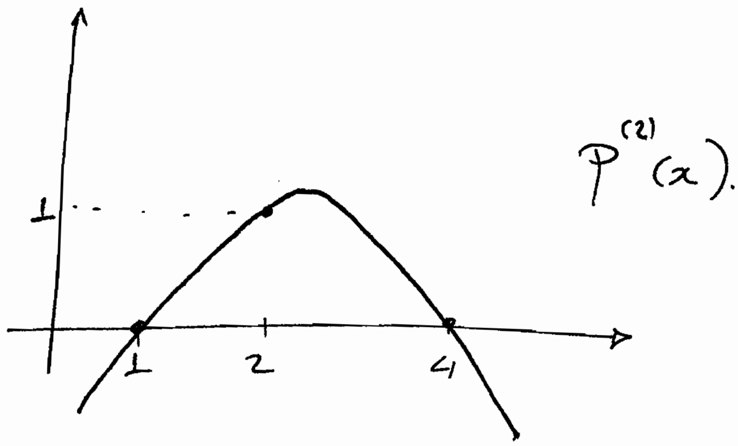
$P^{(1)}(4) = 0$



$P^{(2)}(1) = 0$

$P^{(2)}(2) = 1$

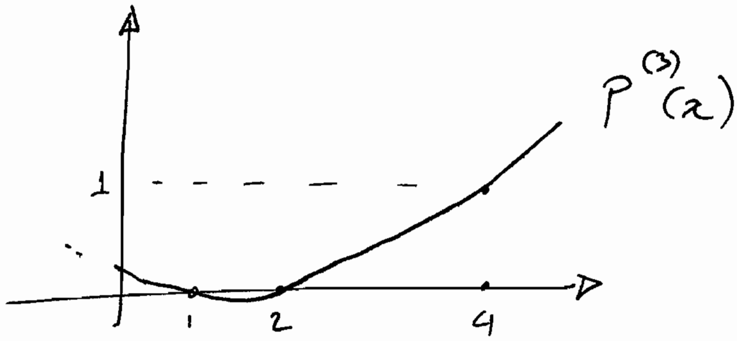
$P^{(2)}(4) = 0$



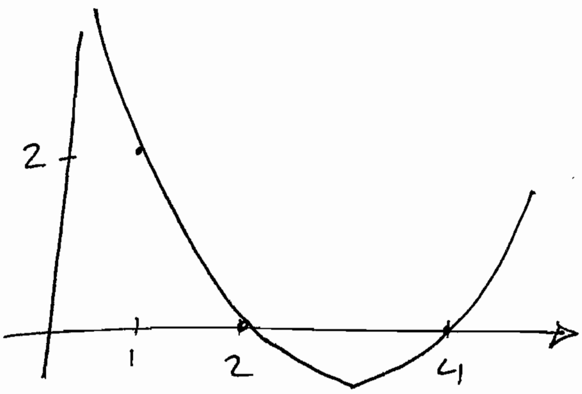
$P^{(3)}(1) = 0$

$P^{(3)}(2) = 0$

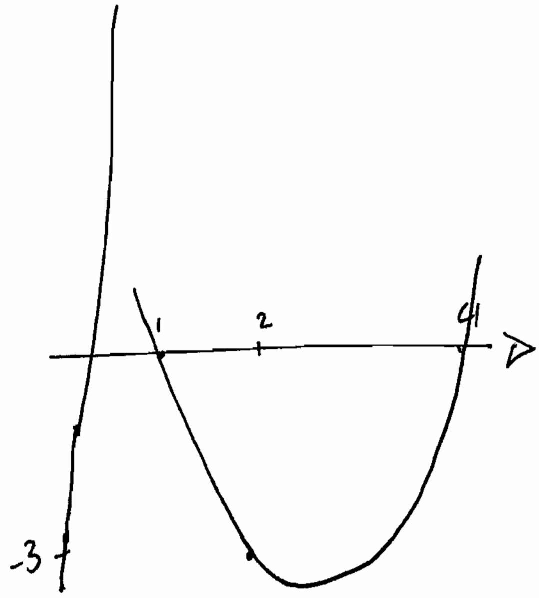
$P^{(3)}(4) = 1$



Now the idea is to scale each $P^{(i)}$, such that $P^{(i)}(x_i) = y_i$ and add them all together

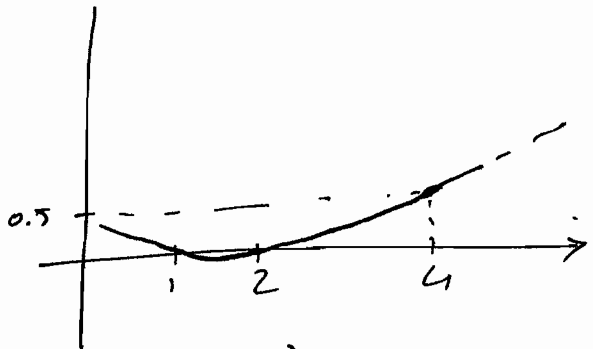


$2 \times P^{(1)}(x)$ +



$-3 \times P^{(2)}(x)$

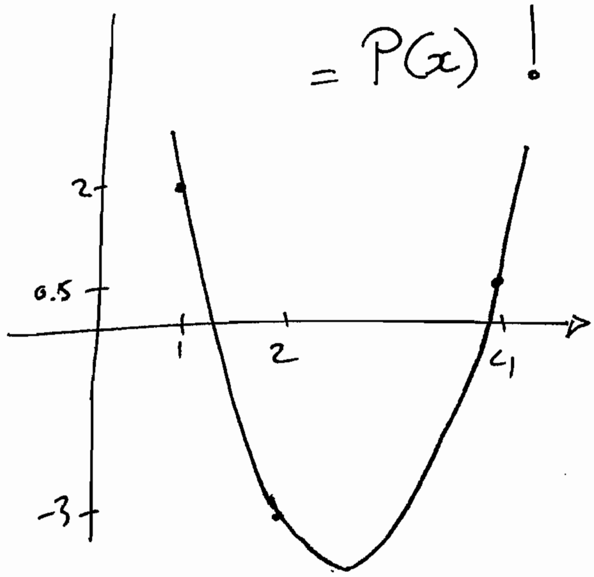
+



$0.5 \times P^{(3)}(x)$

=

$= P(x)$!



In summary, if we have a total of $(n+1)$ data points $(x_0,y_0), (x_1,y_1), \dots, (x_n,y_n)$, define the Lagrange polynomials of n-degree $l_0(x), l_1(x), \dots, l_n(x)$ as:

$$l_i(x_j) = \begin{cases} 1, & \text{if } i=j \\ 0, & \text{if } i\neq j. \end{cases}$$

Then, the interpolating polynomial is simply

$$P(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) = \sum_{i=0}^{n} y_i l_i(x).$$

No solution of a linear system is necessary here. We just have to explain what every $l_i(x)$ looks like...

Since $l_i(x)$ is an n-degree polynomial, with the n-roots $x_0, x_2, x_2, \dots, x_{i-1}, x_{i+1}, x_{i+2}, \dots, x_n$, it must have the form

$$l_i(x) = \underset{\substack{\shortparallel \\ \text{const}}}{C_i}(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)$$

$$= C_i \prod_{j\neq i}(x-x_j)$$

Now, we require $l_i(x_i)=1$, thus: $1 = C_i \prod_{j\neq i}(x_i-x_j)$

$$\Rightarrow C_i = \frac{1}{\prod_{j\neq i}(x_i-x_j)}$$

Thus, for every i, we have

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

$$= \prod_{j\neq i}\left(\frac{x-x_j}{x_i-x_j}\right) = \frac{\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i}(x_i-x_j)}$$

<u>Note</u> This result essentially proves <u>existence</u> of a polynomial interpolant of degree $=n$ that passes through $(n+1)$ data points. We can also use it to prove that the Vandermonde matrix V is non-singular; if it <u>were</u> singular, a right-hand-side $y = (y_0, \ldots, y_n)$ would have existed such that $V\underset{\sim}{a} = y$ would have no solution, which is a contradiction

Let's evaluate the same 4 quality metrics we saw before, for the Vandermonde matrix approach.

- Cost of determining $P(x)$: VERY EASY, we are essentially able to write a formula for $P(x)$ without solving any systems.
  However, if we want to write $P(x) = a_0 + a_1 x + \cdots + a_n x^n$, the cost of evaluating the a_i's would be very high! each l_i would need to be expanded $\Rightarrow$ approx. N^2 operations for each l_i; N^3 operations for $P(x)$.

- Cost of evaluating $P(x)$ [x: arbitrary]. SIGNIFICANT.

We do not really need to compute the a_i's beforehand, if we only need to evaluate $P(x)$ at select few locations. for each $l_i(x)$ the evaluation requires N subtraction & N multiplications $\Rightarrow$ total = about N^2 operations (better than N^3 for computing the a_i's).

- Availability of derivatives: NOT READILY AVAILABLE

Differentiating each l_i (since $P'(x) = \sum y_i l_i'(x)$) is not trivial $\Rightarrow$ yields N terms each, with $(N-1)$ products per term.

- Incremental interpolation. NOT ACCOMMODATED.

Still, Lagrange interpolation is a good quality method, if we can accept its limitations.

Newton interpolation is yet another alternative, which enables both efficient evaluation and allows for incremental construction. Additionally (as we will see in next lecture) it allows both the coefficients $\{a_i\}$ as well as the derivative $P'(x)$ to be evaluated efficiently.