Thm (Horner's Method)

Let
$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Define $b_n = a_n$

$$b_k = a_k + b_{k+1} x_0, \quad k = n-1, n-2, \ldots, 1, 0$$

Then $\boxed{b_0 = p(x_0)}$

Moreover, if $Q(x) = b_n x^{n-1} + b_{n-1} x^{n-2} + \cdots + b_2 x + b_1$,

then $\boxed{p(x) = (x - x_0) Q(x) + b_0}$

Pf: $(x - x_0) \cdot Q(x) + b_0 = (x - x_0)\left[b_n x^{n-1} + \cdots + b_2 x + b_1\right] + b_0$

$$= \left[b_n x^n + b_{n-1} x^{n-1} + \cdots + b_2 x^2 + b_1 x\right] - \left[b_n x_0 x^{n-1} + \cdots + b_2 x_0 x + b_1 x_0\right] + b_0$$

$$= b_n x^n + (b_{n-1} - b_n x_0) x^{n-1} + \cdots + (b_1 - b_2 x_0) x + (b_0 - b_1 x_0)$$

By hypothesis, $b_n = a_n$, $b_k - b_{k+1} x_0 = a_k$

$$\Rightarrow \ = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

$$= p(x)$$

$\Rightarrow p(x) = (x - x_0) Q(x) + b_0$ and $p(x_0) = b_0$ ∎

Remarks:

- Can use $b_k = a_k + b_{k+1} x_0$, $k = n-1, n-2, \dots, 1, 0$, $(b_n = a_n)$ to evaluate p at x_0 in nested manner.
- For P_n $(\deg(P_n) \le n)$, require at most n multiplications and n additions

Ex: Use Horner's method to evaluate

$$P(x) = 2x^4 - 3x^2 + 3x - 4 \quad \text{at} \quad x_0 = -2$$

$x_0 = -2$	$a_4 = 2$	$a_3 = 0$	$a_2 = -3$	$a_1 = 3$	$a_0 = -4$
		$b_4 \cdot x_0 = -4$	$b_3 \cdot x_0 = 8$	$b_2 x_0 = -10$	$b_1 x_0 = 14$
	$b_4 = 2$	$b_3 = -4$	$b_2 = 5$	$b_1 = -7$	$b_0 = 10$ $= P(-2)$

Remarks:

1) $P(x) = (((2x+0)x - 3)x + 3)x + 4) \leftarrow$ nested
4 mult.
3 adds

2) Let $p(x) = (x - x_0) Q(x) + b_0$. Then

$$p'(x) = Q(x) + (x - x_0) Q'(x) \implies p'(x_0) = Q(x_0)$$

This means we can obtain $p'(x_0)$ by applying Horner's method to $Q(x)$ (new iterates $c_k = b_k + c_{k+1} x_0$, $c_n = b_n$)

Here,
$$a_n \to b_n \to c_n$$
$$a_{n-1} \to b_{n-1} \to c_{n-1}$$
$$\vdots$$
$$a_1 \to b_1 \to c_1$$
$$a_0 \to b_0 \to c_0$$

$$\boxed{\begin{aligned} p(x_0) &= b_0 \\ p'(x_0) &= c_0 \end{aligned}}$$

3) If $f(x) = p(x)$ in Newton's, then we can use Horner's method (aka synthetic division) to compute $f(x_n)$ and $f'(x_n)$ more efficiently.

From implementation perspective:

$$[b_0, c_0] = \text{Horners}(\underline{a}, x_0) \qquad \underline{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

4) Can be interpreted as a (linear) neural network! (in 1D)

$$NN(x_0) = u_n(x_0) \quad \text{where} \quad u_{k+1} = \sigma(u_k x_0 + a_k)$$

nonlinear activation function

$$b_k = a_k + b_{k+1} x_0 \qquad \text{vs.} \qquad b_k = \sigma(a_k + b_{k+1} x_0)$$

2.6 Deflation: procedure to find all zeros of polynomials by successively applying Newton's method.

1) Choose initial guess $p_0^{(1)}$, and find approximate zero $\hat{x}_1$ of $P_n(x)$ with degree n using Newton's.

$$P_n(x) \approx (x - \hat{x}_1)\, Q_1(x)$$

2) Find zero of $Q_1(x)$ and get $\hat{x}_2$.
To find $\hat{x}_k$, choose initial guess $p_0^{(k)}$ and apply Newton's method to $Q_k(x)$

$$P_n(x) \approx (x-\hat{x}_1)(x-\hat{x}_2) \cdot Q_2(x) \qquad k=2$$

$$\vdots$$

$$P_n(x) \approx (x-\hat{x}_1)(x-\hat{x}_2) \cdots (x-\hat{x}_{n-2})\, Q_{n-2} \qquad k=n-2$$

$Q_{n-2}(x)$ quadratic $\Rightarrow$ use quadratic formula.

3) Obtain refined solutions $x_1, x_2, \ldots, x_n$ by applying Newton's method to $P_n(x)$ with corresponding initial guesses $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n$, respectively.

Remark: Inaccuracy increases as k increases in step 2
Reduce error via step 3.

3.1 Interpolation

Q: Are polynomials a "good" set of functions to approximate / interpolate arbitrary function f?

Thm 1 (Weierstrass Approximation Thm)

Suppose $f \in C[a,b]$

For any $\varepsilon > 0$, there exists a polynomial $p(x)$ s.t.

$$|f(x) - p(x)| < \varepsilon, \quad \forall \ x \in [a,b]$$

Remark: derivative and integrals are easy to compute

$\Rightarrow$ often used to approximate continuous functions.

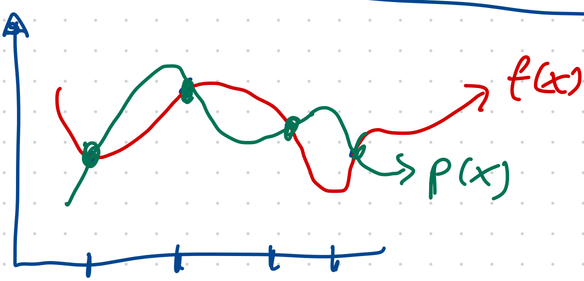
Ex: Let $f(x) = e^x$. Taylor expansion about $x_0 = 0$ yields

$$f(x) = \underbrace{1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}}_{P_n(x)} + \underbrace{\frac{e^{\xi} x^{n+1}}{(n+1)!}}_{R(x)} \quad \xi \text{ between } 0 \text{ and } x.$$

$\Rightarrow$ Can use $P_n(x)$ to approximate $f(x)$ with error R.

Thm 2 Note:

f only matches p at x_k's.



$$\boxed{\begin{array}{l}\underline{\text{Thm 2:}}\ \text{Given } (x_0, \underbrace{f_0}_{f(x_0)}), (x_1, \underbrace{f_1}_{f(x_1)}), \dots, (x_n, \underbrace{f_n}_{f(x_n)}) \\ \text{with } x_k\text{'s } \underline{\text{distinct}},\ k = 0, 1, \dots, n. \\ \text{Then a unique polynomial of degree at most } n \text{ exists} \\ \text{with} \\ \qquad f(x_k) = p(x_k) \quad \text{for each } k = 0, 1, \dots, n\end{array}}$$

Given $(x_0, f_0), (x_1, f_1), \dots, (x_n, f_n)$, how to construct polynomial to interpolate $f(x)$?

__Power Series Approach:__ let $p(x)$ have the form

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

$$\left.\begin{array}{l} p(x_0) = f_0 \\ p(x_1) = f_1 \\ \quad\vdots \\ p(x_n) = f_n \end{array}\right\} \quad \begin{array}{l} a_n x_0^n + a_{n-1} x_0^{n-1} + \dots + a_1 x_0 + a_0 = f_0 \\ a_n x_1^n + a_{n-1} x_1^{n-1} + \dots + a_1 x_1 + a_0 = f_1 \\ \\ a_n x_n^n + a_{n-1} x_n^{n-1} + \dots + a_1 x_n + a_0 = f_n \end{array} \quad (*)$$

$\boxed{x_n\text{'s distinct} \Rightarrow n+1 \text{ eqns}}$

Want to find coefficients a_i's, $i = 0, 1, \dots, n$.

This can be done by solving system $(*)$ consisting of $\underbrace{(n+1) \text{ equations and } n+1 \text{ variables } a_i\text{'s}}_{\text{true since } x_k\text{'s distinct}}$.

We can write (*) as linear system

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$V \underline{a} = \underline{f}$$

Vandermonde Matrix

In MATLAB

$$\underline{a} = V \backslash \underline{f}$$

Remarks: 1) V is often ill-conditioned

2) intuitive to build, but difficult to solve

will discuss more about numerical lin. alg. in Chpt 6.

A more popular approach/form to construct $p(x)$:

Lagrange Form

$$p(x) = f(x_0)\cdot L_{n,0}(x) + f(x_1)\cdot L_{n,1}(x) + \cdots + f(x_n)\cdot L_{n,n}(x)$$

Recall that we want $p(x_k) = f(x_k)$, $k = 0, 1, \dots, n$.

$\Rightarrow$ want

$$L_{n,k}(x) = \begin{cases} 1 & \text{if } x = x_k \\ 0 & \text{else} \end{cases} \qquad (**)$$

We can thus construct L as follows:

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

Note this satisfies $(**)$

$L_{n,k}$ is called a Lagrange interpolating polynomial.

Ex: a) Let $x_0 = 2$ $x_1 = 2.75$, $x_2 = 4$. find $P_2(x)$ for $f(x) = \frac{1}{x}$

b) use $P_2(x)$ to approximate $f(3) = \frac{1}{3}$

$$L_{2,0}(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-2.75)(x-4)}{(-0.75)(-2)}$$

$$L_{2,1}(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{-16}{15}(x-2)(x-4)$$

$$L_{2,2}(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{2}{5}(x-2)(x-2.75)$$

$$\Rightarrow P_2(x) = \sum_{i=0}^{2} L_{2,i}(x) \cdot f(x_i)$$