Summary QR - algorithm

A tridiagonal [through Householder or Givens]

$A^{(0)} = A$, for $k = 0, 1, 2, \ldots$

$$A^{(k)} - \mu^k I = Q^{(k)} R^{(k)} \qquad \text{QR factorization}$$

$$A^{(k+1)} := R^{(k)} Q^{(k)} + \mu^k I \quad \text{mult. in reverse order}$$

end

[shifted version, $\mu^k \in \mathbb{R}$ approx to eigenvalue]

$$A^{(k+1)} = Q^{(k)T} A^{(k)} Q^{(k)}$$

$$\Longrightarrow \underbrace{Q^{(k)T} \cdots Q^{(2)T} Q^{(1)T}}_{\bar{Q}^{(k)T}} A \underbrace{Q^{(1)} Q^{(2)} \cdots Q^{(k)}}_{\bar{Q}^{(k)}} \xrightarrow{k \to \infty} \text{diagonal matrix}$$

$\longrightarrow$ we find the eigenvalues, How about eigenvectors?

Start with $B \in \mathbb{R}^{n \times n}$, $B^T = B$

$\xrightarrow{\text{step 1}} P_n^T B P_n = A$ tridiagonal flops: $\sim n^3$

orthogonal, products of Householder (or Givens) matrices

$\xrightarrow[\text{QR-algo}]{\text{step 2}} \bar{Q}^{(k)T} A \bar{Q}^{(k)} \longrightarrow D$ diagonal flops: $\sim n^2$ per QR iteration (or $\sim n$??)

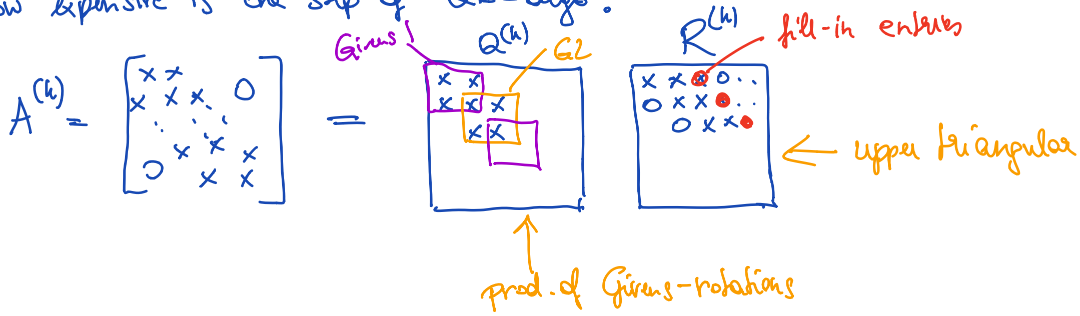
$\xrightarrow{\text{combining}} \bar{Q}^{(k)T} P_n^T B P_n \bar{Q}^{(k)} = D$

$$\longrightarrow B P_n \bar{Q}^{(k)} = P_n \bar{Q}^{(k)} D$$

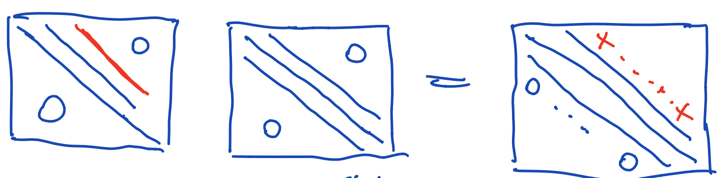
$$B \cdot \begin{bmatrix} | & | & | & \\ | & | & | & \\ | & | & | & \end{bmatrix} = \begin{bmatrix} | & | & | & \\ | & | & | & \\ | & | & | & \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \end{bmatrix}$$

$\Longrightarrow B v_i = \lambda_i v_i$, where v_i is the i-th column of $P_n \bar{Q}^{(k)}$

How expensive is one step of QR-algo?



$$A^{(k+1)} = Q^{(k)^T} A^{(k)} Q^{(k)} = R^{(k)} Q^{(k)} =$$



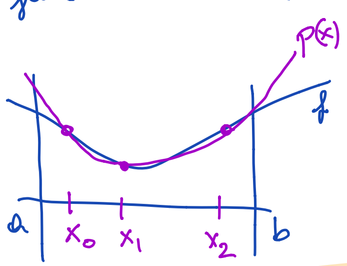
Since $A^{(k+1)}$ is symmetric, x⋯x must vanish and $A^{(k+1)}$ is tridiagonal.

Each QR iteration is cheap since we are only working with tridiagonal matrices!

---

§ 6.2 Polynomial Interpolation

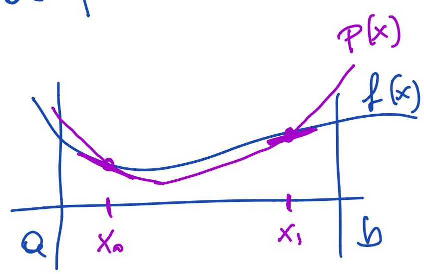
Approximate functions using polynomials that coincide with the function values (and/or derivatives) at node points.



$p(x)$ .. quadratic polynomial

$p(x_i) = f(x_i)$

Lagrange interpolation (only function values)



$p(x)$ .. 3^{rd} order polynomial

$p(x_i) = f(x_i) \quad i = 0, 1$

$p'(x_i) = f'(x_i) \quad i = 0, 1$

Hermite interpolation (function values and derivatives)

Interpolation problem:

$n \geq 1$, $x_0, \dots, x_n$ distinct ($x_i \neq x_j$ for $i \neq j$)

$y_0, \dots, y_n \in \mathbb{R}$; Find $p_n \in \mathbb{P}_n$ such that $p_n(x_i) = y_i \quad i = 0, \dots, n$

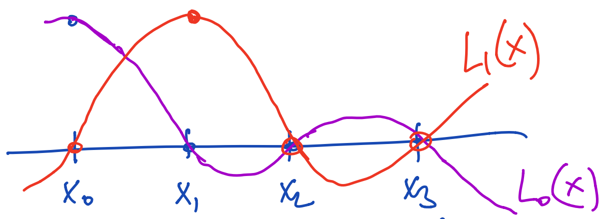
$\mathbb{P}_n$ ... space of polynomials of degree $\leq n$, i.e.

$$\mathbb{P}_n = \{ a_n x^n + \dots + a_1 x + a_0 \mid a_n, \dots, a_0 \in \mathbb{R} \}$$

Lemma: $n \geq 1$, There exist polynomials $L_k \in \mathbb{P}_n$ $k = 0, \dots, n$ such that

$$L_k(x_i) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases} \quad \text{(Lagrange polynomials)}$$

Moreover, $p_n(x) = \sum_{k=0}^{n} L_k(x) y_k$ is the interpolating polynomial



Proof: $L_k(x) = c_k \prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i)$, since $L_k(x_k) \overset{!}{=} 1$

$$\Longrightarrow c_k = \frac{1}{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)} \Longrightarrow L_k(x) = \frac{\overbrace{\prod_{i \neq k} (x - x_i)}^{\text{polynomial}}}{\underbrace{\prod_{i \neq k} (x_k - x_i)}_{\text{number}}}$$

$$p_n(x_i) = \sum_{k=0}^{n} \overbrace{L_k(x_i)}^{= 1 \text{ if } i = k \text{ and } 0 \text{ else}} y_k = y_i \quad \text{interpolating polynomial.}$$

$\square$