Eigenvalue Algorithms

Most of the eigenvalue problems proceed in two phases.

(i) A reduction of the given matrix to some special form

(ii) An iterative process that takes the special form to an eigenvalue-revealing form!

Note:-

* Obvious way in computing eigenvalues is to just find roots of characteristic polynomial! This is a very bad idea to solve eigenvalue problem!

* A Fundamental difficulty

$\underbrace{Ax = b}$

* Eigenvalue problems can be reduced to polynomial root-finding problem!.

Conversely any polynomial root finding

problem can be stated as an eigenvalue problem!

Consider the monic polynomial

$$p(z) = z^m + a_{m-1} z^{m-1} + \cdots + a_1 z + z_0$$

Roots of $p(z)$ are equal to eigenvalues of which matrix?

Such a matrix

$$\underline{A} = \begin{bmatrix} 0 & & & -a_0 \\ 1 & 0 & & -a_1 \\ & 1 & 0 & \vdots \\ & & \ddots & -a_{m-2} \\ & & 1 & -a_{m-1} \end{bmatrix}$$

$\underline{A}$ is a companion matrix to $p(z)$!

The difficulty → A formula for expressing the roots of an arbitrary polynomial does not exists given its coefficients!

Infact for a polynomial of degree $m \geq 5$, such a formula does not exists!

Remarks :-

* No algorithm can exactly produce all the roots of an arbitrary polynomial in a finite number of steps!

* We can say that we cannot design algorithms for computing eigenvalues to an arbitrary accuracy in a finite number of steps!

* **Any eigensolver must be iterative!**

* **Schur factorization and diagonalization!**

→ Most modern general purpose eigenvalue solvers compute Schur factorization of A by elementary unitary / orthogonal similarity transformation

$$X \rightarrow Q_j^{+} X Q_j$$

so that

$$\underbrace{Q_j^+ \cdots Q_2^+ Q_1^+ A Q_1 Q_2 \cdots Q_j}_{\text{as } j \to \infty} = T$$

If A is real and symmetric, 'T' is a diagonal matrix.

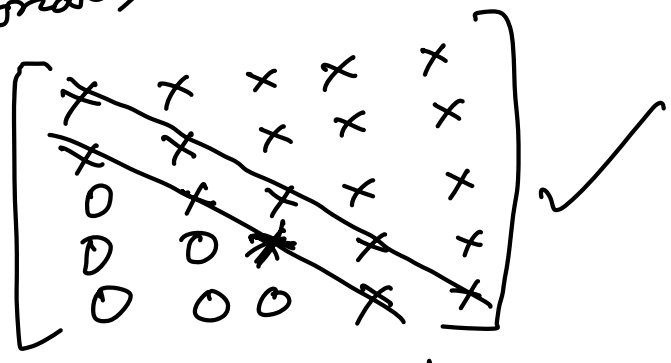
If A is real and not symmetric, then it still may have eigenvalues to be complex and Schur factorization will be complex.

Fortunately! it is possible to carry out entire factorization with only real arithmetic if A is real if we allow 2×2 blocks along the diagonal of T (Recall real Schur factorization)

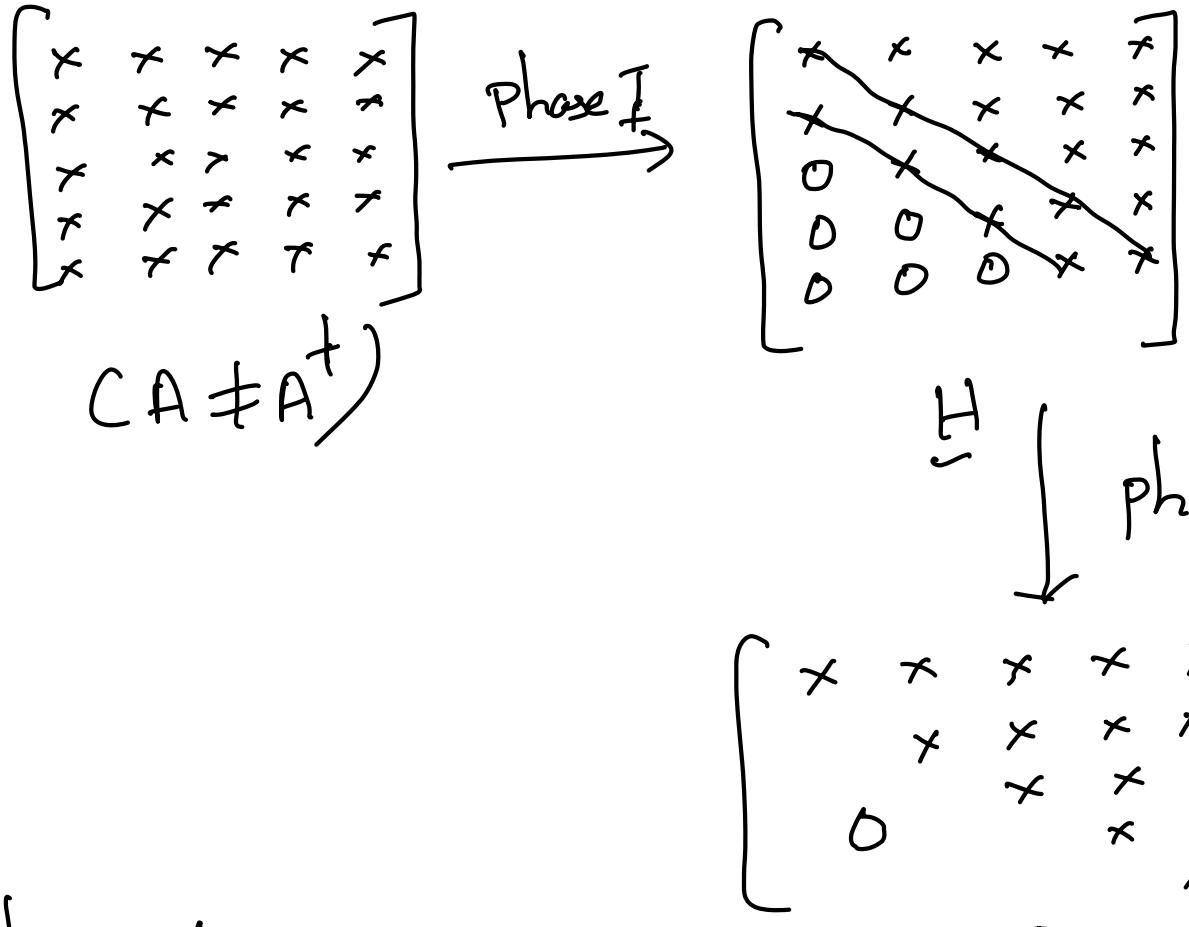
* The two phases of eigenvalue computations!

The construction of the Q_i matrices which reduces $\underline{A}$ to Schur form ($\underline{T}$ upper triangular) is split into two phases!

(i) A direct method to reduce $\underline{A}$ to an upper Hessenberg matrix $\underline{H}$ (i.e a matrix with zeros below the first subdiagonal)



(ii) An iterative method to produce a sequence of similarity transformations that reduces $\underline{H}$ to converge to upper triangular form!



Phase 1:- require $O(m^3)$ flops

Phase 2:- In principle could go on forever, but in practice we see convergence to machine precision within $O(m)$ iterations! Each iteration requires $O(m^2)$ flops so the total work for phase 2 is also $O(m^3)$

Without phase 1! Each iteration of phase 2 would cost $O(m^3)$ flops, the

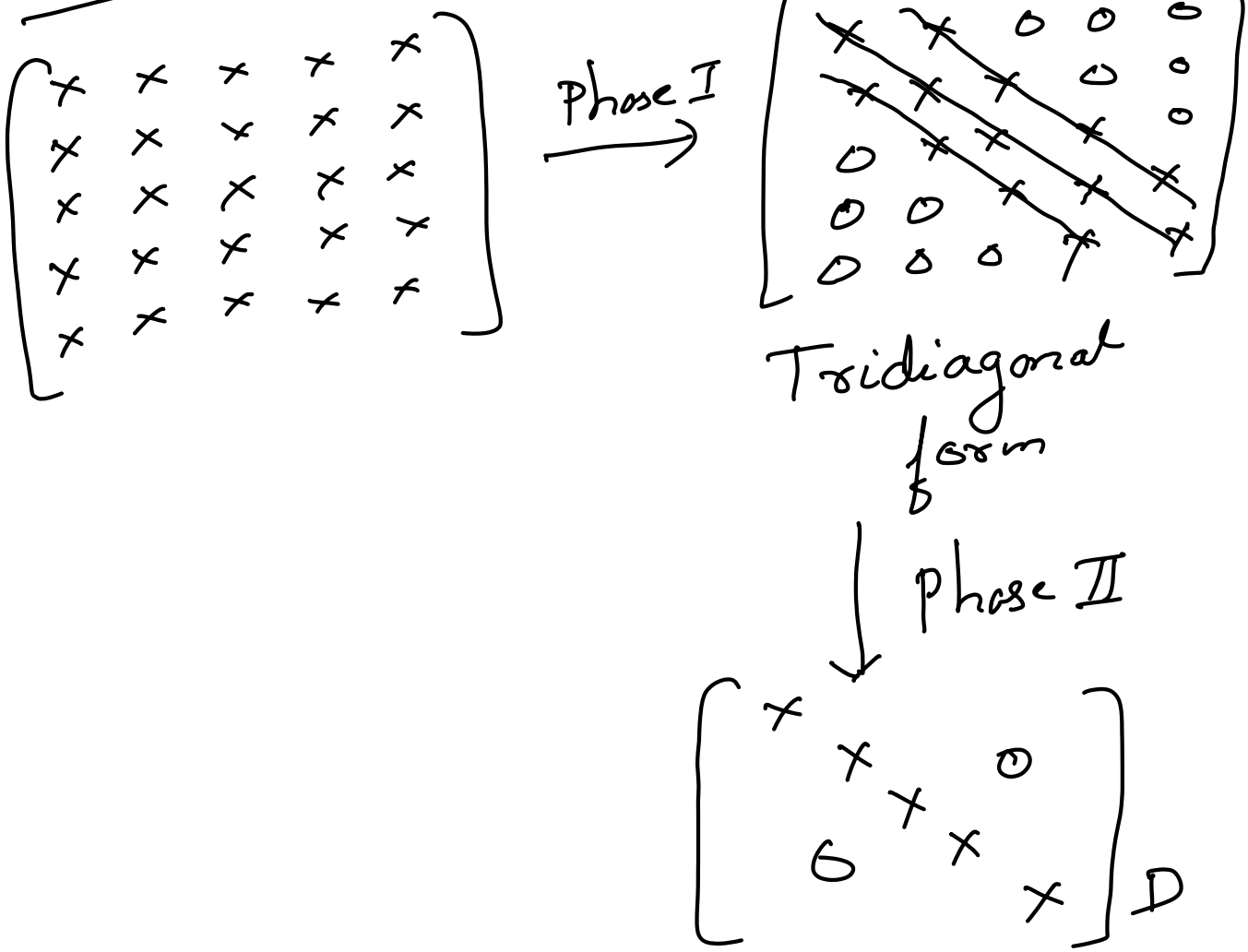
total cost to $O(m^4)$ flops!

Note:- For a symmetric matrix $A = A^T$,

phase I will reduce to tridigonal matrix and phase II will reduce to diagonal matrix

Schematically!

for $A = A^T$



Reduction to Upper Hessenberg form :-

* We want to apply orthogonal/unitary similarity transformations to $\underline{A}$ so as to introduce zeros below the diagonal!
* We can use the idea of Householder reflectors

Suppose we construct a Householder reflector $\underline{Q}_1^T$ acting on the left of $\underline{A}$, to introduce zeros below the diagonal in the first colum of $\underline{A}$

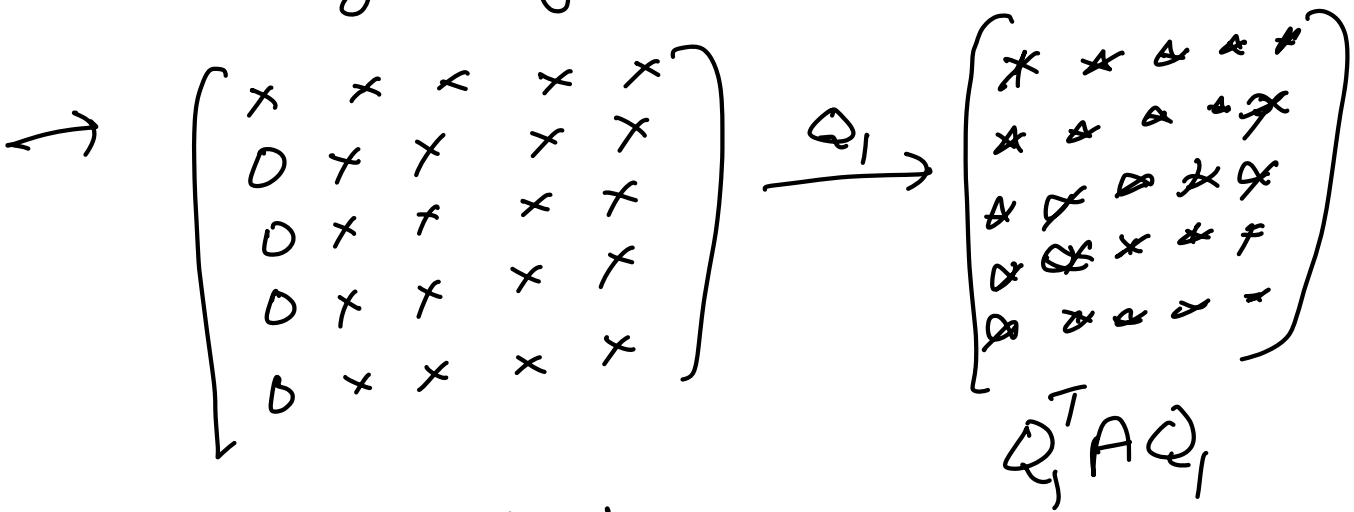
$$\underline{F}\underline{x} = \begin{bmatrix} \|\underline{x}\| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \xrightarrow{Q_1^T} \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

$\underline{Q}_1^T \underline{A}$

To do a similarity transformation, we must multiply $\underline{Q}_1^T \underline{A}$

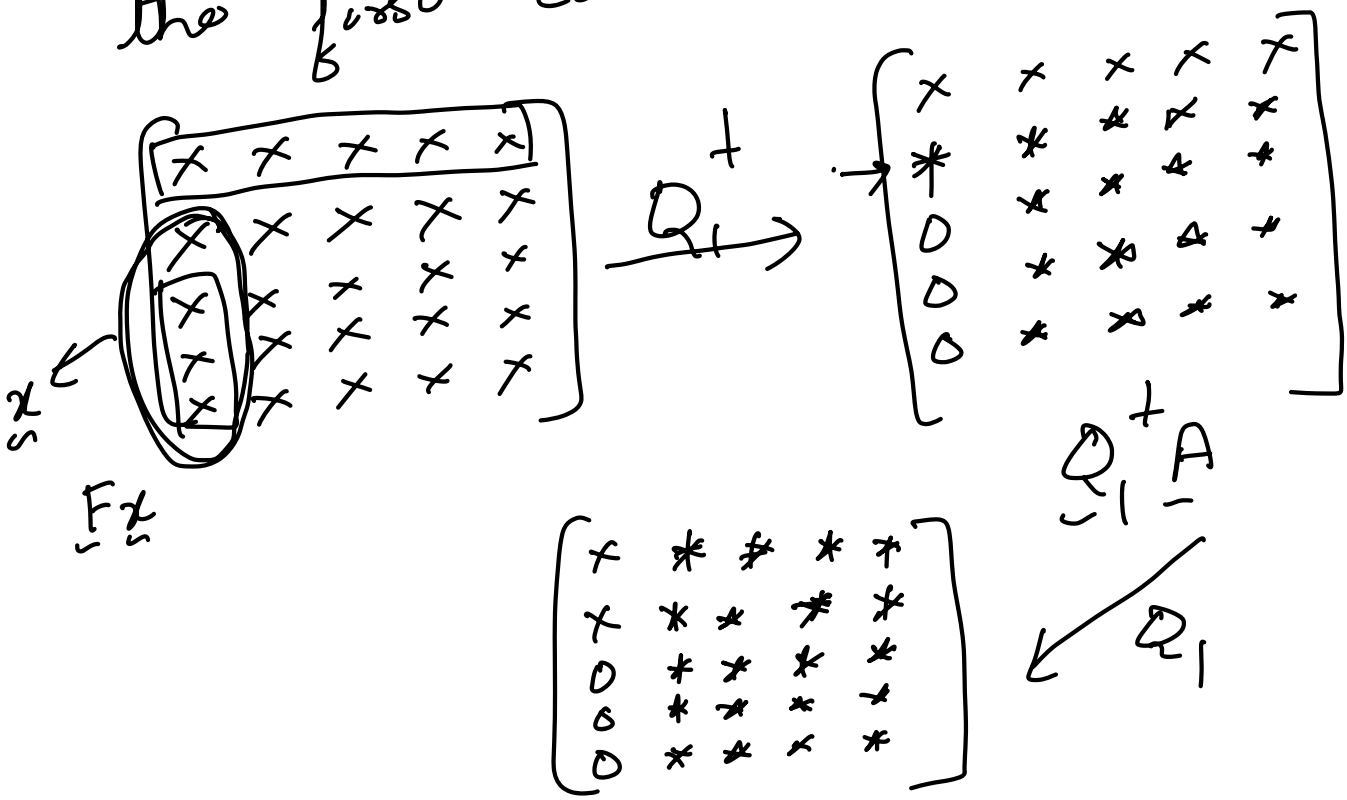
on the right by Q_1



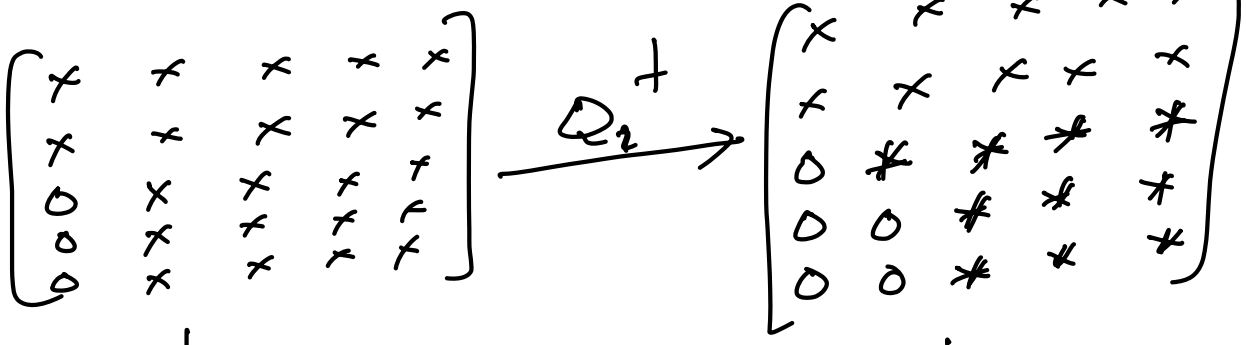
This is not useful to do this!

Instead

→ For the first step, we construct a Householder reflector Q_1^T that does not touch the first row and zeros out the first colum below the first sub diagonal



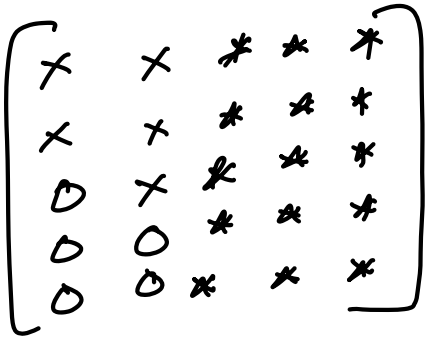
We now repeat this idea in the second column. Construct a Q_2 using a Householder reflector to leave the first 2 rows unchanged and zeros out elements below row 3 etc



$Q_1^+ A Q_1$

$\downarrow Q_2$

$Q_2^+ Q_1^+ A Q_1 Q_2$



You repeat this process $m-2$ times, A is reduced to upper Hessenberg matrix

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix}$$

$$\underline{Q}^{+}_{m-2} \cdots \underline{Q}_2^{+} \underline{Q}_1^{+} \underline{A} \underline{Q}_1 \underline{Q}_2 \cdots \underline{Q}_{m-2} = \underline{H}$$

$$\underline{Q}^{+} \underline{A} \underline{Q} = \underline{H}$$

$$\Rightarrow A = Q H Q^{+}$$

Algo:-

for $k = 1$ to $m-2$ do

$$\underline{x} = A(k+1:m, k)$$

$$\underline{v}_k = \text{sign}(x_1) \|x\|_2 \underline{e}_1 + \underline{x}$$

$$\underline{V}_k = v_k / \|v_k\|_2 \qquad F = (I - 2vv^T)$$

$$A(k+1:m, k:m)$$

$$= A(k+1:m, k:m) - 2 v_k (v_k^*)^T A(k+1:m, k:m)$$

$$A(1:m, k+1:m)$$

$$= A(1:m, k+1:m) - 2 A(1:m, k+1:m) v_k v_k^{*T}$$

end for

* Operation Count :-

Left multiplication $\sim \frac{4m^3}{3}$ flops

Right multiplication $\sim 2m^3$ flops

Total work done to reduce A to upper Hessenberg form

$$\sim \frac{4m^3}{3} + 2m^3$$

$$\sim \frac{10}{3} m^3$$

* Symmetric matrix :-

Pre multiplication with $Q_k^T \sim \frac{4}{3} m^3$

Post multiplication with $Q_k \sim \frac{4}{3} m^3$

Total cost $\sim \frac{8}{3} m^3$

Taking advantage of symmetry

you can get another factor of 2 gain!

Total flop count $\sim \frac{4}{3}m^3$ flops

Let $\tilde{H}$ be computed Hessenberg matrix and as before let $\tilde{Q}$ be exactly orthogonal matrix from the computed reflection vectors $\tilde{v}_k$, then we have

Thm:- Let $A = Q H Q^T$ for some matrix $A \in \mathbb{R}^{m \times m}$ be computed by the above algo. Let $\tilde{H}, \tilde{Q}$ be defined as above, then

$$\tilde{Q}\tilde{H}\tilde{Q}^T = A + \delta A$$

where $\frac{\|\delta A\|}{\|A\|} = O(\epsilon_M)$

for some $\delta A \in \mathbb{R}^{m \times m}$

* Restriction to real symmetric matrices :-

If $\underset{\sim}{A} \in \mathbb{R}^{m \times m}$ and $\underset{\sim}{A} = \underset{\sim}{A}^T$, $\underset{\sim}{A}$ has real eigenvalues and a complete set of orthogonal eigenvectors ie $\underset{\sim}{A} = \underset{\sim}{Q} \underset{\sim}{\Lambda} \underset{\sim}{Q}^T$

$\lambda_1, \lambda_2 \cdots \lambda_m \rightarrow$ real eigenvalues of $\underset{\sim}{A}$

$\underset{\sim}{q}_1, \underset{\sim}{q}_2, \cdots \underset{\sim}{q}_m \rightarrow$ orthonormal eigenvectors

(I) The Rayleigh quotient of a vector $\underset{\sim}{x} \in \mathbb{R}^m$ for a given real symmetric matrix $\underset{\sim}{A} \in \mathbb{R}^{m \times m}$ is the scalar

$$r(\underset{\sim}{x}) = \frac{\underset{\sim}{x}^T \underset{\sim}{A} \underset{\sim}{x}}{\underset{\sim}{x}^T \underset{\sim}{x}}$$

Remarks:-

(i) If $\underset{\sim}{x}$ is an eigenvector then $r(x) = \lambda$

the corresponding eigenvalue of $\underline{A}$ for that eigenvector $\underline{x}$.

(ii) Given $\underline{x} \in \mathbb{R}^m$ (which is not necessarily an eigenvector), what scalar α minimizes

$$\|\underline{A}\underline{x} - \alpha \underline{x}\|_2$$

i.e what scalar acts like an eigenvalue

$$-\underline{x}\alpha = \underline{A}\underline{x}$$

This is an $m \times 1$ least squares problem!

$\underline{x}$ is known matrix $m \times 1$

α is the unknown,

$\underline{A}\underline{x}$ is basically the known right hand side.

$\rightarrow$ m equations for 1 unknown!

Normal equations for our least squares problem

$$(\underline{x}^T\underline{x})\alpha = \underline{x}^T\underline{A}\underline{x}$$

$$\boxed{\alpha = \frac{\underline{x}^T\underline{A}\underline{x}}{\underline{x}^T\underline{x}}}$$

$$\underline{A}\underline{x} = \underline{b}$$

$$\|\underline{A}\underline{x} - \underline{b}\|$$

$$(\underline{A}^T\underline{A})\underline{x} = \underline{A}^T\underline{b}$$

$\alpha = r(x) = \frac{x^T A x}{x^T x}$ which minimizes $\|\alpha x - Ax\|_2$

This "α" is the natural eigenvalue estimate to consider if x is approximately equal to an eigenvector

Given with a vector x, $\boxed{x = \sum_{j=1}^{m} c_j q_j}$

Then $r(x) = \frac{x^T A x}{x^T x} = \frac{\sum_{j=1}^{m} c_j^2 \lambda_j}{\sum_{j=1}^{m} c_j^2}$

If x is close to one of the eigenvectors q_J

$$\frac{|c_j|}{|c_J|} < \varepsilon \quad \text{for all } j \neq J$$

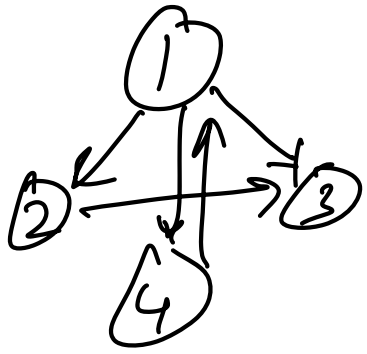
we can show that $\boxed{|r(x) - r(q_J)| = O(\|x - q_J\|^2)}$,

$\boxed{\|x - q_J\| < \varepsilon}$

Rayleigh quotient is quadratically

an accurate estimate of an eigenvalue

Power iteration :-



Suppose $\underset{\sim}{v}^{(0)}$ is a vector such that $\|\underset{\sim}{v}^{(0)}\| = 1$, then power iteration produces a sequence of vectors $\underset{\sim}{v}^{(i)}$ that converges to an eigenvector corresponding to the largest eigenvalue of $\underset{\sim}{A}$.

Algo :- Power iteration

Initialize $\underset{\sim}{v}^{(0)}$ to some vector $\|v^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$$\underset{\sim}{w} = \underset{\sim}{A}\,\underset{\sim}{v}^{(k-1)}$$

$$\underset{\sim}{v}^{(k)} = \frac{\underset{\sim}{w}}{\|\underset{\sim}{w}\|}$$

$$\lambda^{(k)} = \left[\underset{\sim}{v}^{(k)}\right]^T A\, v^{(k)}$$

$$v^{(0)}$$

$$\frac{Av^{(0)}}{\|Av^{(0)}\|}$$

$$A\left[\frac{Av^{(0)}}{\|Av^{(0)}\|}\right]$$

$$\frac{A^k v^{(0)}}{\|A^k v^{(0)}\|}$$

$$v^{(0)} = a_1 q_1 + a_2 q_2 + \cdots a_m q_m$$

$$A^k v^{(0)} = a_1 A^k q_1 + a_2 A^k q_2 + \cdots a_m A^k q_m$$

$$= a_1 \lambda_1^k q_1 + a_2 \lambda_2^k q_2 + \cdots a_m \lambda_m^k q_m$$

$$= a_1 \lambda_1^k \left[q_1 + \left(\frac{a_2}{a_1}\right)\left(\frac{\lambda_2}{\lambda_1}\right)^k q_2 + \cdots + \left(\frac{a_m}{a_1}\right)\left(\frac{\lambda_m}{\lambda_1}\right)^k q_m \right] \quad (*)$$

$\left|\frac{\lambda_j}{\lambda_1}\right| < 1$ for $j > 1$

$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots |\lambda_m| \geq 0$

as $k \to \infty$, $A^k v^{(0)} \to a_1 \lambda_1^k q_1$

$$v^{(k)} = \frac{A^k v^{(0)}}{\|A^k v^{(0)}\|} \longrightarrow q_1 \quad \text{as } k \to \infty$$

Thm:- Let $|\lambda_1| > |\lambda_2| \geq \cdots |\lambda_m| \geq 0$ and $q_1^T v^{(0)} \neq 0$, then iterates of the power iteration satisfy

$$\|v^{(k)} - (\pm q_1)\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \text{ and } |\lambda^{(k)} - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right) \text{ as } k \to \infty$$

If $\lambda_1 > 0$, $\dfrac{\underline{A}^k \underline{v}^{(0)}}{\|\underline{A}^k \underline{v}^{(0)}\|} \to \dfrac{a_1 \lambda_1^k \underline{q}_1}{|a_1 \lambda_1^k|}$

If $\lambda_1 < 0$, $\dfrac{\underline{A}^k \underline{v}^{(0)}}{\|\underline{A}^k \underline{v}^{(0)}\|} \to \dfrac{a_1 \lambda_1^k \underline{q}_1}{|a_1 \lambda_1^k|} \to \underline{q}_1$

if k is even

$$\frac{a_1 \lambda_1^k \underline{q}_1}{|a_1 \lambda_1^k|} = \frac{a_1 \lambda_1^k \underline{q}_1}{a_1 \lambda_1^k}$$

if k is odd, $|a_1 \lambda_1^k| = -a_1 \lambda_1^k$

$$\frac{a_1 \lambda_1^k \underline{q}_1}{|a_1 \lambda_1^k|} = -\underline{q}_1$$

Shortcomings :-

(a) It can only find eigenvector corresponding to largest eigenvalue

(b) Convergence is linear with error being reduced by a constant factor $\simeq \left|\frac{\lambda_2}{\lambda_1}\right|$ at each iteration

(c) If $\lambda_2 \simeq \lambda_1$ convergence can be very slow!

Inverse iteration:-

* We amplify the differences between eigenvalues and hence accelerate the convergence!

* We pick $\mu \in \mathbb{R}$ that is not eigenvalue of $\underline{\underline{A}}$, the eigenvectors of $(\underline{\underline{A}} - \mu \underline{\underline{I}})^{-1}$ are same as $\underline{v}_j$ eigenvectors of $\underline{\underline{A}}$, and the corresponding eigenvalues are $\left\{ \frac{1}{\lambda_j - \mu} \right\}_{j=1}^{m}$ where $\{\lambda_j\}_{j=1}^{m}$ are eigenvalues of $\underline{\underline{A}}$.

Now suppose μ is close eigenvalue λ_J of $\underline{\underline{A}}$, then $\frac{1}{\lambda_J - \mu}$ will be

much larger than $\frac{1}{\lambda_j - \mu}$ for all $j \neq J$

→ If we apply power iteration to $(A - \mu I)^{-1}$, the process would converge rapidly to q_J.

The idea is called inverse iteration

Algo:- Initialize $v^{(0)}$ to some vector with $\|v^{(0)}\| = 1$

Initialize μ to some value near λ_J

for $k = 1, 2, \ldots$

→ Solve $(A - \mu I) w = v^{(k-1)}$ for w | % $(A - \mu I)^{-1}$ on $v^{(k-1)}$

→ $v^{(k)} = \frac{w}{\|w\|}$

→ $\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$

Thm :- Suppose λ_J is the closest eigenvalue to μ and λ_K is the second closest

i.e $|\mu - \lambda_J| < |\mu - \lambda_K| \leq |\mu - \lambda_j|$ for all $j \neq J$

Suppose $q_J^T v^{(0)} \neq 0$, then the iterates of the inverse iteration satisfy

$$\| v^{(k)} - (\pm q_J) \| = O\left(\left(\frac{|\mu - \lambda_J|}{|\mu - \lambda_K|} \right)^k \right)$$

and $$|\lambda^{(k)} - \lambda_J| = O\left(\left| \frac{\mu - \lambda_J}{\mu - \lambda_K} \right|^{2k} \right)$$

Power iteration on A

(i) eigenvector estimate $\rightarrow$ eigenvalue estimate

Inverse iteration

(i) eigenvalue estimate $\rightarrow$ eigenvector estimate

Algo:- Rayleigh quotient iteration.

Initialize $\underline{v}^{(0)}$ to some vector $\|\underline{v}^{(0)}\| = 1$

Initialize $\lambda^{(0)} = (\underline{v}^{(0)})^T \underline{A}\, \underline{v}^{(0)}$

for $k = 1, 2, \ldots$

Solve $(\underline{A} - \lambda^{(k-1)} \underline{I})\, \underline{w} = v^{(k-1)}$;

$v^{(k)} = \frac{w}{\|w\|}$

$\lambda^{(k)} = (v^{(k)})^T \underline{A}\, v^{(k)}$

end for.

Each iteration triples the number of digits of accuracy

When it converges, convergence is cubic i.e if λ_J is an eigenvalue of

A and $v^{(0)}$ is sufficiently close to the eigenvector q_J, then as $k \to \infty$

$$\|v^{(k+1)} - (\pm q_J)\| = O(\|v^{(k)} - q_J\|^3)$$

and $$|\lambda^{(k+1)} - \lambda_J| = O(|\lambda^{(k)} - \lambda_J|^3)$$

* Operation Counts:-

$A \in \mathbb{R}^{m \times m}$ is dense

* Each step of power iteration requires matrix vector multiplication : $O(m^2)$ flops.

* Each step of inverse iteration requires solution of linear system of equations which might seem to require $O(m^3)$ flops. But the

coefficient matrix $\underline{A} - \mu \underline{I}$ does not change at every k^{th} step, you can compute $\underline{L}\underline{U}$ factorization once and every time RHS changes for a given 'k', you just use backward substitution which costs only $O(m^2)$ flops.

* For Rayleigh quotient iteration, the matrix to be inverted at each step changes, so the cost will be $O(m^3)$ flops per step!

Very few iterations may be required and hence it is a practical method

* If $\underline{A}$ is a real symmetric matrix

employ phase I to reduce $\underline{A}$ to a tridiagonal matrix, each of these procedures take $O(m)$ flops per each step!