# Welcome to Ma 221! Lecture 20, Fall 24

## Symmetric Eigenproblem
(SVD analogous)

(1) "Usual accuracy": backward stable:
exact evals, evecs for $A+E$, $\frac{\|E\|}{\|A\|} = O(\epsilon)$

all start with tridiagonal reduction

$A = QTQ^T \qquad QQ^T = I$

$T =$ 

(1.1) Given $T$ find all evals (w or w/o evecs)

(1.1.1): QR iteration with shift (Chap 4)
but better: cubically convergent
cost $= O(n^2)$ for evals
$+ O(n^3)$ for evecs

default for $n \le 25$

LAPACK: ssyev

(1.1.2) Improve $O(n^3)$ cost for evecs to $O(n^2)$ without guarantee of orthogonality (fix later)
- bisection for evals
- inverse iteration for evecs

LAPACK: ssyevx

(1.1.3) Divide + Conquer (Gu etal)

only for all evals + evecs

cost = $O(n^g)$ $2 < g < 3$

default in Matlab ssyevd in LAPACK

(1.1.4) MRRR = Multiple Relatively Robust Representations

like inverse iteration, guarantees orthogonality, cost = $O(n^2)$

LAPACK = ssyevr

default in Julia, SciPy

Extension to SVD still open question (draft implementation, some bugs)

(1.2) Some evals, w or w/o evecs:

Reduce to tridiagonal as before;

use bisection to find a few evals, in range $[x, y]$, or $\lambda_i, \lambda_{i+1}, \ldots \lambda_j$

use inverse iteration or MRRR to get evecs

(2) High Accuracy: Tiny evals or sing vals more accurately: Jacobi works

LAPACK (SVD only): sgesvj

(3) Updating evals + evecs of $A \pm vv^T$

(basis of divide + conquer)

---

Matlab demo of cubic convergence
(code in typed notes

Cubic convergence follows from analysis of Rayleigh Quotient Iteration

$i=0$
choose unit vector $x_0$
repeat
$s_i = \rho(x_i, A) = x_i^T A x_i$
$y = (A - s_i I)^{-1} x_i$
$x_{i+1} = y / \|y\|_2$
until convergence
($s_i$ or $x_i$ stops changing (much))

Inverse iteration using Rayleigh Quotient as shift, best available approx eval given approx evec $x_i$

Suppose $Aq = \lambda q$, $\|q_2\|_2 = 1$, $\|x_i - q\|_2 = e << 1$
Want to show $\|x_{i+1} - q\|_2 = O(e^3)$

Bound $|s_i - \lambda|$:

$$s_i = x_i^T A x_i = (x_i - q + q)^T A (x_i - q + q)$$
$$= (x_i - q)^T A (x_i - q) + \underbrace{q^T A}_{\lambda q^T} (x_i - q)$$
$$+ (x_i - q)^T \underbrace{Aq}_{q\lambda} + \underbrace{q^T Aq}_{\lambda}$$

$$s_i - \lambda = (x_i - q)^T A (x_i - q) + 2\lambda (x_i - q)^T q$$

$$|s_i - \lambda| \le O(\|x_i - q\|_2^2) + O(\|x_c - q\|_2)$$

$$= O(\epsilon^2) + O(\epsilon)$$

$$= O(\epsilon) \quad \ldots \text{ want tighter bound}$$

$$|s_i - \lambda| \le \|A x_i - s_i x_i\|_2^2 / gap$$

... gap = distance from $s_i$ to second closest eval

... Thm 5.5 in book, sketched last time

$$= \|A(x_c - q + q) - s_i (x_c - q + q)\|_2^2 / gap$$

$$= \|(A - s_i I)(x_i - q) + (\lambda - s_i) \cdot q\|_2^2 / gap$$

$$\le \left(\|(A - s_i I)(x_c - q)\|_2 + \|(\lambda - s_i) \cdot q\|_2\right)^2 / gap$$

$$= \left(O(\epsilon) + O(\epsilon)\right)^2 / gap$$

$$= O(\epsilon^2) / gap$$

Use analysis from Chap 4 of one step of inverse iteration

$$\|x_{i+1} - q\|_2 \le \underbrace{\|x_i - q\|_2}_{O(\epsilon)} \cdot \underbrace{\frac{|s_i - \lambda|}{gap}}_{O(\epsilon^2)}$$

$$= O(\epsilon^3) / gap$$

Show that QR iteration doing Rayleigh Quotient iteration implicitly

$$T - s_i I = QR \rightarrow \text{new } T = RQ + s_i I$$

$$(T - s_i I)^{-1} = R^{-1} Q^{-1}$$
$$= R^{-1} Q^T \quad \text{symmetric}$$
$$= (R^{-1} Q^T)^T$$
$$= Q R^{-T}$$
$$(T - s_i I) R^{-T} = Q$$

last column: $(T - s_i I)^{-1} \cdot e_n \, R(n,n) =$ last col of $Q$

$$s_i = T(n,n) = e_n^T T e_n = \rho(e_n, T)$$

$\Rightarrow q_n =$ last col of $Q =$ result of 1 step of Rayleigh Quotient iteration starting with $e_n$

$$\text{new } T = RQ + s_i I = Q^T T Q$$
$$(\text{new } T)(n,n) = q_n^T T q_n = \rho(q_n, T)$$

$\Rightarrow$ QR iteration doing Rayleigh Quotient iteration

Actual implementation uses "bulge chasing" as in Chap 4, cost $= O(n)$ to get from $T_i$ to $T_{i+1}$

$\Rightarrow$ cost $= O(n)$ per eval
cost $= O(n^2)$ for all evals

But cost to compute evecs by taking product of all $O(n^2)$ Givens rotations $= O(n^3)$, want to go faster

# Next algorithm to beat $O(n^3)$

Divide + Conquer

only for all evals + evecs

Main ingredient: cheaply updating evals and evecs of $A = Q\Lambda Q^T$ for $A + \alpha uu^T$

$$A + \alpha uu^T = Q\Lambda Q^T + \alpha uu^T$$
$$= Q(\Lambda + \alpha (Q^T u)(Q^T u)^T) Q^T$$
$$= Q(\Lambda + \alpha \cdot v \cdot v^T) Q^T$$

need evals and evecs of $\Lambda + \alpha \cdot v \cdot v^T$

= diagonal + rank 1

use characteristic polynomial and

HW Q5.14: $\det(I + xy^T) = 1 + y^T x$

$$\det(\Lambda + \alpha vv^T - \lambda I)$$
$$= \det((\Lambda - \lambda I)(I + \alpha (\Lambda - \lambda I)^{-1} v v^T))$$
$$= \prod_{i=1}^{n} (\lambda_i - \lambda) \cdot \underbrace{\left(1 + \alpha \sum_{i=1}^{n} \frac{v_i^2}{\lambda_i - \lambda}\right)}_{f(\lambda),\ \text{want roots}}$$

Fig 5.2 in text: $1 + \frac{.5}{1-\lambda} + \frac{.5}{2-\lambda} + \frac{.5}{3-\lambda} + \frac{.5}{4-\lambda}$

Fig 5.3 in text: $1 + \frac{.001}{1-\lambda} + \frac{.001}{2-\lambda} + \frac{.001}{3-\lambda} + \frac{.001}{4-\lambda}$

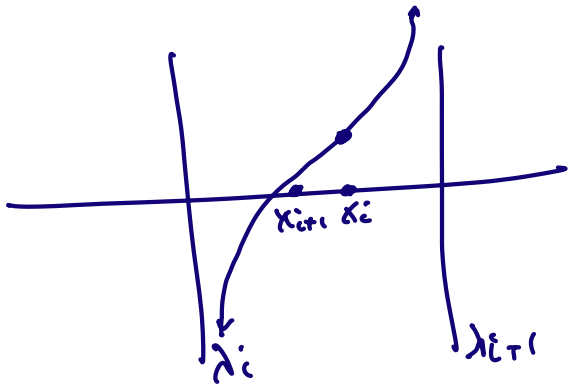
Takeaway: can't use plain Newton

Use Variant of Newton:

approximate $f(\lambda)$ by $g(\lambda)$ that has poles at $\lambda_i$ and $\lambda_{i+1}$ and matches $f$ and $f'$ at last iteration

$$g(\lambda) = c_1 + \frac{c_2}{\lambda_i - \lambda} + \frac{c_3}{\lambda_{i+1} - \lambda}$$

solving $g(\lambda) = 0$ : quadratic




$g(x_i) = f(x_i)$

$g'(x_i) = f'(x_i)$

Need evecs:

Lemma: if $\lambda$ eval of $\Lambda + \alpha v v^T$ then its evec is $(\Lambda - \lambda I)^{-1} v$

proof: $(\Lambda + \alpha v v^T)(\Lambda - \lambda I)^{-1} v$

$= (\Lambda - \lambda I + \lambda I + \alpha v v^T)(\Lambda - \lambda I)^{-1} v$

$= \underbrace{v}_{\text{cancels}} + \lambda (\Lambda - \lambda I)^{-1} v + v \underbrace{(\alpha v^T (\Lambda - \lambda I)^{-1} v)}_{\substack{= -1 \text{ since} \\ f(\lambda) = 0}}$

$\underbrace{\qquad\qquad}_{-v}$

$= \underbrace{\lambda (\Lambda - \lambda I)^{-1} v}_{\text{evec}}$

Oops: not numerically stable because of cancellation in $A - \lambda I$

clever fix in text, due to Gu + Eisenstat

For general eigenproblem

Reduce A to tridiagonal T

Write alg just presented as

$[Q', \Lambda'] = \text{Eig-update}[Q, \Lambda, \alpha, v]$

How to divide T into 2 problems of half size?

$$T = \left[\begin{array}{cc|cc} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & b_{i-1} & \\ & b_{i-1} & a_i & b_i \\ \hline & & b_i & a_{i+1} \; b_{i+1} \\ & & & b_{i+1} \; \ddots \\ & & & \ddots \; a_n \end{array}\right]$$

$$= \left[\begin{array}{cc|cc} a_1 & b_1 & & \\ b_1 & \ddots & & \\ & \ddots & b_{i-1} & \\ & b_{i-1} & a_i - b_i & 0 \\ \hline & & 0 & a_{i+1} - b_i \; b_{i+1} \\ & & & b_{i+1} \; \ddots \end{array}\right] + \left[\begin{array}{cc|cc} 0 & & 0 & \\ & & & \\ & & b_i & b_i \\ \hline & & b_i & b_i \\ 0 & & & 0 \end{array}\right]$$

$$= \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_i \, v v^T \qquad v = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ v \\ \vdots \\ 0 \end{bmatrix} \qquad i = \left\lfloor \frac{n}{2} \right\rfloor$$

function $[Q, \Lambda] = DC\_eig(T)$

... return $T = Q \Lambda Q^T$, $n = \dim(T)$

if n small enough

use QR ... base case

else

$i = \lfloor \frac{n}{2} \rfloor$

... write $T = diag(T_1, T_2) + b_i u u^T$

... no work, just notation

$[Q_1, \Lambda_1] = DC\_eig(T_1)$

$[Q_2, \Lambda_2] = DC\_eig(T_2)$

... $\left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] \left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right] = \left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right]$

$[Q, \Lambda] = Eig\_update(diag(Q_1, Q_2),$
$diag(\Lambda_1, \Lambda_2), b_i, u)$

end if

return

cost $= O(n^g)$ $2 < g < 3$

---

Algorithm for evals only

(just some evals)

use bisection, Sylvester's Thm

inertia $(A) = (\#evals < 0, \#evals = 0, \#evals > 0)$

$= inertia(X A X^T)$

$X$ nonsingular

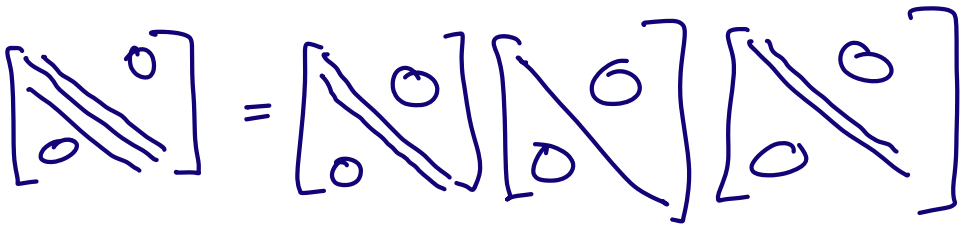
count #evals in any interval $[x,y]$
using inertia of $A-xI$ and $A-yI$

: choose $X$ so $X(A-\lambda I)X^T$ = diagonal $D$

Use Gaussian Elimination without pivoting

$A \rightarrow$ tridiagonal $T$

$T - xI = LDL^T$



why numerically stable?

NegCount$(T,x)$ = # evals of $T < x$

... diag$(T) = a_1, a_2, \ldots a_n$

... off diag$(T) = b_1, b_2, \ldots b_{n-1}$

... only compute diagonal of $D = d_1, d_2, \ldots, d_n$

$b_0 = 0$, $d_0 = 1$, $c = 0$

for $i = 1$ to $n$

$d_i = (a_i - x) - b_{i-1}^2 / d_{i-1}$

... obey parentheses!

if $d_i < 0$, $c = c+1$

end

if $d_{i-1} = 0 \Rightarrow$
$d_i = -\infty \Rightarrow$
$d_{i+1} = a_{i+1} - x$

Why we don't need $l_i = i^{th}$ offdiagonal of $L$

$(T - xI)(i, i+1) = b_i$

$= (LDL^T)(i, i+1) = d_i \cdot l_i$

$\Rightarrow l_i = b_i / d_i$

Usual inner loop of GE:

$$d_i = a_i - x - l_{i-1}^2 \cdot d_{i-1}$$

replaced by

$$d_i = a_i - x - b_{i-1}^2 / d_{i-1}$$

Thm: Assuming we don't divide by $d_{i-1} = 0$

Neg count $(T, x)$ is exactly correct for $T + E$ where $E = E^T$, tridiagonal

$E(i,i) = 0$ and

$|E(i,i+1)| \le 2.5 \cdot \varepsilon \cdot |T(i,i+1)|$

proof: $d_i = \Big( (a_i - x) - b_{i-1}^2 / d_{i-1} \Big) (1+\delta_4)$
$\cdot (1+\delta_1) \quad (1+\delta_2) \cdot (1+\delta_3)$

$d_i (1+\delta_i)^{-1} (1+\delta_4)^{-1} = (a_i - x) - b_{i-1}^2 \underbrace{(1+\delta)}_{\text{several terms}} / d_{i-1}$

same signs

$(d_i \cdot F_i) = (a_i - x) - b_{i-1}^2 \cdot G_i / d_{i-1}$

$= (a_i - x) - b_{i-1}^2 G_i F_{i-1} / (d_{i-1} \cdot F_{i-1})$

$d_i' = (a_i - x) - b_{i-1}'^2 / d_{i-1}'$

$\text{sign}(d_i') = \text{sign}(d_i)$

exact recurrence for $T + E$ where $E$ changes $b_i$ to $b_i'$

values of $d_i'$ have same signs as $d_i$

$\Rightarrow$ correct count for $T + E$

What about $d_{i-1} = 0$? No problem