Chap 5 - Symm. Eigen problem + SVD
Part 3: Algorithms

Overview: Several approaches depending on goals:

(1) "Usual accuracy": backward stable, exact answer for $A + E$, $\|E\|_2 = O(\varepsilon) \|A\|_2$
   \[\rightarrow (1.1) \text{ Get all evals (w or w/o evens)} \]
   \[ (1.2) \text{ Just evals in } [x, y] \text{ (w/o evens)} \]
   \[ (1.3) \text{ Just get } d_1, d_2, \ldots, d_j \ (\text{"1"}) \]
   \[Eq. \text{ just } d_1, \ldots, d_{10} \]
   (1.2) and (1.3) can be much cheaper than (1.1)
   if only few evals (evens desired)

(2) "High Accuracy"

Ex: A well-conditioned, so all sing vals are same size
   say $O(1)$, then usual error bound:
   \[ \sigma_i \pm O(\varepsilon) \sigma_i \]
   \[\rightarrow \text{ all } \sigma_i \text{'s computed with } \text{high relative accuracy } \Rightarrow \]
leading digits correct

B = D \cdot A , \text{ Diagonal, some } D_{ii} \text{ are much smaller than others } \Rightarrow \text{ some tiny } \sigma_i \text{ usual accuracy does not guarantee relative accuracy of tiny } \sigma_i \text{ In fact } B \text{ is "well conditioned" in sense: small relative perturbations in each } B_{ij} \Rightarrow \text{ small relative perturbation in all } \sigma_i \text{ (new perturbation theory: "Relative Weyl's Thm")}

3 algorithms that achieve this
- see links to paper in long Chap 5 notes and web page

(3) Updating: given } A = Q \cdot \Lambda \cdot Q^T \text{, cheaply compute } \text{eivals (basis of } A \pm xx^T \text{ (basis of one alg for (1.1))}
All this applies to SVD.

Algorithms and costs:
1. Start by reducing $A$ to $Q^T T Q$ where $T$ is tridiagonal.
   
   All algorithms work on $T$.
   
   Costs $O(n^3)$ flops, possible to only move $O\left(\frac{n^3}{\text{fastmem size}}\right)$ words main mem $\rightarrow$ fast mem.
   
   See link on web page.
   
   $A$ banded computing $T$ costs $O(n^2 b)$ flops.

   SVD: $A = U^T B V$, $U$, $V$ orthogonal.
   
   $B$ bidiagonal.

(1.1) Given $T$ find all evals, possibly evecs.

   Cost: $O(n^2)$ just for evals, anywhere from $O(n^2)$ to $O(n^3)$ for evecs.
some speed/accuracy tradeoff

(1.1.1) QR Iteration (Chap 4)

Thm (Wilkinson): If choose right shift, tridiagonal QR globally convergent, usually cubically convergent (4 correct digits triples at each iteration)

Cost: \( O(n^2) \) for evals but \( O(n^3) \) for evecs

only multiplies by orthogonal matrices \( \Rightarrow \) backward stable

LAPACK: ssyev

QR Iteration for SVD of bidiagonal \( B \)
guarantees high relative accuracy for all \( \sigma_i \)'s, no matter how small

(1.1.2) Find all evecs in \( O(n^2) \)
but does not guarantee they are orthogonal

(1) Compute evals alone (LAPACK: sstebz)
(2) Computes their evecs using inverse iteration,
\[ X_{i+1} = (T - \lambda_i I)^{-1} X_i \quad \text{(LAPACK: \textsc{stein})} \]

\( T \) tri-diagonal \( \Rightarrow \) one step cost \( O(n) \)
and since \( \lambda_j \) accurate eval,
\( \Rightarrow \) only \( O(1) \) steps
\( \Rightarrow \) total cost \( O(n^2) \)

Problem: if \( \lambda_j \) and \( \lambda_{j+1} \) very close, no guarantee their evecs are orthogonal

E.g.: Suppose \( \lambda_j \) and \( \lambda_{j+1} \)
round to same floating point number

Open problem for many years to get orthogonal evecs in \( O(n^2) \)

\( \Rightarrow \) MRRR algorithm

(1.1.3) "Divide and Conquer" (DC)
faster than QR, not as fast as inv. it., reliable
\( \text{cost } O(n^2) \) \( \leq n^3 \)
\[ A = x \xi^T \quad (\text{LAPACK: ssyeval}) \]

(1.1.4) MRRR = Multiple Relatively Robust Representations

- variant of Inv. It.
  (Parlett + Dhillon) (ssyevr)

- Still open to extend MRRR reliably to SVD (Willems thesis, but some stability gaps)

Theoretical Improvement (Gu):

- can beat \( O(n^3) \) for computing all evecs of \( T \)-why isn't \( \Omega(n^2) \) a lower bound? Trick: represent evecs implicitly as a product of "simple" orthogonal matrices:
  \[ \text{cost} = O(n (\log n)^p) \quad \text{small } p \]

(1.2) or (1.3) - only some evecs and evecs:

- Use bisection (based on Sylvester's Thm) to compute only desired evals
- Use inverse iteration (or MRRR)
to compute their evecs:
\( O(n^3) \) for each eval/evec

(2): Higher Accuracy:
use Jacobi's Algorithm, updated
by Drmac/Veselic for SVD
\( \text{LAPACK: } \text{sgesv} \)
slower, not available for eigen-
problem

(3): got evecs of \( A \pm xx^T \)
given \( A = Q D Q^T \): use same
idea as for \( D \)