C6.1 Numerical Linear Algebra

- **SVD** and its properties, applications
- **Direct methods** for linear systems and least-squares problems
- **Direct methods** for eigenvalue problems
- **Iterative** (Krylov subspace) methods for linear systems
- **Iterative** (Krylov subspace) methods for eigenvalue problems
- **Randomised algorithms** for SVD and least-squares

References

- Trefethen-Bau (97): Numerical Linear Algebra
  - covers essentials, beautiful exposition
- Golub-Van Loan (12): Matrix Computations
  - classic, encyclopedic
- Horn and Johnson (12): Matrix Analysis (& topics (86))
  - excellent theoretical treatise, little numerical treatment
- J. Demmel (97): Applied Numerical Linear Algebra
  - impressive content, some niche
- N. J. Higham (02): Accuracy and Stability of Algorithms
  - bible for stability, conditioning
- H. C. Elman, D. J. Silvester, A. J. Wathen (14): Finite elements and fast iterative solvers
  - PDE applications of linear systems, preconditioning

What is numerical linear algebra?

The study of numerical algorithms for problems involving matrices

Two main (only!?) problems:

1. Linear system

$$Ax = b$$

2. Eigenvalue problem

$$Ax = \lambda x$$

λ: eigenvalue (eigval), x: eigenvector (eigvec)

What is numerical linear algebra?

The study of numerical algorithms for problems involving matrices

Two main (only!?) problems:

1. Linear system

$$Ax = b$$

2. Eigenvalue problem

$$Ax = \lambda x$$

   λ: eigenvalue (eigval), x: eigenvector (eigvec)

3. SVD (singular value decomposition)

$$A = U\Sigma V^T$$

   U, V: orthonormal/orthogonal, Σ diagonal

Why numerical linear algebra?

- Many (in fact most) problems in scientific computing (and even machine learning) boil down to a linear problem
  - Because that's often the only way to deal with the scale of problems we face today! (and in future)
  - For linear problems, so much is understood and reliable algorithms available
- $Ax = b$: e.g. Newton's method for $F(x) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$ nonlinear
  1. start with initial guess $x^{(0)} \in \mathbb{R}^n$
  2. find Jacobian matrix $J \in \mathbb{R}^{n\times n}$, $J_{ij} = \frac{\partial F_i(x)}{\partial x_j}|_{x=x^{(0)}}$
  3. update $x^{(1)} := x^{(0)} - J^{-1}F(x^{(0)})$, repeat
- $Ax = \lambda x$: e.g. Principal component analysis (PCA), data compression, Schrödinger eqn., Google pagerank,
- Other sources: differential equations, optimisation, regression, data analysis, ...

Basic linear algebra review

For $A \in \mathbb{R}^{n \times n}$, (or $\mathbb{C}^{n \times n}$; hardly makes difference)

The following are equivalent (how many can you name?):

1. A is nonsingular.

Basic linear algebra review

For $A \in \mathbb{R}^{n\times n}$, (or $\mathbb{C}^{n\times n}$; hardly makes difference)

The following are equivalent (how many can you name?):

1. A is nonsingular.
2. A is invertible: A^{-1} exists.
3. The map $A : \mathbb{R}^n \to \mathbb{R}^n$ is a bijection.
4. all n eigenvalues of A are nonzero.
5. all n singular values of A are positive.
6. $\mathrm{rank}(A) = n$.
7. the rows of A are linearly independent.
8. the columns of A are linearly independent.
9. $Ax = b$ has a solution for every $b \in \mathbb{C}^n$.
10. A has no nonzero null vector. Neither does A^T.
11. A^*A is positive definite (not just semidefinite).
12. $\det(A) \neq 0$.
13. A^{-1} exists such that $A^{-1}A = AA^{-1} = I_n$.
14. $\ldots$

Structured matrices

For square matrices,

- Symmetric: $A = A^T$, i.e. $A_{ij} = A_{ji}$ (Hermitian: $A_{ij} = \bar{A_{ji}}$) has eigenvalue decomposition $A = V\Lambda V^T$, V orthogonal, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$.
  - symmetric positive (semi)definite $A \succ (\succeq) 0$: symmetric and positive eigenvalues
- Orthogonal: $AA^T = A^TA = I$ (Unitary: $AA^* = A^*A = I$) $\rightarrow$ note $A^TA = I$ implies $AA^T = I$
- Skew-symmetric: $A_{ij} = -A_{ji}$ (skew-Hermitian: $A_{ij} = -\bar{A_{ji}}$)
- Normal: $A^TA = AA^T$
- Tridiagonal: $A_{ij} = 0$ if $|i-j| > 1$
- Triangular: $A_{ij} = 0$ if $i > j$

For (possibly nonsquare) matrices $A \in \mathbb{C}^{m\times n}$, $m \geq n$

- Hessenberg: $A_{ij} = 0$ if $i > j+1$
- "orthonormal": $A^*A = I_n$,
- sparse: most elements are zero

other structures: Hankel, Toeplitz, circulant, symplectic, ...

Vector norms

For vectors $x = [x_1, \ldots, x_n]^T \in \mathbb{C}^n$

- p-norm $\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$
  - Euclidean norm=2-norm $\|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2}$
  - 1-norm $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$
  - ∞-norm $\|x\|_\infty = \max_i |x_i|$

Norm axioms

- $\|\alpha x\| = |\alpha|\|x\|$ for any $\alpha \in \mathbb{C}$
- $\|x\| \geq 0$ and $\|x\| = 0 \Leftrightarrow x = 0$
- $\|x + y\| \leq \|x\| + \|y\|$

Inequalities: For $x \in \mathbb{C}^n$,

- $\frac{1}{\sqrt{n}}\|x\|_2 \leq \|x\|_\infty \leq \|x\|_2$
- $\frac{1}{\sqrt{n}}\|x\|_1 \leq \|x\|_2 \leq \|x\|_1$
- $\frac{1}{n}\|x\|_1 \leq \|x\|_\infty \leq \|x\|_1$

$\|\cdot\|_2$ is **unitarily invariant** as $\|Ux\|_2 = \|x\|_2$ for any unitary U and any $x \in \mathbb{C}^n$.

Cauchy-Schwarz inequality

For any $x, y \in \mathbb{R}^n$,

$$|x^T y| \leq \|x\|_2 \|y\|_2$$

Proof:

- For any scalar c, $\|x - cy\|^2 = \|x\|^2 - 2cx^T y + c^2\|y\|^2$.
- This is minimised w.r.t. c at $c = \frac{x^T y}{\|y\|^2}$ with minimiser $\|x\|^2 - \frac{(x^T y)^2}{\|y\|^2}$.
- Since the minimal value must be ≥ 0, the CS inequality follows.

Matrix norms

For matrices $A \in \mathbb{C}^{m\times n}$,

- p-norm $\|A\|_p = \max_x \frac{\|Ax\|_p}{\|x\|_p}$
  - 2-norm=spectral norm (=operator norm) $\|A\|_2 = \sigma_{\max}(A)$ (largest singular value)
  - 1-norm $\|A\|_1 = \max_i \sum_{j=1}^{m} |A_{ji}|$
  - ∞-norm $\|A\|_\infty = \max_i \sum_{j=1}^{n} |A_{ij}|$
- Frobenius norm $\|A\|_F = \sqrt{\sum_i \sum_j |A_{ij}|^2}$
  (2-norm of vectorization)
- trace norm=nuclear norm $\|A\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i(A)$

Red: **unitarily invariant** norms $\|A\| = \|UAV\|$ for any unitary (or orthogonal) U, V

Norm axioms hold for each. Inequalities: For $A \in \mathbb{C}^{m\times n}$, (exercise)

- $\frac{1}{\sqrt{n}}\|A\|_\infty \le \|A\|_2 \le \sqrt{m}\|A\|_\infty$
- $\frac{1}{\sqrt{m}}\|A\|_1 \le \|A\|_2 \le \sqrt{n}\|A\|_1$
- $\|A\|_2 \le \|A\|_F \le \sqrt{\min(m,n)}\|A\|_2$

Subspaces and orthonormal matrices

Subspace $\mathcal{S}$ of $\mathbb{R}^n$: vectors of form $\sum_{i=1}^{d} c_i v_i$, $c_i \in \mathbb{R}$

- $v_1, \ldots, v_d$ are **basis vectors**, linearly independent
- $x \in \mathcal{S} \Leftrightarrow \sum_{i=1}^{d} c_i v_i$
- d is the *dimension* of $\mathcal{S}$

Representation: $\mathcal{S} = \text{span}(V)$ (i.e., $x \in \mathcal{S} \Leftrightarrow x = Vc$), or just V; often convenient if $V (= Q)$ is orthonormal

Subspaces and orthonormal matrices

Subspace $\mathcal{S}$ of $\mathbb{R}^n$: vectors of form $\sum_{i=1}^{d} c_i v_i$, $c_i \in \mathbb{R}$

- $v_1, \dots, v_d$ are **basis vectors**, linearly independent
- $x \in \mathcal{S} \Leftrightarrow \sum_{i=1}^{d} c_i v_i$
- d is the *dimension* of $\mathcal{S}$

Representation: $\mathcal{S} = \text{span}(V)$ (i.e., $x \in \mathcal{S} \Leftrightarrow x = Vc$), or just V; often convenient if $V(= Q)$ is orthonormal

Lemma

$\mathcal{S}_1 = \text{span}(V_1)$ and $\mathcal{S}_2 = \text{span}(V_2)$ where $V_1 \in \mathbb{R}^{n \times d_1}$ and $V_2 \in \mathbb{R}^{n \times d_2}$, with $d_1 + d_2 > n$. Then $\exists x \neq 0$ in $\mathcal{S}_1 \cap \mathcal{S}_2$, i.e., $x = V_1 c_1 = V_2 c_2$ for some vectors c_1, c_2.

Proof: Let $M := [V_1, V_2]$, of size $n \times (d_1 + d_2)$. Since $d_1 + d_2 > n$ by assumption, M has a right null vector. $Mc = 0$. Write $c = \begin{bmatrix} c_1 \\ -c_2 \end{bmatrix}$.

Some useful results

- $(AB)^T = B^T A^T$
- If A, B invertible, $(AB)^{-1} = B^{-1}A^{-1}$
- If A, B square and $AB = I$, then $BA = I$
- $\begin{bmatrix} I_m & X \\ 0 & I_n \end{bmatrix}^{-1} = \begin{bmatrix} I_m & -X \\ 0 & I_n \end{bmatrix}$
- Neumann series: if $\|X\| < 1$ in any norm,

$$(I - X)^{-1} = I + X + X^2 + X^3 + \cdots$$

- Trace $\text{Trace}(A) = \sum_{i=1}^{n} A_{i,i}$ (sum of diagonals, $A \in \mathbb{R}^{m \times n}$). For any X, Y s.t. $\text{Trace}(XY) = \text{Trace}(YX)$. For $B \in \mathbb{R}^{m \times n}$, we have $\|B\|_F^2 = \sum_i \sum_j |B_{ij}|^2 = \text{Trace}(B^T B)$.
- Triangular structure is invariant under addition, multiplication, and inversion
- Symmetry is invariant under addition and inversion, *but not multiplication*; AB usually not symmetric even if A, B are

SVD: the most important matrix decomposition

- **Symmetric eigenvalue decomposition**: $A = V\Lambda V^T$
  for symmetric $A \in \mathbb{R}^{n\times n}$, where $V^TV = I_n$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$.

- **Singular Value Decomposition (SVD)**: $A = U\Sigma V^T$
  for any $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Here $U^TU = V^TV = I_n$, $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$,
  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

SVD: the most important matrix decomposition

- **Symmetric eigenvalue decomposition**: $A = V\Lambda V^T$
  for symmetric $A \in \mathbb{R}^{n\times n}$, where $V^TV = I_n$, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

- **Singular Value Decomposition (SVD)**: $A = U\Sigma V^T$
  for any $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Here $U^TU = V^TV = I_n$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$,
  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Terminologies:

- σ_i: *singular values* of A.
- *rank*(A): number of positive singular values.
- The columns of U: the *left singular vectors*, columns of V: *right singular vectors*

SVD: the most important matrix decomposition

- **Symmetric eigenvalue decomposition**: $A = V \Lambda V^T$
  for symmetric $A \in \mathbb{R}^{n \times n}$, where $V^T V = I_n$, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$.
- **Singular Value Decomposition (SVD)**: $A = U \Sigma V^T$
  for any $A \in \mathbb{R}^{m \times n}$, $m \geq n$. Here $U^T U = V^T V = I_n$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$,
  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

SVD proof: Take Gram matrix $A^T A$ and its eigendecomposition $A^T A = V \Lambda V^T$. Λ is nonnegative, and $(AV)^T (AV)$ is diagonal, so $AV = U\Sigma$ for some orthonormal U. Right-multiply V^T.

SVD: the most important matrix decomposition

- **Symmetric eigenvalue decomposition**: $A = V\Lambda V^T$
  for symmetric $A \in \mathbb{R}^{n\times n}$, where $V^TV = I_n$, $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$.
- **Singular Value Decomposition (SVD)**: $A = U\Sigma V^T$
  for any $A \in \mathbb{R}^{m\times n}$, $m \ge n$. Here $U^TU = V^TV = I_n$, $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_n)$,
  $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n \ge 0$.

SVD proof: Take Gram matrix A^TA and its eigendecomposition $A^TA = V\Lambda V^T$. Λ is nonnegative, and $(AV)^T(AV)$ is diagonal, so $AV = U\Sigma$ for some orthonormal U. Right-multiply V^T.

Full SVD: $A = U\begin{bmatrix}\Sigma \\ 0\end{bmatrix}V^T$ where $U \in \mathbb{R}^{m\times m}$ orthogonal

Example: computation

Let $A = \begin{bmatrix} -1 & -2 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$. To compute the SVD,

1. Compute the Gram matrix $A^T A = \begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix}$.
2. $\lambda(A^T A) = \{10, 2\}$ (e.g. via characteristic polynomial). The eigvec matrix is $V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ (e.g. via the null vectors of $A - \lambda I$). So $A^T A = V \Sigma^2 V^T$ where $\Sigma = \begin{bmatrix} \sqrt{10} & \\ & \sqrt{2} \end{bmatrix}$.
3. Let $U = AV\Sigma^{-1} = \begin{bmatrix} -3/\sqrt{20} & -1/2 \\ 3/\sqrt{20} & -1/2 \\ 1/\sqrt{20} & -1/2 \\ 1/\sqrt{20} & 1/2 \end{bmatrix}$, which is orthonormal. Thus $A = U\Sigma V^T$.

rank, column/row space, etc

From the SVD one gets

- rank r of $A \in \mathbb{R}^{m \times n}$: number of nonzero singular values $\sigma_i(A)$ (=# linearly indep. columns, rows)
  - We can always write $A = \sum_{i=1}^{\text{rank}(A)} \sigma_i u_i v_i^T$.
- column space (linear subspace spanned by vectors Ax): span of $U = [u_1, \ldots, u_r]$
- row space: row span of $v_1^T, \ldots, v_r^T$
- null space: $v_{r+1}, \ldots, v_n$

SVD and eigenvalue decomposition

- V eigvecs of A^TA
- U eigvecs (for nonzero eigvals) of AA^T (up to sign)
- $\sigma_i = \sqrt{\lambda_i(A^TA)}$
- Think of eigenvalues vs. SVD of symmetric matrices, unitary, skew-symmetric, normal matrices, triangular,...
- Jordan-Wieldant matrix $\left[\begin{smallmatrix} 0 & A \\ A^T & 0 \end{smallmatrix}\right]$: eigvals $\pm\sigma_i(A)$, and $m-n$ copies of 0. Eigvec matrix is $\left[\begin{smallmatrix} U & U & U_\perp \\ V & -V & 0 \end{smallmatrix}\right]$, $A^TU_\perp = 0$

Uniqueness etc

- U, V (clearly) not unique: ± 1 multiplication possible (but be careful—not arbitarily)

- When multiple singvals exist $\sigma_i = \sigma_{i+1}$, more degrees of freedom

- Extreme example: what is the SVD(s) of an orthogonal matrix?

Recap: spectral norm of matrix

$$\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2 = \sigma_1(A)$$

Proof: Use SVD

Recap: spectral norm of matrix

$$\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2 = \sigma_1(A)$$

Proof: Use SVD

$$\begin{aligned}
\|Ax\|_2 &= \|U\Sigma V^T x\|_2 \\
&= \|\Sigma V^T x\|_2 \quad \text{by unitary invariance} \\
&= \|\Sigma y\|_2 \quad \text{with } \|y\|_2 = 1 \\
&= \sqrt{\sum_{i=1}^n \sigma_i^2 y_i^2} \\
&\le \sqrt{\sum_{i=1}^n \sigma_1^2 y_i^2} = \sigma_1 \|y\|_2^2 = \sigma_1.
\end{aligned}$$

Recap: spectral norm of matrix

$$\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2 = \sigma_1(A)$$

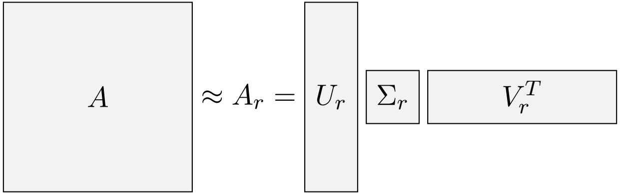
Proof: Use SVD

$$\begin{aligned}
\|Ax\|_2 &= \|U\Sigma V^T x\|_2 \\
&= \|\Sigma V^T x\|_2 \quad \text{by unitary invariance} \\
&= \|\Sigma y\|_2 \quad \text{with } \|y\|_2 = 1 \\
&= \sqrt{\sum_{i=1}^n \sigma_i^2 y_i^2} \\
&\leq \sqrt{\sum_{i=1}^n \sigma_1^2 y_i^2} = \sigma_1 \|y\|_2^2 = \sigma_1.
\end{aligned}$$

Frobenius norm: $\|A\|_F = \sqrt{\sum_i \sum_j |A_{ij}|^2} = \sqrt{\sum_{i=1}^n (\sigma_i(A))^2}$ (exercise)

Low-rank approximation of a matrix

Given $A \in \mathbb{R}^{m \times n}$, find A_r such that



- Storage savings (data compression)

Optimal low-rank approximation by SVD

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

$$A = \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_1 u_1 v_1} + \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_2 u_2 v_2} + \cdots + \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_n u_n v_n},$$

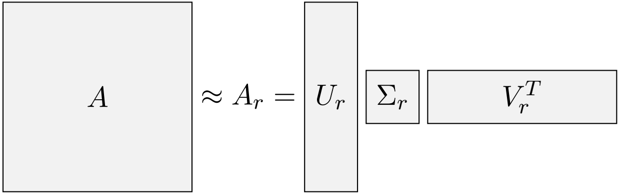
$$A_r = \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_1 u_1 v_1} + \cdots + \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}}_{\sigma_n u_r v_r}.$$

Optimal low-rank approximation by SVD

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \mathrm{diag}(\sigma_1, \dots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\mathrm{rank}(B)=r} \|A - B\|_2$$

- Good approximation if $\sigma_{r+1} \ll \sigma_1$:



- Optimality holds for any unitarily invariant norm
- Prominent application: PCA
- Many matrices have explicit or hidden low-rank structure (nonexaminable)

SVD optimality proof in spectral norm

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

SVD optimality proof in spectral norm

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

- Since $\operatorname{rank}(B) \leq r$, we can write $B = B_1 B_2^T$ where B_1, B_2 have r columns.

SVD optimality proof in spectral norm

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

- Since $\operatorname{rank}(B) \leq r$, we can write $B = B_1 B_2^T$ where B_1, B_2 have r columns.
- There exists orthonormal $W \in \mathbb{C}^{n\times(n-r)}$ s.t. $BW = 0$. Then $\|A - B\|_2 \geq \|(A - B)W\|_2 = \|AW\|_2 = \|U\Sigma(V^T W)\|_2$.

SVD optimality proof in spectral norm

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

- Since $\text{rank}(B) \le r$, we can write $B = B_1 B_2^T$ where B_1, B_2 have r columns.
- There exists orthonormal $W \in \mathbb{C}^{n \times (n-r)}$ s.t. $BW = 0$. Then $\|A - B\|_2 \ge \|(A - B)W\|_2 = \|AW\|_2 = \|U\Sigma(V^T W)\|_2$.
- Now since W is $(n-r)$-dimensional, there is an intersection between W and $[v_1, \dots, v_{r+1}]$, the $(r+1)$-dimensional subspace spanned by the leading $r+1$ left singular vectors ($[W, v_1, \dots, v_{r+1}]\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right] = 0$ has a solution; then Wx_1 is such a vector).

SVD optimality proof in spectral norm

Truncated SVD: $A_r = U_r \Sigma_r V_r^T$, $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$

$$\|A - A_r\|_2 = \sigma_{r+1} = \min_{\text{rank}(B)=r} \|A - B\|_2$$

- Since $\text{rank}(B) \leq r$, we can write $B = B_1 B_2^T$ where B_1, B_2 have r columns.
- There exists orthonormal $W \in \mathbb{C}^{n\times(n-r)}$ s.t. $BW = 0$. Then $\|A - B\|_2 \geq \|(A - B)W\|_2 = \|AW\|_2 = \|U\Sigma(V^T W)\|_2$.
- Now since W is $(n - r)$-dimensional, there is an intersection between W and $[v_1, \ldots, v_{r+1}]$, the $(r + 1)$-dimensional subspace spanned by the leading $r + 1$ left singular vectors ($[W, v_1, \ldots, v_{r+1}]\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right] = 0$ has a solution; then Wx_1 is such a vector).
- Then scale x_1, x_2 to have unit norm, and $\|U\Sigma V^T W x_1\|_2 = \|U_{r+1}\Sigma_{r+1} x_2\|_2$, Where U_{r+1}, Σ_{r+1} are leading $r + 1$ parts of U, Σ. Then $\|U_{r+1}\Sigma_{r+1} y_1\|_2 \geq \sigma_{r+1}$ can be verified directly.

SVD application: Netflix prize via matrix completion

Movie \ Person	A	B	C	D	E	F	G	H	I	J
Movie 1	?	3	2	4	1	?	2	?	3	4
Movie 2	0	0	?	0	?	?	0	?	0	0
Movie 3	4	3	1	2	2	1	?	2	?	3
Movie 4	?	?	1	2	?	1	2	2	4	3
Movie 5	2	2	0	2	1	1	1	1	?	2

Can we complete the matrix by finding the entries with "?"
thus give recommendations to each person

SVD application: Netflix prize via matrix completion

Movie \ Person	A	B	C	D	E	F	G	H	I	J
Movie 1	5	3	2	4	1	2	2	4	3	4
Movie 2	0	0	0	0	0	0	0	0	0	0
Movie 3	4	3	1	2	2	1	2	2	4	3
Movie 4	4	3	1	2	2	1	2	2	4	3
Movie 5	2	2	0	2	1	1	1	1	2	2

Can we complete the matrix by finding the entries with "?"
thus give recommendations to each person

SVD application: Netflix prize via matrix completion

Movie \ Person	A	B	C	D	E	F	G	H	I	J
Movie 1	5	3	2	4	1	2	2	4	3	4
Movie 2	0	0	0	0	0	0	0	0	0	0
Movie 3	4	3	1	2	2	1	2	2	4	3
Movie 4	4	3	1	2	2	1	2	2	4	3
Movie 5	2	2	0	2	1	1	1	1	2	2

Can we complete the matrix by finding the entries with "?"

thus give recommendations to each person

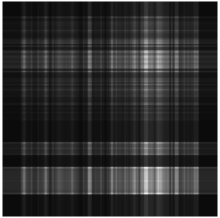
Yes! Key idea: low-rank matrix completion

Choose entries s.t. the matrix is low rank (interpretation: rank $\approx$ groups of people)

Low-rank approximation: image compression

grayscale image=matrix

original



rank 1

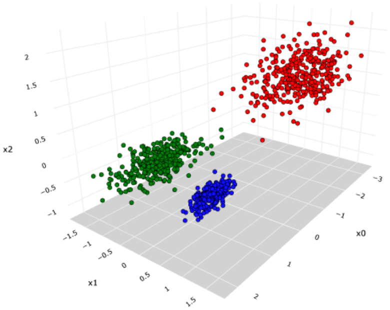
rank 5

rank 10

rank 20

rank 50

Low-rank approximation: PCA



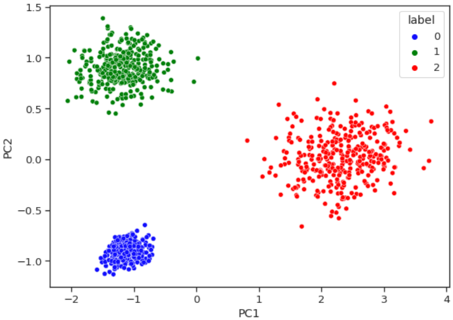


image from towardsdatascience.com

- Find 'most active' directions via SVD
- Project data onto low-dimensional space, then visualize, cluster, etc

Courant-Fischer minmax theorem

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{x^TAx}{x^Tx} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{x^TAx}{x^Tx} \right)$$

Analogously, for any rectangular $A \in \mathbb{C}^{m\times n}(m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \right).$$

- $\min_{x\in\mathcal{S},\|x\|_2=1} \|Ax\|_2 = \min_{Q^TQ=I_i,\|y\|_2=1} \|AQy\|_2 = \sigma_{\min}(AQ) = \sigma_i(AQ)$, where $\text{span}(Q) = \mathcal{S}$.
- C-F says $\sigma_i(A)$ is maximum possible value over all subspaces $\mathcal{S}$ of dimension i.

Courant-Fischer minmax theorem

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \right) \tag{1}$$

Analogously, for any rectangular $A \in \mathbb{C}^{m \times n} (m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \right). \tag{2}$$

Proof for (2):

Courant-Fischer minmax theorem

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \right) \tag{1}$$

Analogously, for any rectangular $A \in \mathbb{C}^{m \times n} (m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \right). \tag{2}$$

Proof for (2):

1. Fix S and let $V_i = [v_i, \ldots, v_n]$. We have $\dim(\mathcal{S}) + \dim(\text{span}(V_i)) = i + (n - i + 1) = n + 1$, so $\exists$intersection $w \in S \cap V_i$, $\|w\|_2 = 1$.

Courant-Fischer minmax theorem

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{x^TAx}{x^Tx} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{x^TAx}{x^Tx}\right) \tag{1}$$

Analogously, for any rectangular $A \in \mathbb{C}^{m\times n}(m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2}\right). \tag{2}$$

Proof for (2):

1. Fix S and let $V_i = [v_i, \dots, v_n]$. We have $\dim(\mathcal{S}) + \dim(\text{span}(V_i)) = i + (n-i+1) = n+1$, so $\exists$intersection $w \in S \cap V_i$, $\|w\|_2 = 1$.
2. For this w, $\|Aw\|_2 = \|\text{diag}(\sigma_i, \dots, \sigma_n)(V_i^Tw)\|_2 \leq \sigma_i$; thus $\sigma_i(A) \geq \min_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2}$.

Courant-Fischer minmax theorem

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{x^T A x}{x^T x} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{x^T A x}{x^T x} \right) \tag{1}$$

Analogously, for any rectangular $A \in \mathbb{C}^{m\times n} (m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \right). \tag{2}$$

Proof for (2):

1. Fix S and let $V_i = [v_i, \ldots, v_n]$. We have $\dim(\mathcal{S}) + \dim(\text{span}(V_i)) = i + (n-i+1) = n+1$, so $\exists$intersection $w \in S \cap V_i$, $\|w\|_2 = 1$.
2. For this w, $\|Aw\|_2 = \|\text{diag}(\sigma_i, \ldots, \sigma_n)(V_i^T w)\|_2 \leq \sigma_i$; thus $\sigma_i(A) \geq \min_{x\in\mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2}$.
3. For the reverse inequaltiy, take $S = [v_1, \ldots, v_i]$, for which $w = v_i$.

Weyl's inequality

ith largest eigval λ_i of symmetric/Hermitian A is (below $x \neq 0$)

$$\lambda_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{x^T A x}{x^T x} \right)$$

Analogously, for any rectangular $A \in \mathbb{C}^{m \times n} (m \geq n)$, we have

$$\sigma_i(A) = \max_{\dim \mathcal{S}=i} \min_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \quad \left(= \min_{\dim \mathcal{S}=n-i+1} \max_{x \in \mathcal{S}} \frac{\|Ax\|_2}{\|x\|_2} \right).$$

Corollary: Weyl's inequality (Proof: exercise)

- for singular values
  - $\sigma_i(A+E) \in \sigma_i(A) + [-\|E\|_2, \|E\|_2]$
  - Special case: $\|A\|_2 - \|E\|_2 \leq \|A+E\|_2 \leq \|A\|_2 + \|E\|_2$
- for symmetric eigenvalues $\lambda_i(A+E) \in \lambda_i(A) + [-\|E\|_2, \|E\|_2]$

Singvals and symmetric eigvals are insensitive to perturbation (well conditioned).

Nonsymmetric eigvals are different!

Eigenvalues of nonsymmetric matrices are sensitive

Consider eigenvalues of a Jordan block and its perturbation

$$J = \begin{bmatrix} 1 & 1 & & \\ & 1 & \ddots & \\ & & \ddots & 1 \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad J + E = \begin{bmatrix} 1 & 1 & & \\ & 1 & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 1 \end{bmatrix}$$

$\lambda(J) = 1$ (n copies), but $|\lambda(J+E) - 1| \approx \epsilon^{1/n}$

More applications of C-F

- $\sigma_i\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

More applications of C-F

- $\sigma_i\left(\begin{bmatrix}A_1\\A_2\end{bmatrix}\right) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

  Proof (sketch): LHS $= \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}, \|x\|_2=1} \left\|\begin{bmatrix}A_1\\A_2\end{bmatrix} x\right\|_2$, and for any x,

  $\left\|\begin{bmatrix}A_1\\A_2\end{bmatrix} x\right\|_2 \geq \max(\|A_1x\|_2, \|A_2x\|_2)$.

More applications of C-F

- $\sigma_i\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

  Proof (sketch): LHS $= \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S}, \|x\|_2=1} \left\| \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} x \right\|_2$, and for any x,

  $\left\| \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} x \right\|_2 \geq \max(\|A_1 x\|_2, \|A_2 x\|_2)$.
- $\sigma_i(\begin{bmatrix} A_1 & A_2 \end{bmatrix}) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

More applications of C-F

- $\sigma_i\left(\begin{bmatrix}A_1\\A_2\end{bmatrix}\right) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

  Proof (sketch): LHS $= \max_{\dim \mathcal{S}=i} \min_{x\in\mathcal{S},\|x\|_2=1} \left\|\begin{bmatrix}A_1\\A_2\end{bmatrix} x\right\|_2$, and for any x,

  $\left\|\begin{bmatrix}A_1\\A_2\end{bmatrix} x\right\|_2 \geq \max(\|A_1x\|_2, \|A_2x\|_2)$.
- $\sigma_i(\begin{bmatrix}A_1 & A_2\end{bmatrix}) \geq \max(\sigma_i(A_1), \sigma_i(A_2))$

  Proof: LHS $= \max_{\dim \mathcal{S}=i} \min_{\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right]\in\mathcal{S}, \left\|\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right]\right\|_2=1} \left\|\begin{bmatrix}A_1 & A_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\right\|_2$, while

  $\sigma_i(A_1) =$

  $\max_{\dim \mathcal{S}=i,\text{range}(\mathcal{S})\in\text{range}(\begin{bmatrix}I_n\\0\end{bmatrix})} \min_{\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right]\in\mathcal{S}, \left\|\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right]\right\|_2=1} \left\|\begin{bmatrix}A_1 & A_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\right\|_2$.

  Since the latter maximises over a smaller $\mathcal{S}$, the former is at least as big.

Matrix decompositions

- SVD $A = U\Sigma V^T$
- Eigenvalue decomposition $A = X\Lambda X^{-1}$
  - Normal: X unitary $X^*X = I$
  - Symmetric: X unitary and Λ real
- Jordan decomposition: $A = XJX^{-1}$, $J = \mathrm{diag}(\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix})$
- Schur decomposition $A = QTQ^*$: Q orthogonal, T upper triangular
- QR: Q orthonormal, U upper triangular
- LU: L lower triangular, U upper triangular

Red: Orthogonal decompositions, stable computation available

Solving $Ax = b$ via LU decomposition

If $A = LU$ is available

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} = \begin{bmatrix} * & & & & \\ * & * & & & \\ * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} = LU$$

solving $Ax = b$ can be done as follows:

1. Solve $Ly = b$ for y,
2. solve $Ux = y$ for x.

Each is a **triangular** system, which is easy to solve via forward (or backward) substitution for $Ly = b$ ($Ux = y$).

LU decomposition

Let $A \in \mathbb{R}^{n\times n}$. Suppose we can decompose (or factorise)

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} = \begin{bmatrix} * & & & & \\ * & * & & & \\ * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} = LU$$

L: lower triangular, U: upper triangular. How to find L, U?

LU decomposition

Let $A \in \mathbb{R}^{n\times n}$. Suppose we can decompose (or factorise)

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} = \begin{bmatrix} * & & & & \\ * & * & & & \\ * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} = LU$$

L: lower triangular, U: upper triangular. How to find L, U?

$$A = \begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix} + \begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix}}_{L_1U_1} + \underbrace{\begin{bmatrix} 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} 0 & * & * & * & * \end{bmatrix}}_{L_2U_2} + \begin{bmatrix} & & & & \\ & & & & \\ & & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix} = \cdots$$

LU decomposition cont'd

First step:

$$A = \underbrace{\begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix}}_{L_1 U_1} + \begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$$

algorithm:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & & & & \\ A_{31} & & & & \\ A_{41} & & & & \\ A_{51} & & & & \end{bmatrix} = \begin{bmatrix} L_{11} & & & & \\ L_{21} & & & & \\ L_{31} & & & & \\ L_{41} & & & & \\ L_{51} & & & & \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} + \begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & & & & \\ A_{21}/a & & & & \\ A_{31}/a & & & & \\ A_{41}/a & & & & \\ A_{51}/a & & & & \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}}_{= L_1 U_1 \quad (a = A_{11})} + \begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$$

LU decomposition cont'd 2

$$A = \begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix} + \begin{bmatrix} 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} 0 & * & * & * & * \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} 0 & 0 & * & * & * \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ * \\ * \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & * & * \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ * \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & * \end{bmatrix}$$

$$= L_1 U_1 + L_2 U_2 + L_3 U_3 + L_4 U_4 + L_5 U_5$$

$$= [L_1, L_2, \ldots, L_5] \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_5 \end{bmatrix} = \begin{bmatrix} * & & & & \\ * & * & & & \\ * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix}$$

(note: nonzero structure crucial in final equality)

Solving $Ax = b$ via LU

$$A = LU \in \mathbb{R}^{n\times n}$$

L: lower triangular, U: upper triangular

- Cost $\frac{2}{3}n^3$ flops (floating-point operations)
- For $Ax = b$,
  - *first solve* $Ly = b$, *then* $Ux = y$. Then $b = Ly = LUx = Ax$.
  - triangular solve is always backward stable: e.g. $(L + \Delta L)\hat{y} = b$ (see Higham's book)
- Pivoting crucial for numerical stability: $PA = LU$, where P: permutation matrix. Then stability means $\hat{L}\hat{U} = PA + \Delta A$
  - Even with pivoting, unstable examples exist, but still always stable in practice and used everywhere!
- Special case where $A \succ 0$ positive definite: $A = R^T R$, Cholesky factorization, ALWAYS stable, $\frac{1}{3}n^3$ flops

LU decomposition with pivots

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & & & & \\ A_{31} & & & & \\ A_{41} & & & & \\ A_{51} & & & & \end{bmatrix} = \begin{bmatrix} 1 \\ A_{21}/a \\ A_{31}/a \\ A_{41}/a \\ A_{51}/a \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} + \begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$$

Trouble if $a = A_{11} = 0$! e.g. no LU for $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ solution: pivot, permute rows s.t. largest entry of first (active) column is at top. $\Rightarrow PA = LU$, P: permutation matrix

- $PA = LU$ exists for any nonsingular A (exercise)
- for $Ax = b$, solve $LUx = P^T b$
- the nonzero structure of L_i, U_i is preserved under P
- cost still $\frac{2}{3}n^3 + O(n^2)$

Cholesky factorisation for $A \succ 0$

If $A \succ 0$ (symmetric positive definite (S)PD$\Leftrightarrow\lambda_i(A) > 0$), two simplifications:

- We can take $U_i = L_i^T =: R_i$ by symmetry $\Rightarrow \frac{1}{3}n^3$ flops
- No pivot needed

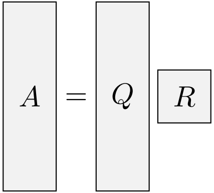
$$A = \underbrace{\begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix}}_{R_1 R_1^T} + \underbrace{\begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}}_{\text{also PD}}$$

Notes:

- diag(R) no longer 1's
- A can be written as $A = R^T R$ for some $R \in \mathbb{R}^{n\times n}$ iff $A \succeq 0$ $(\lambda_i(A) \geq 0)$
- Indefinite case: when $A = A^*$ but A not PD, $\exists$ $A = LDL^*$ where D diagonal (when $A \in \mathbb{R}^{n\times n}$, D can have 2×2 diagonal blocks), L has 1's on diagonal

QR factorisation

For any $A \in \mathbb{C}^{m \times n}$, $\exists$ factorisation



$Q \in \mathbb{R}^{m \times n}$: orthonormal, $R \in \mathbb{R}^{n \times n}$: upper triangular

- Many algorithms available: Gram-Schmidt, Householder, CholeskyQR, ...
- various applications: least-squares, orthogonalisation, computing SVD, manifold retraction...
- With Householder, pivoting $A = QRP$ not needed for numerical stability
  - but pivoting gives rank-revealing QR (nonexaminable)

QR via Gram-Schmidt

Gram-Schmidt: Given $A = [a_1, a_2, \ldots, a_n] \in \mathbb{R}^{m\times n}$ (assume full rank $\text{rank}(A) = n$), find orthonormal $[q_1, \ldots, q_n]$ s.t. $\text{span}(q_1, \ldots, q_n) = \text{span}(a_1, \ldots, a_n)$

G-S process: $q_1 = \frac{a_1}{\|a_1\|}$, then $\tilde{q}_2 = a_2 - q_1 q_1^T a_2$, $q_2 = \frac{\tilde{q}_2}{\|\tilde{q}_2\|}$,
repeat for $j = 3, \ldots, n$: $\tilde{q}_j = a_j - \sum_{i=1}^{j-1} q_i q_i^T a_j$, $q_j = \frac{\tilde{q}_j}{\|\tilde{q}_j\|}$.

QR via Gram-Schmidt

Gram-Schmidt: Given $A = [a_1, a_2, \dots, a_n] \in \mathbb{R}^{m \times n}$ (assume full rank rank$(A) = n$), find orthonormal $[q_1, \dots, q_n]$ s.t. span$(q_1, \dots, q_n) =$ span$(a_1, \dots, a_n)$

G-S process: $q_1 = \frac{a_1}{\|a_1\|}$, then $\tilde{q}_2 = a_2 - q_1 q_1^T a_2$, $q_2 = \frac{\tilde{q}_2}{\|\tilde{q}_2\|}$,
repeat for $j = 3, \dots, n$: $\tilde{q}_j = a_j - \sum_{i=1}^{j-1} q_i q_i^T a_j$, $q_j = \frac{\tilde{q}_j}{\|\tilde{q}_j\|}$.

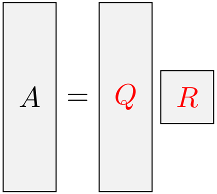
This gives QR! Let $r_{ij} = q_i^T a_j$ $(i \neq j)$ and $r_{jj} = \|a_j - \sum_{i=1}^{j-1} r_{ij} q_i\|$,

$$q_1 = \frac{a_1}{r_{11}}, \quad q_2 = \frac{a_2 - r_{12} q_1}{r_{22}}, \quad q_j = \frac{a_j - \sum_{i=1}^{j-1} r_{ij} q_i}{r_{jj}}$$

$$\Leftrightarrow$$

$$a_1 = r_{11} q_1, \quad a_2 = r_{12} q_1 + r_{22} q_2, \quad a_j = r_{1j} q_1 + r_{2j} q_2 + \cdots + r_{jj} q_j$$

$$\Leftrightarrow$$

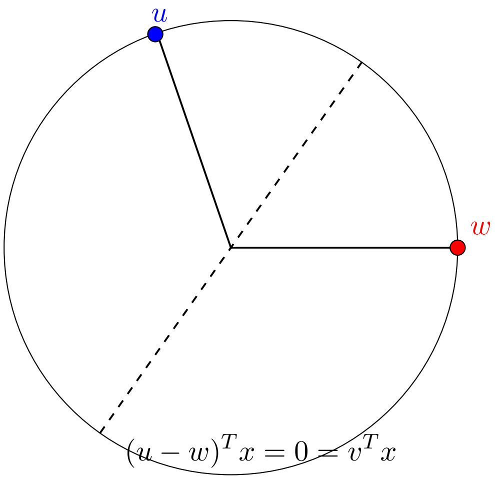


- But this isn't the recommended way to do QR; numerically unstable

Householder reflectors

$$H = I - 2vv^T, \qquad \|v\| = 1$$

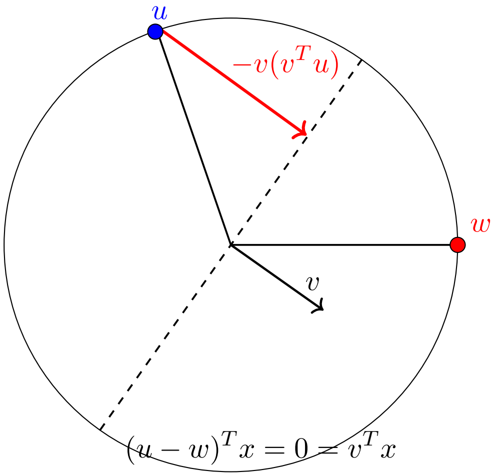
- H orthogonal and symmetric: $H^T H = H^2 = I$, eigvals 1 ($n-1$ copies) and -1 (1 copy)
- For any given $u, w \in \mathbb{R}^n$ s.t. $\|u\| = \|w\|$ and $u \neq v$, $H = I - 2vv^T$ with $v = \frac{w-u}{\|w-u\|}$ gives $Hu = w$ ($\Leftrightarrow u = Hw$, thus 'reflector')
- We'll use this mostly for $w = [*, 0, 0, \ldots, 0]^T$



Householder reflectors

$$H = I - 2vv^T, \qquad \|v\| = 1$$

- H orthogonal and symmetric: $H^T H = H^2 = I$, eigvals 1 ($n-1$ copies) and -1 (1 copy)
- For any given $u, w \in \mathbb{R}^n$ s.t. $\|u\| = \|w\|$ and $u \neq v$, $H = I - 2vv^T$ with $v = \frac{w-u}{\|w-u\|}$ gives $Hu = w$ ($\Leftrightarrow u = Hw$, thus 'reflector')
- We'll use this mostly for $w = [*, 0, 0, \dots, 0]^T$



Householder reflectors for QR

Householder reflectors:

$$H = I - 2vv^T, \qquad v = \frac{x - \|x\|_2 e}{\|x - \|x\|_2 e\|_2}, \qquad e = [1, 0, \ldots, 0]^T$$

satisfies $Hx = [\|x\|, 0, \ldots, 0]^T$

Householder reflectors for QR

Householder reflectors:

$$H = I - 2vv^T, \qquad v = \frac{x - \|x\|_2 e}{\|x - \|x\|_2 e\|_2}, \qquad e = [1, 0, \dots, 0]^T$$

satisfies $Hx = [\|x\|, 0, \dots, 0]^T$

$\Rightarrow$ To do QR, find H_1 s.t. $H_1 a_1 = \begin{bmatrix} \|a_1\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$,

repeat to get $H_n \cdots H_2 H_1 A = R$ upper triangular, then
$A = (H_1 \cdots H_{n-1} H_n) R = QR$

Householder QR factorisation, diagram

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

Apply sequence of Householder reflectors

$$H_1 A = (I - 2v_1 v_1^T)A = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \\ & * & * & * \end{bmatrix}, \qquad H_2 H_1 A = (I - 2v_2 v_2^T)H_1 A = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & * & * \\ & & * & * \\ & & * & * \end{bmatrix},$$

$$H_3 H_2 H_1 A = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & * & * \\ & & & * \\ & & & * \end{bmatrix}, \qquad H_n \cdots H_3 H_2 H_1 A = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & * & * \\ & & & * \\ & & & \end{bmatrix},$$

Note $v_k = [\underbrace{0, 0, \ldots, 0}_{k-1 \text{ 0's}}, *, *, \ldots, *]^T$

Householder QR factorisation, example

$$A = \begin{bmatrix} 0.302 & -0.629 & 2.178 & 0.164 \\ 0.400 & -1.204 & 1.138 & 0.748 \\ -0.930 & -0.254 & -2.497 & -0.273 \\ -0.177 & -1.429 & 0.441 & 1.576 \\ -2.132 & -0.021 & -1.398 & -0.481 \\ 1.145 & -0.561 & -0.255 & 0.328 \end{bmatrix}$$

Householder QR factorisation, example

$$H_1 A = \begin{bmatrix} 2.647 & -0.295 & 2.284 & 0.652 \\ 0 & -1.261 & 1.120 & 0.665 \\ 0 & -0.121 & -2.455 & -0.080 \\ 0 & -1.403 & 0.449 & 1.613 \\ 0 & 0.283 & -1.301 & -0.038 \\ 0 & -0.724 & -0.307 & 0.090 \end{bmatrix}$$

Householder QR factorisation, example

$$H_2 H_1 A = \begin{bmatrix} 2.647 & -0.295 & 2.284 & 0.652 \\ 0 & 2.044 & -0.925 & -1.550 \\ 0 & 0 & -2.530 & -0.161 \\ 0 & 0 & -0.419 & 0.673 \\ 0 & 0 & -1.126 & 0.152 \\ 0 & 0 & -0.755 & -0.395 \end{bmatrix}$$

Householder QR factorisation, example

$$H_3 H_2 H_1 A = \begin{bmatrix} 2.647 & -0.295 & 2.284 & 0.652 \\ 0 & 2.044 & -0.925 & -1.550 \\ 0 & 0 & 2.901 & 0.087 \\ 0 & 0 & 0 & 0.692 \\ 0 & 0 & 0 & 0.203 \\ 0 & 0 & 0 & -0.361 \end{bmatrix}$$

Householder QR factorisation, example

$$H_4 H_3 H_2 H_1 A = \begin{bmatrix} 2.647 & -0.295 & 2.284 & 0.652 \\ 0 & 2.044 & -0.925 & -1.550 \\ 0 & 0 & 2.901 & 0.087 \\ 0 & 0 & 0 & 0.806 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Householder QR factorisation

$$H_n \cdots H_2 H_1 A = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & * & * \\ & & & * \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

$\Leftrightarrow A = (H_1^T \cdots H_{n-1}^T H_n^T) \begin{bmatrix} R \\ 0 \end{bmatrix} =: Q_F \begin{bmatrix} R \\ 0 \end{bmatrix}$ (**full** QR; Q_F is square orthogonal)

Writing $Q_F = [Q\ Q_\perp]$ where $Q \in \mathbb{R}^{m \times n}$ orthonormal, $A = QR$ (**'thin'** QR or just QR)

Properties

- Cost $\frac{4}{3}n^3$ flops with Householder-QR (twice that of LU when $m = n$; if $m > n$, $2mn^2 - \frac{2}{3}n^3$)
- Unconditionally backward stable: $\hat{Q}\hat{R} = A + \Delta A$, $\|\hat{Q}^T\hat{Q} - I\|_2 = \epsilon$ (next lec)
- Constructive proof for $A = QR$ existence
- To solve $Ax = b$, solve $Rx = Q^T b$ via triangle solve.

  $\rightarrow$ Excellent method, but twice slower than LU (so rarely used)

Givens rotation

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \quad c^2 + s^2 = 1$$

Designed to 'zero' one element at a time. E.g. QR for upper Hessenberg matrix

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}, \quad G_1 A = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}, \quad G_2 G_1 A = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix},$$

$$G_3 G_2 G_1 A = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & * & * \end{bmatrix}, \quad G_4 G_3 G_2 G_1 A = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix} =: R$$

$\Leftrightarrow A = G_1^T G_2^T G_3^T G_4^T R$ is the QR factorisation.

- G acts locally on two rows (two columns if right-multiplied)
- Non-neighboring rows/cols allowed

Least-squares problem

Given $A \in \mathbb{R}^{m \times n}, m \geq n$ and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$ s.t.

$$\min_x \left\| A x - b \right\|_2$$

- More data than degrees of freedom
- 'Overdetermined' linear system; $Ax = b$ usually impossible
- Thus minimise $\|Ax - b\|$; usually $\|Ax - b\|_2$ but sometimes e.g. $\|Ax - b\|_1$ of interest (we focus on $\|Ax - b\|_2$)
- Assume full rank rank$(A) = n$; this makes solution unique

Least-squares problem via QR

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, m \geq n$$

Least-squares problem via QR

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, m \geq n$$

Let $A = [Q\ Q_\perp]\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right] = Q_F\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ be 'full' QR factorization. Then

$$\|Ax - b\|_2 = \|Q_F^T(Ax - b)\|_2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} Q^T b \\ Q_\perp^T b \end{bmatrix} \right\|_2$$

so $x = R^{-1}Q^T b$ is the solution. This also gives algorithm:

Least-squares problem via QR

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, m \geq n$$

Let $A = [Q\ Q_\perp]\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right] = Q_F\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ be 'full' QR factorization. Then

$$\|Ax - b\|_2 = \|Q_F^T(Ax - b)\|_2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} Q^T b \\ Q_\perp^T b \end{bmatrix} \right\|_2$$

so $x = R^{-1}Q^Tb$ is the solution. This also gives algorithm:

1. Compute **thin** QR factorization $A = QR$
2. Solve linear system $Rx = Q^Tb$.

Least-squares problem via QR

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, m \geq n$$

Let $A = [Q\ Q_\perp]\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right] = Q_F\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ be 'full' QR factorization. Then

$$\|Ax - b\|_2 = \|Q_F^T(Ax - b)\|_2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} Q^T b \\ Q_\perp^T b \end{bmatrix} \right\|_2$$

so $x = R^{-1}Q^T b$ is the solution. This also gives algorithm:

1. Compute **thin** QR factorization $A = QR$
2. Solve linear system $Rx = Q^T b$.

- This is backward stable: computed $\hat{x}$ solution for $\min_x \|(A + \Delta A)x + (b + \Delta b)\|_2$ (see Higham's book Ch.20)
- Unlike square system $Ax = b$, one really needs QR: LU won't do the job

Normal equation: Cholesky-based least-squares solver

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, m \geq n$$

$x = R^{-1}Q^T b$ is the solution $\Leftrightarrow$ x solution for $n \times n$ **normal equation**

$$(A^T A)x = A^T b$$

- $A^T A \succeq 0$ (always) and $A^T A \succ 0$ if $\text{rank}(A) = n$; then PD linear system; use Cholesky to solve.
- Fast! but NOT backward stable; $\kappa_2(A^T A) = (\kappa_2(A))^2$ where $\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$ **condition number** (next lecture)

Application: regression/function approximation

Given function $f : [-1, 1] \to \mathbb{R}$,

Consider approximating via polynomial $f(x) \approx p(x) = \sum_{i=0} c_i x^i$.

Very common technique: **Regression**

1. Sample f at points $\{z_i\}_{i=1}^m$, and
2. Find coefficients c defined by Vandermonde system $Ac \approx f$,

$$\begin{bmatrix} 1 & z_1 & \cdots & z_1^n \\ 1 & z_2 & \cdots & z_2^n \\ \vdots & \vdots & & \vdots \\ 1 & z_m & \cdots & z_m^n \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \approx \begin{bmatrix} f(z_1) \\ f(z_2) \\ \vdots \\ f(z_m) \end{bmatrix}.$$

- Numerous applications, e.g. in statistics, numerical analysis, approximation theory, data analysis!

Example: regression to denoise

$$m\left\{\begin{bmatrix}1 & z_1 & \cdots & z_1^n\\ 1 & z_2 & \cdots & z_2^n\\ \vdots & \vdots & & \vdots\\ \vdots & \vdots & & \vdots\\ \vdots & \vdots & & \vdots\\ 1 & z_{m-1} & \cdots & z_{m-1}^n\\ 1 & z_m & \cdots & z_m^n\end{bmatrix}\right. \begin{bmatrix}c_0\\ \vdots\\ c_n\end{bmatrix} \approx \begin{bmatrix}f(z_1)\\ f(z_2)\\ \vdots\\ \vdots\\ \vdots\\ f(z_{m-1})\\ f(z_m)\end{bmatrix}.$$

$f(z_i) = \frac{1}{25x^2+1} + \delta$, δ: iid noise $\sim \mathcal{N}(0,1)$

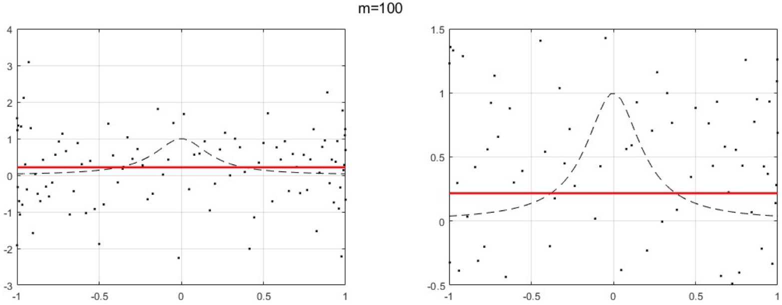
See [Matsuda-N. 2024]

Example: regression to denoise

$$
m\left\{\begin{bmatrix} 1 & z_1 & \cdots & z_1^n \\ 1 & z_2 & \cdots & z_2^n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & z_{m-1} & \cdots & z_{m-1}^n \\ 1 & z_m & \cdots & z_m^n \end{bmatrix}\right. \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \approx \begin{bmatrix} f(z_1) \\ f(z_2) \\ \vdots \\ \vdots \\ \vdots \\ f(z_{m-1}) \\ f(z_m) \end{bmatrix}.
$$

$f(z_i) = \frac{1}{25x^2+1} + \delta$, δ: iid noise $\sim \mathcal{N}(0,1)$

See [Matsuda-N. 2024]



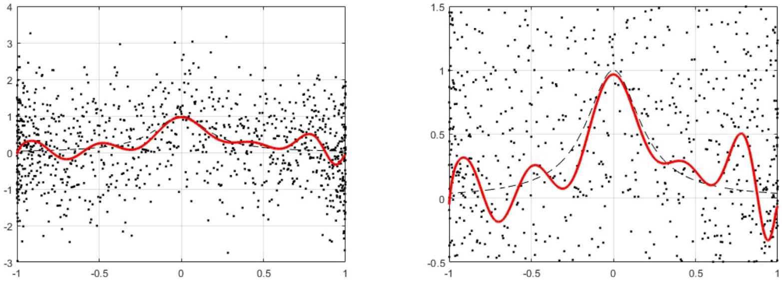
Example: regression to denoise

$$
m\left\{\begin{bmatrix} 1 & z_1 & \cdots & z_1^n \\ 1 & z_2 & \cdots & z_2^n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & z_{m-1} & \cdots & z_{m-1}^n \\ 1 & z_m & \cdots & z_m^n \end{bmatrix}\right. \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \approx \begin{bmatrix} f(z_1) \\ f(z_2) \\ \vdots \\ \vdots \\ \vdots \\ f(z_{m-1}) \\ f(z_m) \end{bmatrix}.
$$

$f(z_i) = \frac{1}{25x^2+1} + \delta$, δ: iid noise $\sim \mathcal{N}(0,1)$

See [Matsuda-N. 2024]

m=1000



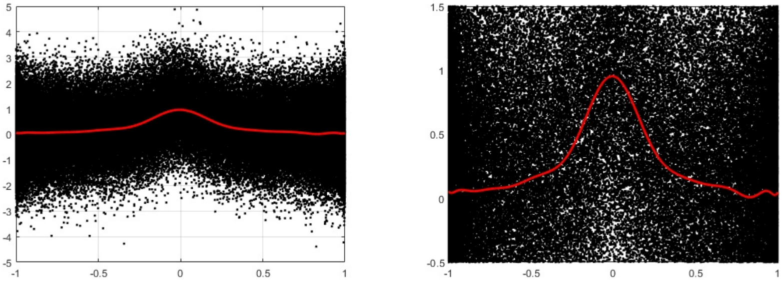
Example: regression to denoise

$$
m\left\{\begin{bmatrix} 1 & z_1 & \cdots & z_1^n \\ 1 & z_2 & \cdots & z_2^n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & z_{m-1} & \cdots & z_{m-1}^n \\ 1 & z_m & \cdots & z_m^n \end{bmatrix}\right. \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \approx \begin{bmatrix} f(z_1) \\ f(z_2) \\ \vdots \\ \vdots \\ \vdots \\ f(z_{m-1}) \\ f(z_m) \end{bmatrix}.
$$

$f(z_i) = \frac{1}{25x^2+1} + \delta$, δ: iid noise $\sim \mathcal{N}(0,1)$

See [Matsuda-N. 2024]

m=100000



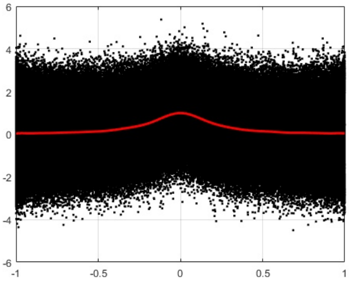
Example: regression to denoise

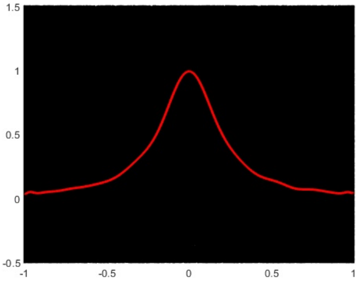
$$m\left\{\begin{bmatrix} 1 & z_1 & \cdots & z_1^n \\ 1 & z_2 & \cdots & z_2^n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & z_{m-1} & \cdots & z_{m-1}^n \\ 1 & z_m & \cdots & z_m^n \end{bmatrix}\right. \begin{bmatrix} c_0 \\ \vdots \\ c_n \end{bmatrix} \approx \begin{bmatrix} f(z_1) \\ f(z_2) \\ \vdots \\ \vdots \\ \vdots \\ f(z_{m-1}) \\ f(z_m) \end{bmatrix}.$$

$f(z_i) = \frac{1}{25x^2+1} + \delta$, δ: iid noise $\sim \mathcal{N}(0,1)$

See [Matsuda-N. 2024]

m=1000000





Numerical stability

Question: Can a computed result be trusted?

e.g. is $Ax = b$ always solved correctly via the LU algorithm?

Numerical stability

Question: Can a computed result be trusted?

e.g. is $Ax = b$ always solved correctly via the LU algorithm?

- The situation is complicated. For example, let $A = U\Sigma V^T$, where $U = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & \\ & 10^{-15} \end{bmatrix}$, $V = I$, and let $b = A\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ (i.e., $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$).

Numerical stability

Question: Can a computed result be trusted?

e.g. is $Ax = b$ always solved correctly via the LU algorithm?

- The situation is complicated. For example, let
  $A = U\Sigma V^T$, where $U = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & \\ & 10^{-15} \end{bmatrix}$, $V = I$, and let
  $b = A\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ (i.e., $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$).
  In MATLAB, `x = A\b` outputs $\begin{bmatrix} 1.0000 \\ 0.94206 \end{bmatrix}$
- Did something go wrong?

Numerical stability

Question: Can a computed result be trusted?

e.g. is $Ax = b$ always solved correctly via the LU algorithm?

- The situation is complicated. For example, let
  $A = U\Sigma V^T$, where $U = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, $\Sigma = \begin{bmatrix}1 & \\ & 10^{-15}\end{bmatrix}$, $V = I$, and let
  $b = A\begin{bmatrix}1\\1\end{bmatrix}$ (i.e., $x = \begin{bmatrix}1\\1\end{bmatrix}$).
  In MATLAB, $\texttt{x} = \texttt{A}\backslash\texttt{b}$ outputs $\begin{bmatrix}1.0000\\0.94206\end{bmatrix}$
- Did something go wrong?
  NO—this is a ramification of ill-conditioning, not instability
- In fact, $\|\texttt{Ax}-\texttt{b}\|_2 (= \|A\hat{x}-b\|_2) \approx 10^{-16}$

(After this section, make sure you can explain what happened above!)

Floating-point arithmetic

- Computers store number in base 2 with finite/fixed memory (bits)
- Irrational numbers are stored inexactly, e.g. $1/3 \approx 0.333...$
- Calculations are rounded to nearest floating-point number (rounding error)
- Thus the accuracy of the final error is nontrivial

Two examples with MATLAB

- $((\texttt{sqrt}(2))^2 - 2) * \texttt{1e15} = 0.4441$ (should be 0..)
- $\sum_{n=1}^{\infty} \frac{1}{n} \approx 30$ (should be ∞..)

An important (but not main) part of numerical analysis/NLA is to study the effect of rounding errors

Best reference: Higham's book (2002)

Conditioning and stability

- Conditioning is the sensitivity of a problem (e.g. of finding $y = f(x)$ given x) to perturbation in inputs, i.e., how large $\kappa := \sup_{\delta x} \|f(x+\delta x) - f(x)\|/\|\delta x\|$ is in the limit $\delta x \to 0$.
  (this is *absolute* condition number; equally important is *relative* condition number $\kappa_r := \lim_{\|\delta x\|_2 \to 0} \sup_{\delta x} \frac{\|f(x+\delta x)-f(x)\|}{\|f(x)\|} / \frac{\|\delta x\|}{\|x\|}$)

- (Backward) Stability is a property of an algorithm, which describes if the computed solution $\hat{y}$ is a 'good' solution, in that it is an exact solution of a nearby input, that is, $\hat{y} = f(x + \Delta x)$ for a small Δx.

Conditioning and stability

- Conditioning is the sensitivity of a problem (e.g. of finding $y = f(x)$ given x) to perturbation in inputs, i.e., how large $\kappa := \sup_{\delta x} \|f(x+\delta x) - f(x)\|/\|\delta x\|$ is in the limit $\delta x \to 0$.
  (this is *absolute* condition number; equally important is *relative* condition number $\kappa_r := \lim_{\|\delta x\|_2 \to 0} \sup_{\delta x} \frac{\|f(x+\delta x)-f(x)\|}{\|f(x)\|} / \frac{\|\delta x\|}{\|x\|}$)
- (Backward) Stability is a property of an algorithm, which describes if the computed solution $\hat{y}$ is a 'good' solution, in that it is an exact solution of a nearby input, that is, $\hat{y} = f(x + \Delta x)$ for a small Δx.

If problem is ill-conditioned $\kappa \gg 1$, then blame the problem not the algorithm

Notation/convention: $\hat{x}$ denotes a computed approximation to x (e.g. of $x = A^{-1}b$)
ϵ denotes a small term $O(u)$, on the order of unit roundoff/working precision; so we write e.g. $u, 10u, (m+n)u, mnu$ all as ϵ

- Consequently (in this lecture/discussion) norm choice does not matter today

Numerical stability: backward stability

For computational task $Y = f(X)$ and computed approximant $\hat{Y}$,

- Ideally, error $\|Y - \hat{Y}\|/\|Y\| = \epsilon$: seldom true

  (u: unit roundoff, $\approx 10^{-16}$ in standard double precision)
- Good alg. has Backward stability $\hat{Y} = f(X + \Delta X)$, $\frac{\|\Delta X\|}{\|X\|} = \epsilon$ "exact solution of slightly wrong input "

Numerical stability: backward stability

For computational task $Y = f(X)$ and computed approximant $\hat{Y}$,

- Ideally, error $\|Y - \hat{Y}\|/\|Y\| = \epsilon$: seldom true
  (u: unit roundoff, $\approx 10^{-16}$ in standard double precision)
- Good alg. has Backward stability $\hat{Y} = f(X + \Delta X)$, $\frac{\|\Delta X\|}{\|X\|} = \epsilon$ "exact solution of slightly wrong input "

- Justification: **Input (matrix) is usually inexact anyway!** $f(X + \Delta X)$ is just as good at $f(X)$ at approximating $f(X_*)$ where $\|\Delta X\| = O(\|X - X_*\|)$
  We shall 'settle with' such solution, though it may not mean $\hat{Y} - Y$ is small

- Forward stability $\|Y - \hat{Y}\|/\|Y\| = O(\kappa(f)u)$ "error is as small as backward stable alg." (sometimes used to mean small error; we follow Higham's book [2002])

Backward stable+well conditioned=accurate solution

Suppose

- $Y = f(X)$ computed backward stably i.e., $\hat{Y} = f(X + \Delta X)$, $\|\Delta X\| = \epsilon$.

Then with conditioning $\kappa = \lim_{\|\delta x\|_2 \to 0} \sup_{\delta x} \frac{\|f(X) - f(X + \Delta X)\|}{\|\Delta X\|}$,

$$\|\hat{Y} - Y\| \lesssim \kappa\epsilon$$

(relative version possible)

Backward stable+well conditioned=accurate solution

Suppose

- $Y = f(X)$ computed backward stably i.e., $\hat{Y} = f(X + \Delta X)$, $\|\Delta X\| = \epsilon$.

Then with conditioning $\kappa = \lim_{\|\delta x\|_2 \to 0} \sup_{\delta x} \frac{\|f(X) - f(X+\Delta X)\|}{\|\Delta X\|}$,

$$\|\hat{Y} - Y\| \lesssim \kappa\epsilon$$

(relative version possible) 'proof':

$$\|\hat{Y} - Y\| = \|f(X + \Delta X) - f(X)\| \lesssim \kappa\|\Delta X\|\|f(X)\| = \kappa\epsilon$$

Backward stable+well conditioned=accurate solution

Suppose

- $Y = f(X)$ computed backward stably i.e., $\hat{Y} = f(X + \Delta X)$, $\|\Delta X\| = \epsilon$.

Then with conditioning $\kappa = \lim_{\|\delta x\|_2 \to 0} \sup_{\delta x} \frac{\|f(X) - f(X+\Delta X)\|}{\|\Delta X\|}$,

$$\|\hat{Y} - Y\| \lesssim \kappa\epsilon$$

(relative version possible) 'proof':

$$\|\hat{Y} - Y\| = \|f(X + \Delta X) - f(X)\| \lesssim \kappa\|\Delta X\|\|f(X)\| = \kappa\epsilon$$

If well-conditioned $\kappa = O(1)$, good accuracy! Important examples:

- Well-conditioned linear system $Ax = b$, $\kappa_2(A) \approx 1$
- Eigenvalues of symmetric matrices (via Weyl's bound $\lambda_i(A + E) \in \lambda_i(A) + [-\|E\|_2, \|E\|_2]$)
- Singular values of any matrix $\sigma_i(A + E) \in \sigma_i(A) + [-\|E\|_2, \|E\|_2]$

Note: eigvecs/singvecs can be highly ill-conditioned

Matrix condition number

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} (\geq 1)$$

e.g. for linear systems. (when A is $m \times n (m > n)$, $\kappa_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)}$) A backward stable soln for $Ax = b$, s.t. $(A + \Delta A)\hat{x} = b$ satisfies, assuming backward stability $\|\Delta A\| \leq \epsilon \|A\|$ and $\kappa_2(A) \ll \epsilon^{-1}$ (so $\|A^{-1}\Delta A\| \ll 1$),

$$\frac{\|\hat{x} - x\|}{\|x\|} \lesssim \epsilon \kappa_2(A)$$

Matrix condition number

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} (\geq 1)$$

e.g. for linear systems. (when A is $m \times n (m > n)$, $\kappa_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)}$) A backward stable soln for $Ax = b$, s.t. $(A + \Delta A)\hat{x} = b$ satisfies, assuming backward stability $\|\Delta A\| \leq \epsilon \|A\|$ and $\kappa_2(A) \ll \epsilon^{-1}$ (so $\|A^{-1}\Delta A\| \ll 1$),

$$\frac{\|\hat{x} - x\|}{\|x\|} \lesssim \epsilon \kappa_2(A)$$

'proof': By Neumann series

$$(A + \Delta A)^{-1} = (A(I + A^{-1}\Delta A))^{-1} = (I - A^{-1}\Delta A + O(\|A^{-1}\Delta A\|^2))A^{-1}$$

So $\hat{x} = (A + \Delta A)^{-1} b = A^{-1}b - A^{-1}\Delta A A^{-1} b + O(\|A^{-1}\Delta A\|^2) = x - A^{-1}\Delta A x + O(\|A^{-1}\Delta A\|^2)$, Hence

$$\|x - \hat{x}\| \lesssim \|A^{-1}\Delta A x\| \leq \|A^{-1}\| \|\Delta A\| \|x\| \leq \epsilon \|A\| \|A^{-1}\| \|x\| = \epsilon \kappa_2(A) \|x\|$$

Backward stability of triangular systems

Recall $Ax = b$ via $Ly = b$, $Ux = y$ (triangular systems).

The computed solution $\hat{x}$ for a (upper/lower) triangular linear system $Rx = b$ solved via back/forward substitution is backward stable, i.e., it satisfies

$$(R + \Delta R)\hat{x} = b, \qquad \|\Delta R\| = O(\epsilon\|R\|).$$

Proof: Trefethen-Bau or Higham (nonexaminable but interesting)

- backward error can be bounded componentwise
- this means $\|\hat{x} - x\|/\|x\| \leq \epsilon\kappa_2(R)$
  - (unavoidably) poor worst-case (and attainable) bound when ill-conditioned
  - often better with triangular systems

(In)stability of $Ax = b$ via LU with pivots

Fact (proof nonexaminable): Computed $\hat{L}\hat{U}$ satisfies $\frac{\|\hat{L}\hat{U}-A\|}{\|L\|\|U\|} = \epsilon$

(note: not $\frac{\|\hat{L}\hat{U}-A\|}{\|A\|} = \epsilon$)

- If $\|L\|\|U\| = O(\|A\|)$, then $(L + \Delta L)(U + \Delta U)\hat{x} = b$
  $\Rightarrow$ $\hat{x}$ backward stable solution (exercise)

(In)stability of $Ax = b$ via LU with pivots

Fact (proof nonexaminable): Computed $\hat{L}\hat{U}$ satisfies $\frac{\|\hat{L}\hat{U}-A\|}{\|L\|\|U\|} = \epsilon$

(note: not $\frac{\|\hat{L}\hat{U}-A\|}{\|A\|} = \epsilon$)

- If $\|L\|\|U\| = O(\|A\|)$, then $(L + \Delta L)(U + \Delta U)\hat{x} = b$
  $\Rightarrow$ $\hat{x}$ backward stable solution (exercise)

Question: Does $LU = A + \Delta A$ or $LU = PA + \Delta A$ with $\|\Delta A\| = \epsilon\|A\|$ hold?

Without pivot $(P = I)$: $\|L\|\|U\| \gg \|A\|$ unboundedly (e.g. $\left[\begin{smallmatrix}\epsilon & 1\\ 1 & 1\end{smallmatrix}\right]$) unstable

(In)stability of $Ax = b$ via LU with pivots

Fact (proof nonexaminable): Computed $\hat{L}\hat{U}$ satisfies $\frac{\|\hat{L}\hat{U}-A\|}{\|L\|\|U\|} = \epsilon$

(note: not $\frac{\|\hat{L}\hat{U}-A\|}{\|A\|} = \epsilon$)

- If $\|L\|\|U\| = O(\|A\|)$, then $(L + \Delta L)(U + \Delta U)\hat{x} = b$
  $\Rightarrow \hat{x}$ backward stable solution (exercise)

Question: Does $LU = A + \Delta A$ or $LU = PA + \Delta A$ with $\|\Delta A\| = \epsilon\|A\|$ hold?

Without pivot $(P = I)$: $\|L\|\|U\| \gg \|A\|$ unboundedly (e.g. $\left[\begin{smallmatrix} \epsilon & 1 \\ 1 & 1 \end{smallmatrix}\right]$) unstable

With pivots:

- Worst-case: $\|L\|\|U\| \gg \|A\|$ grows exponentially with n, unstable
  - growth governed by that of $\|L\|\|U\|/\|A\| \Rightarrow \|U\|/\|A\|$
- In practice (average case): perfectly stable
  - Hence this is how $Ax = b$ is solved, despite alternatives with guaranteed stability exist (but slower; e.g. via SVD, or QR (next))

Resolution/explanation: among biggest open problems in numerical linear algebra!

Examples of stability and instability

Forthcoming examples: nonexaminable

Stability of Cholesky for $A \succ 0$

Cholesky $A = R^T R$ for $A \succ 0$

- succeeds without pivot (active matrix is always positive definite)
- R never contains entries $> \sqrt{\|A\|_2}$

$$A = \underbrace{\begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * & * & * & * & * \end{bmatrix}}_{R_1 R_1^T} + \underbrace{\begin{bmatrix} & & & & \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}}_{\text{also PSD}}$$

(exercise: show $\|R_1\|_2 \leq \sqrt{\|A\|_2}$)

$\Rightarrow$ backward stable! Hence positive definite linear system $Ax = b$ stable via Cholesky

(In)stability of Gram-Schmidt

- Gram-Schmidt is subtle
  - plain (classical) version: $\|\hat{Q}^T\hat{Q} - I\| \leq \epsilon(\kappa_2(A))^2$
  - modified Gram-Schmidt (orthogonalise 'one vector at a time'): $\|\hat{Q}^T\hat{Q} - I\| \leq \epsilon\kappa_2(A)$
  - Gram-Schmidt twice (G-S again on computed $\hat{Q}$): $\|\hat{Q}^T\hat{Q} - I\| \leq \epsilon$

Matrix multiplication is not backward stable

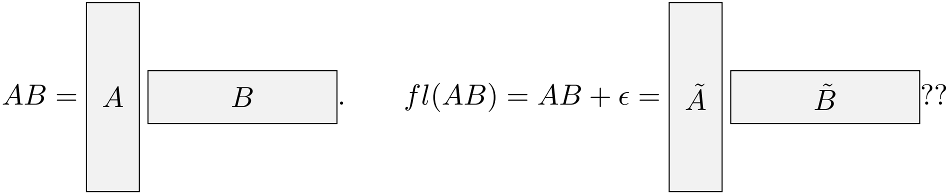
Shock! It is not always true that $fl(AB)$ equal to $(A+\Delta A)(B+\Delta B)$ for small $\Delta A, \Delta B$

- Vec-vec mult. backward stable: $fl(y^T x) = (y+\Delta y)(x+\Delta x)$; in fact $fl(y^T x) = (y+\Delta y)x$.
- Hence mat-vec also backward stable: $fl(Ax) = (A+\Delta A)x$.
- Still mat-mat is not backward stable.

Matrix multiplication is not backward stable

Shock! It is not always true that $fl(AB)$ equal to $(A + \Delta A)(B + \Delta B)$ for small $\Delta A, \Delta B$

- Vec-vec mult. backward stable: $fl(y^T x) = (y + \Delta y)(x + \Delta x)$; in fact $fl(y^T x) = (y + \Delta y)x$.
- Hence mat-vec also backward stable: $fl(Ax) = (A + \Delta A)x$.
- Still mat-mat is not backward stable.



with $\tilde{A} = A + \epsilon\|A\|$, $\tilde{B} = B + \epsilon\|B\|$? No—e.g., $fl(AB)$ is usually not low rank

Matrix multiplication is not backward stable

Shock! It is not always true that $fl(AB)$ equal to $(A+\Delta A)(B+\Delta B)$ for small $\Delta A, \Delta B$

- Vec-vec mult. backward stable: $fl(y^T x) = (y+\Delta y)(x+\Delta x)$; in fact $fl(y^T x) = (y+\Delta y)x$.
- Hence mat-vec also backward stable: $fl(Ax) = (A+\Delta A)x$.
- Still mat-mat is not backward stable.

What **is** true: $\|fl(AB) - AB\| \leq \epsilon \|A\| \|B\|$, so $\|fl(AB) - AB\| / \|AB\| \leq \epsilon \min(\kappa_2(A), \kappa_2(B))$.

- Great when A or B orthogonal (or square well-conditioned): say if $A = Q$ orthogonal,

$$\|fl(QB) - QB\| \leq \epsilon \|B\|,$$

so $fl(QB) = QB + \epsilon \|B\|$, hence $fl(QB) = Q(B + \Delta B)$ where $\Delta B = Q^T \epsilon \|B\|$

orthogonal multiplication is backward stable

Stability of Householder QR

With Householder QR, the computed $\hat{Q}, \hat{R}$ satisfy

$$\|\hat{Q}^T\hat{Q} - I\| = O(\epsilon), \quad \|A - \hat{Q}\hat{R}\| = O(\epsilon\|A\|),$$

and (of course) R upper triangular.

Rough proof

- Each reflector orthogonal, so satisfies $fl(H_iA) = H_iA + \epsilon_i\|A\|$
- Hence $(\hat{R} =) fl(H_n \cdots H_1 A) = H_n \cdots H_1 A + \epsilon\|A\|$
- $fl(H_n \cdots H_1) =: \hat{Q}^T = H_n \cdots H_1 + \epsilon$,
- Thus $\hat{Q}\hat{R} = A + \epsilon\|A\|$

Stability of Householder QR

With Householder QR, the computed $\hat{Q}, \hat{R}$ satisfy

$$\|\hat{Q}^T\hat{Q} - I\| = O(\epsilon), \quad \|A - \hat{Q}\hat{R}\| = O(\epsilon\|A\|),$$

and (of course) R upper triangular.

Rough proof

- Each reflector orthogonal, so satisfies $fl(H_iA) = H_iA + \epsilon_i\|A\|$
- Hence $(\hat{R} =) fl(H_n \cdots H_1 A) = H_n \cdots H_1 A + \epsilon\|A\|$
- $fl(H_n \cdots H_1) =: \hat{Q}^T = H_n \cdots H_1 + \epsilon$,
- Thus $\hat{Q}\hat{R} = A + \epsilon\|A\|$

Notes:

- This doesn't mean $\|\hat{Q} - Q\|, \|\hat{R} - R\|$ are small at all! Indeed Q, R are as ill-conditioned as A
- QR for $Ax = b$, least-squares are stable (NB normal eqn $A^TAx =$ is NOT)

Orthogonal Linear Algebra

With orthogonal matrices Q,

$$\frac{\|fl(QA) - QA\|}{\|QA\|} \leq \epsilon, \qquad \frac{\|fl(AQ) - AQ\|}{\|AQ\|} \leq \epsilon$$

Orthogonal Linear Algebra

With orthogonal matrices Q,

$$\frac{\|fl(QA) - QA\|}{\|QA\|} \leq \epsilon, \qquad \frac{\|fl(AQ) - AQ\|}{\|AQ\|} \leq \epsilon$$

whereas in general, $\|fl(AB) - AB\| \leq \epsilon \|A\| \|B\|$, so
$\|fl(AB) - AB\| / \|AB\| \leq \epsilon \min(\kappa_2(A), \kappa_2(B))$

Orthogonal Linear Algebra

With orthogonal matrices Q,

$$\frac{\|fl(QA) - QA\|}{\|QA\|} \le \epsilon, \qquad \frac{\|fl(AQ) - AQ\|}{\|AQ\|} \le \epsilon$$

whereas in general, $\|fl(AB) - AB\| \le \epsilon \|A\| \|B\|$, so
$\|fl(AB) - AB\| / \|AB\| \le \epsilon \min(\kappa_2(A), \kappa_2(B))$

Hence algorithms involving ill-conditioned matrices are unstable (e.g. eigenvalue decomposition of non-normal matrices, Jordan form, etc), whereas those based on orthogonal matrices are stable, e.g.

- Householder QR factorisation
- **QR algorithm** for $Ax = \lambda x$
- **Golub-Kahan** algorithm for $A = U\Sigma V^T$
- **QZ algorithm** for $Ax = \lambda Bx$

We next turn to the algorithms in boldface

Key points on stability

- Definition: (backward) stability vs. conditioning
- Orthogonal linear algebra is backward stable

- Significance of $\kappa_2(A) = \|A\|_2 \|A^{-1}\|$
- Stable operations: triangular systems, Cholesky,...

Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p

Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p
- Let $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, $a_i \in \mathbb{C}$. Then $p(\lambda) = 0 \Leftrightarrow \lambda$ eigenvalue of

$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{C}^{n\times n}$$

Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p
- So no finite-step algorithm exists for $Ax = \lambda x$

Eigenvalue algorithms are necessarily iterative and approximate

- Same for SVD, as $\sigma_i(A) = \sqrt{\lambda_i(A^T A)}$
- But this doesn't mean they're inaccurate!

Usual goal: compute the Schur decomposition $A = UTU^*$: U unitary, T upper triangular

- For normal matrices $A^*A = AA^*$, automatically diagonalised (T diagonal)
- For nonnormal A, if diagonalisation $A = X\Lambda X^{-1}$ really necessary, done via Sylvester equations but nonorthogonal/unstable (nonexaminable)

Schur decomposition

Let $A \in \mathbb{C}^{n\times n}$ (square arbitrary matrix). Then $\exists$ unitary $U \in \mathbb{C}^{n\times n}$ s.t.

$$A = UTU^*,$$

with T upper triangular.

- $\text{eig}(A) = \text{eig}(T) = \text{diag}(T)$
- T diagonal iff A normal $A^*A = AA^*$

Proof:

Schur decomposition

Let $A \in \mathbb{C}^{n\times n}$ (square arbitrary matrix). Then $\exists$ unitary $U \in \mathbb{C}^{n\times n}$ s.t.

$$A = UTU^*,$$

with T upper triangular.

- eig$(A) =$ eig$(T) =$ diag(T)
- T diagonal iff A normal $A^*A = AA^*$

Proof: Let $Av = \lambda_1 v$ and find $U_1 = [v_1, V_\perp]$ unitary. Then

$$AU_1 = U_1 \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix} \Leftrightarrow U_1^* A U_1 = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}.$$

Repeat on the lower-right $(n-1)\times(n-1)$ part to get $U_{n-1}^* U_{n-2}^* \dots U_1^* A U_1 U_2 \dots U_{n-1} = T$.

Recap: Matrix decompositions

- SVD $A = U\Sigma V^T$
- Eigenvalue decomposition $A = X\Lambda X^{-1}$
  - Normal: X unitary $X^*X = I$
  - Symmetric: X unitary and Λ real
- Jordan decomposition: $A = XJX^{-1}$, $J = \text{diag}\left(\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}\right)$
- **Schur decomposition** $A = QTQ^*$: Q orthogonal, T upper triangular
- QR: Q orthonormal, U upper triangular
- LU: L lower triangular, U upper triangular

Red: Orthogonal decompositions, stable computation available

Recap: Matrix decompositions

- SVD $A = U\Sigma V^T$
- Eigenvalue decomposition $A = X\Lambda X^{-1}$
  - Normal: X unitary $X^*X = I$
  - Symmetric: X unitary and Λ real
- Jordan decomposition: $A = XJX^{-1}$, $J = \text{diag}(\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix})$
- **Schur decomposition** $A = QTQ^*$: Q orthogonal, T upper triangular
- QR: Q orthonormal, U upper triangular
- LU: L lower triangular, U upper triangular
- QZ for $Ax = \lambda Bx$: (genearlised eigenvalue problem) Q, Z orthogonal s.t. QAZ, QBZ are both upper triangular

Red: Orthogonal decompositions, stable computation available

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

- Convergence analysis: suppose A is diagonalisable (generic assumption). We can write $x_0 = \sum_{i=1}^{n} c_i v_i$, $Av_i = \lambda_i v_i$ with $|\lambda_1| > |\lambda_2| > \cdots$. Then after k iterations,
$$x = C \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i v_i \to C c_1 v_1 \quad \text{as } k \to \infty$$
- Converges geometrically $(\lambda, x) \to (\lambda_1, v_1)$ with **linear rate** $\frac{|\lambda_2|}{|\lambda_1|}$
- What does this imply about $A^k = QR$ as $k \to \infty$? First vector of $Q \to v_1$

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

- Convergence analysis: suppose A is diagonalisable (generic assumption). We can write $x_0 = \sum_{i=1}^n c_i v_i$, $Av_i = \lambda_i v_i$ with $|\lambda_1| > |\lambda_2| > \cdots$. Then after k iterations,

$$x = C \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i v_i \to C c_1 v_1 \quad \text{as } k \to \infty$$

- Converges geometrically $(\lambda, x) \to (\lambda_1, v_1)$ with **linear rate** $\frac{|\lambda_2|}{|\lambda_1|}$
- What does this imply about $A^k = QR$ as $k \to \infty$? First vector of $Q \to v_1$

Notes:

- Google pagerank & Markov chain linked to power method
- As we'll see, power method is basis for refined algs (QR algorithm, Krylov methods (Lanczos, Arnoldi,...))

Why compute eigenvalues? Google PageRank

'Importance' of websites via dominant eigenvector of column-stochastic matrix

$$A = \alpha P + (1-\alpha) \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

P: adjacency matrix, $\alpha \in (0,1)$

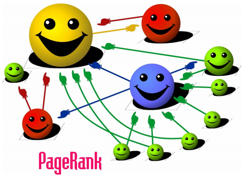


image from wikipedia

Google does (did) a few steps of Power method: with initial guess x_0, $k = 0, 1, \ldots$

1. $x_{k+1} = Ax_k$
2. $x_{k+1} = x_{k+1}/\|x_{k+1}\|_2, \quad k \leftarrow k+1$, repeat.

- $x_k \to$ PageRank vector $v_1 : Av_1 = \lambda_1 v_1$

Inverse power method

Inverse (shift-and-invert) power method: $x := (A - \mu I)^{-1}x$, $x = x/\|x\|$

- Converges with improved **linear rate** $\frac{|\lambda_{\sigma(2)} - \mu|}{|\lambda_{\sigma(1)} - \mu|}$ to eigval closest to μ (σ: permutation)

Inverse power method

Inverse (shift-and-invert) power method: $x := (A - \mu I)^{-1} x$, $x = x / \|x\|$

- Converges with improved **linear rate** $\frac{|\lambda_{\sigma(2)} - \mu|}{|\lambda_{\sigma(1)} - \mu|}$ to eigval closest to μ (σ: permutation)
- μ can change adaptively with the iterations. The choice $\mu := x^T A x$ gives Rayleigh quotient iteration, with **quadratic** convergence $\|Ax^{(k+1)} - \lambda^{(k+1)} x^{(k+1)}\| = O(\|Ax^{(k)} - \lambda^{(k)} x^{(k)}\|^2)$ (cubic if A symmetric)

Solving an eigenvalue problem

Given $A \in \mathbb{R}^{n\times n}$ or $\mathbb{C}^{n\times n}$,

$$Ax = \lambda x$$

Goal: find *all* eigenvalues (and eigenvectors) of a matrix

- Look for Schur form $A = UTU^*$

We'll describe an algorithm called the QR algorithm that is used universally, e.g. by MATLAB's `eig`. It

- finds all eigenvalues (approximately but reliably) in $O(n^3)$ flops,
- is backward stable.

Sister problem: Given $A \in \mathbb{R}^{m\times n}$ or $\mathbb{C}^{m\times n}$, compute SVD $A = U\Sigma V^*$

- 'ok' algorithm: eig($A^T A$) to find V, then normalise AV
- there's a better algorithm: Golub-Kahan bidiagonalisation

QR algorithm for eigenproblems

Set $A_1 = A$, and

$$A_1 = Q_1 R_1, \quad A_2 = R_1 Q_1, \quad A_2 = Q_2 R_2, \quad A_3 = R_2 Q_2, \quad \ldots$$

- A_k are all similar: $A_{k+1} = Q_k^T A_k Q_k$
- We shall 'show' that $A \to$ **triangular** (diagonal if A normal)
- Basically: QR(factorise)$\to RQ$(swap)$\to QR \to RQ \to \cdots$

QR algorithm for eigenproblems

Set $A_1 = A$, and

$$A_1 = Q_1 R_1, \quad A_2 = R_1 Q_1, \quad A_2 = Q_2 R_2, \quad A_3 = R_2 Q_2, \quad \dots$$

- A_k are all similar: $A_{k+1} = Q_k^T A_k Q_k$
- We shall 'show' that $A \to$ **triangular** (diagonal if A normal)
- Basically: QR(factorise)$\to$ RQ(swap)$\to$ $QR \to RQ \to \cdots$

- Fundamental work by Francis (61,62) and Kublanovskaya (63)

- Truly Magical algorithm!
  - backward stable, as based on orthogonal transforms
  - always converges (with shifts), but global proof unavailable(!)
  - uses 'shifted inverse power method' (rational functions) without inversions

QR algorithm and power method

QR algorithm: $A_k = Q_k R_k$, $A_{k+1} = R_k Q_k$, repeat. Claims: for $k \geq 1$,

$$A^k = (Q_1 \cdots Q_k)(R_k \cdots R_1) =: Q^{(k)} R^{(k)}, \qquad A_{k+1} = (Q^{(k)})^T A Q^{(k)}.$$

Proof: recall $A_{k+1} = Q_k^T A_k Q_k$, repeat.

Proof by induction: $k = 1$ trivial.
Suppose $A^{k-1} = Q^{(k-1)} R^{(k-1)}$. We have

$$A_k = (Q^{(k-1)})^T A Q^{(k-1)} = Q_k R_k.$$

Then $AQ^{(k-1)} = Q^{(k-1)} Q_k R_k$, and so

$$A^k = AQ^{(k-1)} R^{(k-1)} = Q^{(k-1)} Q_k R_k R^{(k-1)} = Q^{(k)} R^{(k)} \square$$

QR algorithm and power method

QR algorithm: $A_k = Q_k R_k$, $A_{k+1} = R_k Q_k$, repeat.

$$A^k = (Q_1 \cdots Q_k)(R_k \cdots R_1) =: Q^{(k)} R^{(k)}, \qquad A_{k+1} = (Q^{(k)})^T A Q^{(k)}.$$

QR factorisation of A^k: 'dominated by leading eigenvector' x_1,
where $Ax_1 = \lambda_1 x_1$ (recall power method)

In particular, consider $A^k[1, 0, \ldots, 0]^T = A^k e_n$:

- $A^k e_n = R^{(k)}(1,1) Q^{(k)}(:,1)$, parallel to 1st column of $Q^{(k)}$
- By power method, this implies $Q^{(k)}(:,1) \to x_1$
- Hence by $A_{k+1} = (Q^{(k)})^T A Q^{(k)}$, $A_k(:,1) \to [\lambda_1, 0, \ldots, 0]^T$

Progress! But there is much better news

QR algorithm and inverse power method

QR algorithm: $A_k = Q_k R_k$, $A_{k+1} = R_k Q_k$, repeat.

$$A^k = (Q_1 \cdots Q_k)(R_k \cdots R_1) =: Q^{(k)} R^{(k)}, \qquad A_{k+1} = (Q^{(k)})^T A Q^{(k)}.$$

Now take inverse: $A^{-k} = (R^{(k)})^{-1}(Q^{(k)})^T$,

transpose: $(A^{-k})^T = Q^{(k)}(R^{(k)})^{-T}$

$\Rightarrow$ QR factorization of matrix $(A^{-k})^T$ with eigvals $r(\lambda_i) = \lambda_i^{-k}$

$\Rightarrow$ Connection also with (unshifted) inverse power method

NB no matrix inverse performed

- This means final column of $Q^{(k)}$ converges to minimum left eigenvector x_n with factor $\frac{|\lambda_n|}{|\lambda_{n-1}|}$, hence $A_k(n,:) \to [0, \ldots, 0, \lambda_n]$
- (Very) fast convergence if $|\lambda_n| \ll |\lambda_{n-1}|$
- Can we force this situation? **Yes by shifts**

QR algorithm with shifts and shifted inverse power method

1. $A_k - s_k I = Q_k R_k$ (QR factorization)
2. $A_{k+1} = R_k Q_k + s_k I$, $\quad k \leftarrow k+1$, repeat.

Roughly, if $s_k \approx \lambda_n$, then $A_{k+1} \approx \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ & & & & \lambda_n \end{bmatrix}$ by argument just made.

QR algorithm with shifts and shifted inverse power method

1. $A_k - s_k I = Q_k R_k$ (QR factorization)
2. $A_{k+1} = R_k Q_k + s_k I$, $\quad k \leftarrow k+1$, repeat.

$$\prod_{i=1}^{k}(A - s_i I) = Q^{(k)} R^{(k)} \; (= (Q_1 \cdots Q_k)(R_k \cdots R_1))$$

Proof: Suppose true for $k-1$. Then QR alg. computes $(Q^{(k-1)})^T (A - s_k I) Q^{(k-1)} = Q_k R_k$, so $(A - s_k I) Q^{(k-1)} = Q^{(k-1)} Q_k R_k$, hence

$$\prod_{i=1}^{k}(A - s_i I) = (A - s_k I) Q^{(k-1)} R^{(k-1)} = Q^{(k-1)} Q_k R_k R^{(k-1)} = Q^{(k)} R^{(k)}.$$

Inverse transpose: $\prod_{i=1}^{k}(A - s_i I)^{-T} = Q^{(k)} (R^{(k)})^{-T}$

- QR factorization of matrix with eigvals $r(\lambda_j) = \prod_{i=1}^{k} \frac{1}{\lambda_j - s_i}$
- Converges like ratio of $\prod_{i=1}^{k}(\bar{\lambda}_j - s_i)$; very fast if $s_i \approx \lambda_j$. Ideally, choose $s_k \approx \lambda_n$
- Connection with shifted inverse power method, hence rational approximation

QR algorithm preprocessing

We've seen the QR iterations drives colored entries to 0 (esp. red ones)

$$A = \begin{bmatrix} * & * & * & * & * \\ \color{blue}{*} & * & * & * & * \\ \color{blue}{*} & * & * & * & * \\ \color{blue}{*} & * & * & * & * \\ \color{red}{*} & \color{red}{*} & \color{red}{*} & \color{red}{*} & * \end{bmatrix}$$

- Hence $A_{n,n} \to \lambda_n$, so choosing $s_k = A_{n,n}$ is sensible
- This reduces #QR iterations to $O(n)$ (empirical but reliable estimate)
- But each iteration is $O(n^3)$ for QR, overall $O(n^4)$
- We next discuss a preprocessing technique to reduce to $O(n^3)$

QR algorithm preprocessing: Hessenberg reduction

To improve cost of QR factorisation, first reduce via orthogonal Householder transformations

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}, \quad H_1 A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}, \quad H_1 = I - 2v_1 v_1^T,\ v_1 = \begin{bmatrix} {\color{red}0} \\ * \\ * \\ * \\ * \end{bmatrix}$$

Then $H_1 A H_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}$. Repeat with $H_2 = I - 2v_2 v_2^T, v_2 = [0, 0, *, *, *]^T$, ...:

$$H_2 H_1 A H_1 H_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}, \qquad H_3 H_2 H_1 A H_1 H_2 H_3 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix},$$

Hessenberg reduction continued

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \overset{H_1}{\rightarrow} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix} \overset{H_2}{\rightarrow} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix} \overset{H_3}{\rightarrow} \quad \cdots \quad \overset{H_{n-2}}{\rightarrow} \begin{bmatrix} * & * & * & * & * \\ {\color{red}*} & * & * & * & * \\ & {\color{red}*} & * & * & * \\ & & {\color{red}*} & * & * \\ & & & {\color{red}*} & * \end{bmatrix}.$$

- QR iterations preserve structure: if $A_1 = QR$ Hessenberg, then so is $A_2 = RQ$
- using Givens rotations, each QR iter is $O(n^2)$ (not $O(n^3)$)
- overall shifted QR algorithm cost is $O(n^3), \approx 25n^3$ flops

- Remaining task (done by shifted QR): drive subdiagonal ${\color{red}*}$ to 0
- bottom-right $* \rightarrow \lambda_n$, can be used for shift s_k

Deflation

Once bottom-right $|*| < \epsilon$,

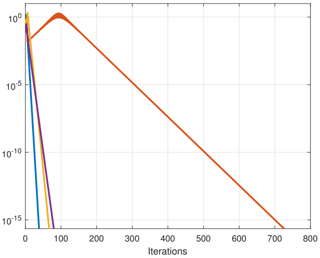
$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix} \approx \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & & * \end{bmatrix}$$

and continue with shifted QR on $(n-1) \times (n-1)$ block, repeat

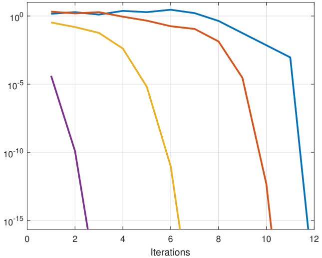
QR algorithm in action

Convergence of $|A_{i+1,i}|$

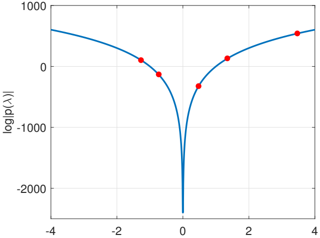
No shift (plain QR)

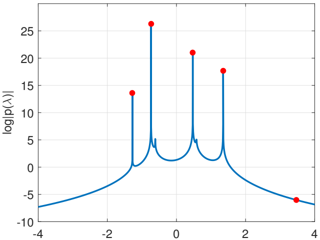


QR with shifts



underlying functions (red dots: eigvals)





QR algorithm: other improvements/simplifications (nonexaminable)

- Double-shift strategy for $A \in \mathbb{R}^{n\times n}$
  - $(A - sI)(A - \bar{s}I) = QR$ using only real arithmetic if A real
- Aggressive early deflation [Braman-Byers-Mathias 2002]
  - Examine lower-right (say 100×100) block instead of $(n, n-1)$ element
  - dramatic speedup ($\approx \times 10$)
- Balancing $A \leftarrow DAD^{-1}$, D: diagonal
  - reduce $\|DAD^{-1}\|$: better-conditioned eigenvalues

- For nonsymmetric A, global convergence is NOT established

  (except [Banks-Garza-Vargas-Srivastava 2021] for possible argument)
  - of course it always converges in practice.. another big open problem in numerical linear algebra

QR algorithm for symmetric A

- Initial reduction to Hessenberg form $\rightarrow$ tridiagonal

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \overset{Q_1}{\rightarrow} \begin{bmatrix} * & * & & & \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix} \overset{Q_2}{\rightarrow} \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix} \overset{Q_3}{\rightarrow} \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ & & & * & * \end{bmatrix}$$

- QR steps for tridiagonal: $O(n)$ instead of $O(n^2)$ per step
- Powerful alternatives available for tridiagonal eigenproblem (divide-conquer [Gu-Eisenstat 95], HODLR [Kressner-Susnjara 19],...)
- Cost: $\frac{4}{3}n^3$ flops for eigvals, $\approx 10n^3$ for eigvecs (store Givens rotations)

Golub-Kahan for SVD

Apply Householder reflectors from left and right (different ones) to **bidiagonalize**

$$A \to B = H_{L,n} \cdots H_{L,1} A H_{R,1} H_{R,2} \cdots H_{R,n-2}$$

$$A \overset{H_{L,1}}{\to} \begin{bmatrix} \star & \star & \star & \star \\ & \star & \star & \star \\ & \star & \star & \star \\ & \star & \star & \star \\ & \star & \star & \star \end{bmatrix} \overset{H_{R,1}}{\to} \begin{bmatrix} \star & \star & & \\ & \star & \star & \star \\ & \star & \star & \star \\ & \star & \star & \star \\ & \star & \star & \star \end{bmatrix} \overset{H_{L,2}}{\to} \begin{bmatrix} \star & \star & & \\ & \star & \star & \star \\ & & \star & \star \\ & & \star & \star \\ & & \star & \star \end{bmatrix} \overset{H_{R,2}}{\to} \begin{bmatrix} \star & \star & & \\ & \star & \star & \\ & & \star & \star \\ & & \star & \star \\ & & \star & \star \end{bmatrix} \overset{H_{L,3}}{\to} \begin{bmatrix} \star & \star & & \\ & \star & \star & \\ & & \star & \star \\ & & & \star \\ & & & \star \end{bmatrix} \overset{H_{L,4}}{\to} B,$$

- $\sigma_i(A) = \sigma_i(B)$
- Once bidiagonalized,
  - Mathematically, do QR alg on $B^T B$ (symmetric tridiagonal)
  - More elegant: divide-and-conquer [Gu-Eisenstat 1995] or dqds algorithm [Fernando-Parlett 1994]; nonexaminable
- Cost: $\approx 4mn^2$ flops for singvals Σ, $\approx 20mn^2$ flops for singvecs U, V

QZ algorithm for generalised eigenvalue problems

Generalised eigenvalue problem

$$Ax = \lambda Bx, \qquad A, B \in \mathbb{C}^{n\times n}$$

- A, B given, find eigenvalues λ and eigenvector x
- n eigenvalues, roots of $\det(A - \lambda B)$
- Important case: A, B symmetric, B positive definite: λ all real

QZ algorithm: look for unitary Q, Z s.t. QAZ, QBZ both upper triangular

- then $\mathrm{diag}(QAZ)/\mathrm{diag}(QBZ)$ are eigenvalues
- Algorithm: first reduce A, B to Hessenberg-triangular form
- then implicitly do QR to $B^{-1}A$ (without inverting B)
- Cost: $\approx 50n^3$
- See [Golub-Van Loan] for details

Tractable eigenvalue problems

- Standard eigenvalue problems $Ax = \lambda x$
  - symmetric ($4/3n^3$ flops for eigvals, $+9n^3$ for eigvecs)
  - nonsymmetric ($10n^3$ flops for eigvals, $+15n^3$ for eigvecs)
- SVD $A = U\Sigma V^T$ for $A \in \mathbb{C}^{m\times n}$: ($\frac{8}{3}mn^2$ flops for singvals, $+20mn^2$ for singvecs)
- Generalized eigenvalue problems $Ax = \lambda Bx$, $A, B \in \mathbb{C}^{n\times n}$
- Polynomial eigenvalue problems, e.g. (degree $k = 2$) $P(\lambda)x = (\lambda^2 A + \lambda B + C)x = 0$, $A, B, C \in \mathbb{C}^{n\times n}$:$\approx 20(nk)^3$
- Nonlinear problems, e.g. $N(\lambda)x = (A\exp(\lambda) + B)x = 0$
  - often solved via approximating by polynomial $N(\lambda) \approx P(\lambda)$
  - more difficult: $A(x)x = \lambda x$: eigenvector nonlinearity

Further speedup when structure present (e.g. sparse, low-rank)

Iterative methods

We've covered direct methods (LU for $Ax = b$, QR for $\min \|Ax - b\|_2$, QRalg for $Ax = \lambda x$). These are

- Incredibly reliable, backward stable
- Works like magic if $n \lesssim 10000$
- But not if n larger!

A 'big' matrix problem is one for which direct methods aren't feasible. Historically,

- 1950: $n \geq 20$
- 1965: $n \geq 200$
- 1980: $n \geq 2000$
- 1995: $n \geq 20000$
- 2010: $n \geq 100000$
- 2020: $n \geq 1000000$ ($n \geq 50000$ on a standard desktop)

was considered 'very large'. For such problems, we need to turn to alternative algorithms: we'll cover **iterative** and **randomised** methods.

Direct vs. iterative methods

Idea of iterative methods:

- gradually refine solution iteratively
- each iteration should be (a lot) cheaper than direct methods, usually $O(n^2)$ or less
- can be (but not always) much faster than direct methods
- tends to be (slightly) less robust, nontrivial/problem-dependent analysis
- often, after $O(n^3)$ work it still gets the exact solution (ignoring roundoff errors)

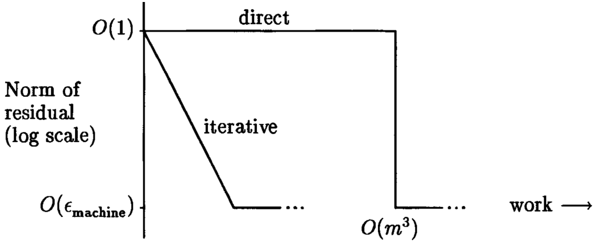


image from [Trefethen-Bau]

We'll focus on **Krylov subspace methods**.

Basic idea of Krylov: polynomial approximation

In Krylov subspace methods, we look for an (approximate) solution $\hat{x}$ (for $Ax = b$ or $Ax = \lambda x$) of the form (after kth iteration)

$$\hat{x} = p_{k-1}(A)v \,,$$

where p_{k-1} is a polynomial of degree $k-1$, and $v \in \mathbb{R}^n$ arbitrary (usually $v = b$ for linsys, for eigenproblems v usually random)

Natural questions:

- Why would this be a good idea?
  - Clearly, 'easy' to compute
  - One example: recall power method $\hat{x} = A^{k-1}v = p_{k-1}(A)v$
    Krylov finds a "better/optimal" polynomial $p_{k-1}(A)$
  - We'll see more cases where Krylov is powerful
- How to turn into an algorithm?
  - Arnoldi (next), Lanczos

Orthonormal basis for $\mathcal{K}_k(A, b)$

Find approximate solution $\hat{x} = p_{k-1}(A)b$, i.e. in Krylov subspace

$$\mathcal{K}_k(A, b) := \text{span}([b, Ab, A^2b, \ldots, A^{k-1}b])$$

First step: form an orthonormal basis Q, s.t. solution can be written as $x = Qy$

- Naive idea: Form matrix $[b, Ab, A^2b, \ldots, A^{k-1}b]$, then QR
  - $[b, Ab, A^2b, \ldots, A^{k-1}b]$ is usually terribly conditioned! Dominated by leading eigvec
  - Q is therefore extremely ill-conditioned, inaccurately computed

Orthonormal basis for $\mathcal{K}_k(A, b)$

Find approximate solution $\hat{x} = p_{k-1}(A)b$, i.e. in Krylov subspace

$$\mathcal{K}_k(A, b) := \text{span}([b, Ab, A^2b, \ldots, A^{k-1}b])$$

First step: form an orthonormal basis Q, s.t. solution can be written as $x = Qy$

- Naive idea: Form matrix $[b, Ab, A^2b, \ldots, A^{k-1}b]$, then QR
  - $[b, Ab, A^2b, \ldots, A^{k-1}b]$ is usually terribly conditioned! Dominated by leading eigvec
  - Q is therefore extremely ill-conditioned, inaccurately computed
- Much better solution: Arnoldi process
  - Multiply A once at a time to the latest orthonormal vector q_i
  - Then orthogonalise Aq_i against previous q_j's $(j = 1, \ldots, i-1)$ (as in Gram-Schmidt)
  - Even better news: **Arnoldi decomposition** makes subsequent computation very convenient

Arnoldi iteration and Arnoldi decomposition

Set $q_1 = b/\|b\|_2$
For $k = 1, 2, \ldots,$
set $v = Aq_k$
for $j = 1, 2, \ldots, k$
$h_{jk} = q_j^T v$, $v = v - h_{jk} q_j$ % orthogonalise against q_j via modified G-S
end for
$h_{k+1,k} = \|v\|_2$, $q_{k+1} = v/h_{k+1,k}$
End for

Theorem

Suppose that $h_{k+1,k} \neq 0$ for $k = 1, \ldots, \ell$. Then for $k = 1, \ldots, \ell$,

$$\mathit{Span}(q_1, \ldots, q_k) = \mathcal{K}_k(A, b).$$

Proof: Induction on ℓ. Suppose true for $\ell = \hat{\ell}$ with $q_{\hat{\ell}} = p_{\ell-1}(A)b$. Then $q_{\hat{\ell}+1} = \frac{1}{h_{\hat{\ell}+1,\hat{\ell}}}(Aq_{\hat{\ell}} - \sum_{j=1}^{\hat{\ell}} h_{j,\hat{\ell}} q_j)$, which is of exact degree $\hat{\ell}$.

Arnoldi iteration and Arnoldi decomposition

Set $q_1 = b/\|b\|_2$
For $k = 1, 2, \ldots,$
  set $v = Aq_k$
    for $j = 1, 2, \ldots, k$
      $h_{jk} = q_j^T v$, $v = v - h_{jk} q_j$ % orthogonalise against q_j via modified G-S
    end for
      $h_{k+1,k} = \|v\|_2$, $q_{k+1} = v/h_{k+1,k}$
End for

- After k steps, $AQ_k = Q_{k+1}\tilde{H}_k = Q_k H_k + q_{k+1}[0, \ldots, 0, h_{k+1,k}]$, with $Q_k = [q_1, q_2, \ldots, q_k], Q_{k+1} = [Q_k, q_{k+1}]$, span$(Q_k) =$ span$([b, Ab, \ldots, A^{k-1}b])$

$$A \; Q_k = Q_{k+1} \; \tilde{H}_k, \quad \tilde{H}_k = \underbrace{\begin{bmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,k} \\ h_{2,1} & h_{2,2} & \ldots & h_{2,k} \\ & \ddots & & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix}}_{\mathbb{R}^{(k+1)\times k} \text{ upper Hessenberg}}, \quad Q_{k+1}^T Q_{k+1} = I_{k+1}$$

- Cost k A-multiplications$+O(k^2)$ inner products ($O(nk^2)$)

GMRES for $Ax = b$

Idea (very simple!): minimise residual in Krylov subspace: [Saad-Schulz 86]

$$x_k = \text{argmin}_{x \in \mathcal{K}_k(A,b)} \|Ax - b\|_2$$

GMRES for $Ax = b$

Idea (very simple!): minimise residual in Krylov subspace: [Saad-Schulz 86]

$$x_k = \text{argmin}_{x \in \mathcal{K}_k(A,b)} \|Ax - b\|_2$$

Algorithm: Given $AQ_k = Q_{k+1}\tilde{H}_k$ and writing $x_k = Q_k y$, rewrite as

$$\begin{aligned}
\min_y \|AQ_k y - b\|_2 &= \min_y \|Q_{k+1}\tilde{H}_k y - b\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{H}_k \\ 0 \end{bmatrix} y - \begin{bmatrix} Q_k^T \\ Q_{k,\perp}^T \end{bmatrix} b \right\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{H}_k \\ 0 \end{bmatrix} y - \|b\|_2 e_1 \right\|_2, \quad e_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^n
\end{aligned}$$

(where $[Q_k, Q_{k,\perp}]$ orthogonal; same trick as in least-squares)

- Minimised when $\|\tilde{H}_k y - \tilde{Q}_k^T b\| \to \min$; Hessenberg least-squares problem
- Solve via QR (k Givens rotations)+triangular solve, $O(k^2)$ in addition to Arnoldi

GMRES convergence: polynomial approximation

Recall that $x_k \in \mathcal{K}_k(A,b) \Rightarrow x_k = p_{k-1}(A)b$. Hence GMRES solution is

$$\begin{aligned}
\min_{x_k \in \mathcal{K}_k(A,b)} \|Ax_k - b\|_2 &= \min_{p_{k-1} \in \mathcal{P}_{k-1}} \|Ap_{k-1}(A)b - b\|_2 \\
&= \min_{\tilde{p} \in \mathcal{P}_k, \tilde{p}(0)=0} \|(\tilde{p}(A) - I)b\|_2 \\
&= \min_{p \in \mathcal{P}_k, p(0)=1} \|p(A)b\|_2
\end{aligned}$$

If A diagonalizable $A = X\Lambda X^{-1}$,

$$\begin{aligned}
\|p(A)\|_2 = \|Xp(\Lambda)X^{-1}\|_2 &\le \|X\|_2\|X^{-1}\|_2\|p(\Lambda)\|_2 \\
&= \kappa_2(X) \max_{z \in \lambda(A)} |p(z)|
\end{aligned}$$

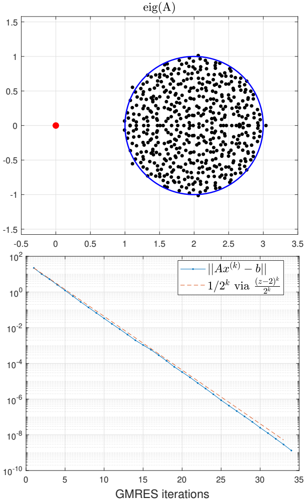
Interpretation: find polynomial s.t. $p(0) = 1$ and $|p(\lambda_i)|$ small for all i

GMRES example

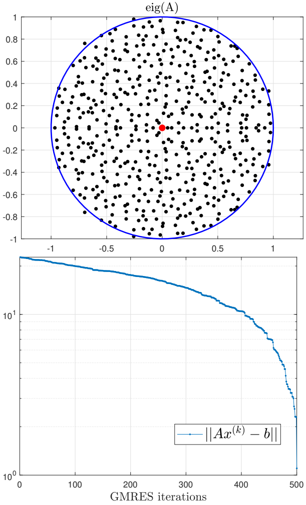
G: Gaussian random matrix ($G_{ij} \sim N(0,1)$, i.i.d.) $G/\sqrt{n}$: eigvals in unit disk

$A = 2I + G/\sqrt{n}$,

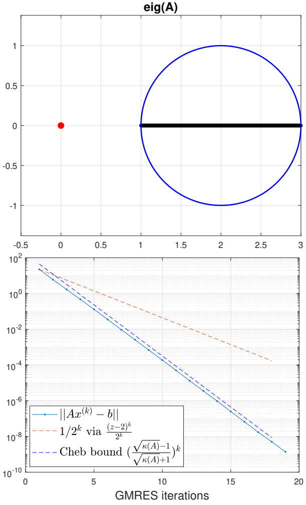
$p(z) = 2^{-k}(z-2)^k$



$A = G/\sqrt{n}$



eig$(A) \in [1, 3]$



When does GMRES converge fast?

Recall GMRES solution satisfies (assuming A diagonalisable+nonsingular)

$$\min_{x_k \in \mathcal{K}_k(A,b)} \|Ax_k - b\|_2 = \min_{p \in \mathcal{P}_k, p(0)=1} \|p(A)b\|_2 \leq \kappa_2(X) \max_{z \in \lambda(A)} |p(z)| \|b\|_2.$$

$\max_{z \in \lambda(A)} |p(z)|$ is small when

- $\lambda(A)$ are clustered away from 0
  - a good p can be found quite easily
  - e.g. example 2 slides ago

- When $\lambda(A)$ takes $k(\ll n)$ distinct values
  - Then convergence in k GMRES iterations (why?)

Preconditioning for GMRES

We've seen that GMRES is great if spectrum clustered away from 0. If not true with

$$Ax = b,$$

then precondition: find $M \in \mathbb{R}^{n \times n}$ and solve

$$MAx = Mb$$

Desiderata of M:

- M simple enough s.t. applying M to vector is easy (note that each GMRES iteration requires MA-multiplication), and one of
  1. MA has clustered eigenvalues away from 0
  2. MA has a small number of distinct eigenvalues
  3. MA is well-conditioned $\kappa_2(MA) = O(1)$; then solve normal equation $(MA)^T MAx = (MA)^T Mb$

Preconditioners: examples

- ILU (Incomplete LU) preconditioner: $A \approx LU, M = (LU)^{-1} = U^{-1}L^{-1}$, L, U 'as sparse as A' $\Rightarrow MA \approx I$ (hopefully; 'cluster away from 0')
- For $\tilde{A} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}$, set $M = \begin{bmatrix} A^{-1} & \\ & (CA^{-1}B)^{-1} \end{bmatrix}$. Then if M nonsingular, $M\tilde{A}$ has eigvals$\in \{1, \frac{1}{2}(1 \pm \sqrt{5})\} \Rightarrow$ 3-step convergence [Murphy-Golub-Wathen 2000]
- Multigrid-based, operator preconditioning, ...

Finding effective preconditioners is never-ending research topic

Prof. Andy Wathen is our Oxford expert!

Restarted GMRES

For k iterations, GMRES costs k matrix multiplications$+O(nk^2)$ for orthogonalization $\rightarrow$ Arnoldi eventually becomes expensive.

Practical solution: restart by solving 'iterative refinement':

1. Stop GMRES after $k_{\max}$ (prescribed) steps to get approx. solution $\hat{x}_1$
2. Solve $A\tilde{x} = b - A\hat{x}_1$ via GMRES
3. Obtain solution $\hat{x}_1 + \tilde{x}$

Sometimes multiple restarts needed

Lanczos iteration

Recall Arnoldi decomposition $AQ_k = Q_{k+1}\tilde{H}_k = Q_k H_k + q_{k+1}[0,\dots,0,h_{k+1,k}]$.
When A symmetric, Arnoldi decomposition simplifies to

$$AQ_k = Q_k T_k + q_{k+1}[0,\dots,0,t_{k+1,k}],$$

where T_k is symmetric tridiagonal (proof: just note $H_k = Q_k^T A Q_k$ in Arnoldi)

$$A\,Q_k = Q_{k+1}\,\tilde{T}_k, \quad \tilde{T}_k = \underbrace{\begin{bmatrix} t_{1,1} & t_{1,2} & & \\ t_{2,1} & t_{2,2} & \ddots & \\ & \ddots & & t_{k-1,k} \\ & & t_{k,k-1} & t_{k,k} \\ & & & t_{k+1,k} \end{bmatrix}}_{\mathbb{R}^{(k+1)\times k} \text{ symmetric tridiagonal}}, \quad Q_{k+1}^T Q_{k+1} = I_{k+1}$$

- 3-term recurrence $t_{k+1,k}q_{k+1} = (A - t_{k,k})q_k - t_{k-1,k}q_{k-1}$; orthogonalisation necessary only against last two vecs q_k, q_{k-1}
- Significant speedup over Arnoldi; cost k A-mult.$+O(k)$ inner products ($O(nk)$)

CG: Conjugate Gradient method for $Ax = b$, $A \succ 0$

When A symmetric, Lanczos gives $AQ_k = Q_k T_k + q_{k+1}[0, \ldots, 0, 1]$, T_k: tridiagonal

CG: when $A \succ 0$ PD, solve $Q_k^T(AQ_k y - b) = T_k y - Q_k^T b = 0$, and $x = Q_k y$
→"Galerkin orthogonality": residual $Ax - b$ orthogonal to Q_k

CG: Conjugate Gradient method for $Ax = b$, $A \succ 0$

When A symmetric, Lanczos gives $AQ_k = Q_kT_k + q_{k+1}[0, \ldots, 0, 1]$, T_k: tridiagonal

CG: when $A \succ 0$ PD, solve $Q_k^T(AQ_ky - b) = T_ky - Q_k^Tb = 0$, and $x = Q_ky$
→"Galerkin orthogonality": residual $Ax - b$ orthogonal to Q_k

- $T_ky = Q_k^Tb$ is tridiagonal linear system, $O(k)$ operations to solve
- three-term recurrence reduces cost to $O(k)$ A-multiplications
- minimises A-norm of error $x_k = \text{argmin}_{x \in Q_k} \|x - x_*\|_A$ $(Ax_* = b)$:

$$\begin{aligned}(x - x_*)^TA(x - x_*) &= (Q_ky - x_*)^TA(Q_ky - x_*)\\ &= y^T(Q_k^TAQ_k)y - 2b^TQ_ky + b^Tx_*,\end{aligned}$$

minimiser is $y = (Q_k^TAQ_k)^{-1}Q_k^Tb$, so $Q_k^T(AQ_ky - b) = 0$

- Note $\|x\|_A = \sqrt{x^TAx}$ defines a norm (exercise)
- More generally, for inner-product norm $\|z\|_M = \sqrt{< z, z >_M}$, $\min_{x=Qy} \|x_* - x\|_M$ attained when $< q_i, x_* - x >_M = 0$, $\forall q_i$ (cf. Part A NA)

CG algorithm for $Ax = b$, $A \succ 0$

Set $x_0 = 0$, $r_0 = -b$, $p_0 = r_0$ and do for $k = 1, 2, 3, \ldots$

$$
\begin{aligned}
\alpha_k &= \langle r_k, r_k \rangle / \langle p_k, Ap_k \rangle \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k A p_k \\
\beta_k &= \langle r_{k+1}, r_{k+1} \rangle / \langle r_k, r_k \rangle \\
p_{k+1} &= r_{k+1} + \beta_k p_k
\end{aligned}
$$

where $r_k = Ax_k - b$ (residual) and p_k (search direction).

One can show among others (exercise/sheet)

- $\mathcal{K}_k(A, b) = \text{span}(r_0, r_1, \ldots, r_{k-1}) = \text{span}(x_1, x_2, \ldots, x_k)$ (also equal to $\text{span}(p_0, p_1, \ldots, p_{k-1})$)
- $r_j^T r_k = 0$, $j = 0, 1, 2, \ldots, k-1$

Thus x_k is kth CG solution, satisfying orthogonality $Q_k^T (Ax_k - b) = 0$

CG convergence

Let $e_k := x_* - x_k$. We have $e_0 = x_*$ $(x_0 = 0)$, and

$$
\begin{aligned}
\frac{\|e_k\|_A}{\|e_0\|_A} &= \min_{x \in \mathcal{K}_k(A,b)} \|x_k - x_*\|_A / \|x_*\|_A \\
&= \min_{p_{k-1} \in \mathcal{P}_{k-1}} \|p_{k-1}(A)b - A^{-1}b\|_A / \|e_0\|_A \\
&= \min_{p_{k-1} \in \mathcal{P}_{k-1}} \|(p_{k-1}(A)A - I)e_0\|_A / \|e_0\|_A \\
&= \min_{p \in \mathcal{P}_k, p(0)=1} \|p(A)e_0\|_A / \|e_0\|_A \\
&= \min_{p \in \mathcal{P}_k, p(0)=1} \left\| V \begin{bmatrix} p(\lambda_1) & & \\ & \ddots & \\ & & p(\lambda_n) \end{bmatrix} V^T e_0 \right\|_A / \|e_0\|_A
\end{aligned}
$$

Now (blue)$^2 = \sum_i \lambda_i p(\lambda_i)^2 (V^T e_0)_i^2 \leq \max_j p(\lambda_j)^2 \sum_i \lambda_i (V^T e_0)_i^2 = \max_j p(\lambda_j)^2 \|e_0\|_A^2$

CG convergence cont'd

We've shown

$$\frac{\|e_k\|_A}{\|e_0\|_A} \le \min_{p\in\mathcal{P}_k, p(0)=1} \max_j |p(\lambda_j)| \le \min_{p\in\mathcal{P}_k, p(0)=1} \max_{x\in[\lambda_{\min}(A), \lambda_{\max}(A)]} |p(x)|$$

Now

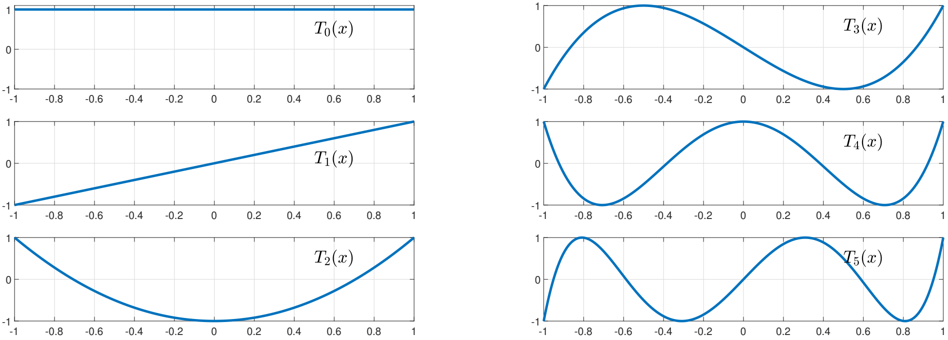
$$\min_{p\in\mathcal{P}_k, p(0)=1} \max_{x\in[\lambda_{\min}(A), \lambda_{\max}(A)]} |p(x)| \le 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k$$

- note $\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} (=: \frac{b}{a})$
- above bound obtained by Chebyshev polynomials on $[\lambda_{\min}(A), \lambda_{\max}(A)]$

Chebyshev polynomials

For $z = \exp(i\theta)$, $x = \frac{1}{2}(z + z^{-1}) = \cos\theta \in [-1, 1]$, $\theta = \text{acos}(x)$,
$T_k(x) = \frac{1}{2}(z^k + z^{-k}) = \cos(k\theta)$. $T_k(x)$ is a polynomial in x:

$$\frac{1}{2}(z+z^{-1})(z^k+z^{-k}) = \frac{1}{2}(z^{k+1}+z^{-(k+1)})+\frac{1}{2}(z^{k-1}+z^{-(k-1)}) \Leftrightarrow \underbrace{2xT_k(x) = T_{k+1}(x) + T_{k-1}(x)}_{\text{3-term recurrence; } 2\cos\theta\cos(k\theta)=\cos((k+1)\theta)+\cos((k-1)\theta)}$$

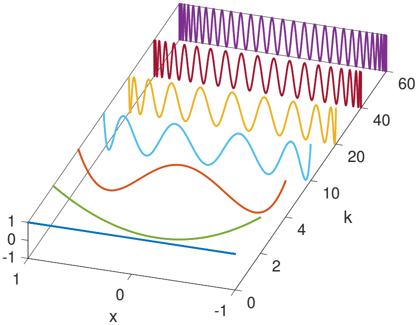


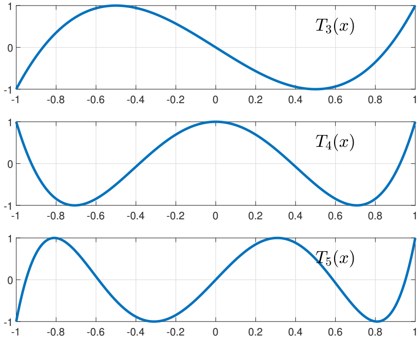
Chebyshev polynomials

For $z = \exp(i\theta)$, $x = \frac{1}{2}(z + z^{-1}) = \cos\theta \in [-1, 1]$, $\theta = \text{acos}(x)$,
$T_k(x) = \frac{1}{2}(z^k + z^{-k}) = \cos(k\theta)$. $T_k(x)$ is a polynomial in x:

$$\frac{1}{2}(z+z^{-1})(z^k+z^{-k}) = \frac{1}{2}(z^{k+1}+z^{-(k+1)})+\frac{1}{2}(z^{k-1}+z^{-(k-1)}) \Leftrightarrow \underbrace{2xT_k(x) = T_{k+1}(x) + T_{k-1}(x)}_{\text{3-term recurrence;}}$$

$2\cos\theta\cos(k\theta)=\cos((k+1)\theta)+\cos((k-1)\theta)$

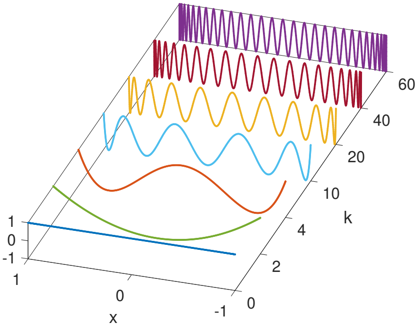


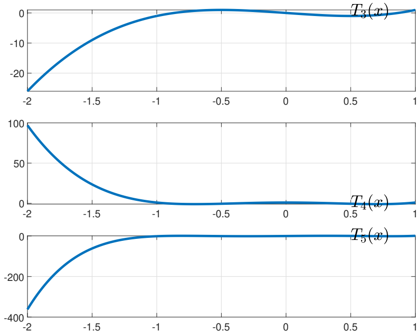


Chebyshev polynomials

For $z = \exp(i\theta)$, $x = \frac{1}{2}(z + z^{-1}) = \cos\theta \in [-1, 1]$, $\theta = \text{acos}(x)$,
$T_k(x) = \frac{1}{2}(z^k + z^{-k}) = \cos(k\theta)$. $T_k(x)$ is a polynomial in x:

$$\frac{1}{2}(z+z^{-1})(z^k+z^{-k}) = \frac{1}{2}(z^{k+1}+z^{-(k+1)})+\frac{1}{2}(z^{k-1}+z^{-(k-1)}) \Leftrightarrow \underbrace{2xT_k(x) = T_{k+1}(x) + T_{k-1}(x)}_{\substack{\text{3-term recurrence;} \\ 2\cos\theta\cos(k\theta)=\cos((k+1)\theta)+\cos((k-1)\theta)}}$$





Chebyshev polynomials cont'd

For $z = \exp(i\theta)$, $x = \frac{1}{2}(z + z^{-1}) = \cos\theta \in [-1, 1]$, $\theta = \mathrm{acos}(x)$,
$T_k(x) = \frac{1}{2}(z^k + z^{-k}) = \cos(k\theta)$.

- Inside $[-1, 1]$, $|T_k(x)| \le 1$
- Outside $[-1, 1]$, $|T_k(x)| \gg 1$ grows rapidly with $|x|, k$(fastest growth among $\mathcal{P}_k$)

Shift+scale s.t. $p(x) = c_k T_k(\frac{2x-b-a}{b-a})$ where $c_k = 1/T_k(\frac{-(b+a)}{b-a})$ so $p(0) = 1$. Then

- $|p(x)| \le 1/|T_k(\frac{-(b+a)}{b-a})| = 1/|T_k(\frac{b+a}{b-a})|$ on $x \in [a, b]$
- $T_k(z) = \frac{1}{2}(z^k + z^{-k})$ with $\frac{1}{2}(z + z^{-1}) = \frac{b+a}{b-a} \Rightarrow z = \frac{\sqrt{b/a}+1}{\sqrt{b/a}-1} = \frac{\sqrt{\kappa_2(A)}+1}{\sqrt{\kappa_2(A)}-1}$, so

$$|p(x)| \le 1/T_k(\frac{b+a}{b-a}) \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$$

For much more about T_k, see C6.3 Approximation of Functions

MINRES: symmetric (indefinite) version of GMRES (nonexaminable)

Recall GMRES

$$x = \text{argmin}_{x \in \mathcal{K}_k(A,b)} \|Ax - b\|_2$$

Algorithm: Given $AQ_k = Q_{k+1}\tilde{H}_k$ and writing $x = Q_k y$, rewrite as

$$\begin{aligned}
\min_y \|AQ_k y - b\|_2 &= \min_y \|Q_{k+1}\tilde{H}_k y - b\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{H}_k \\ 0 \end{bmatrix} y - \begin{bmatrix} Q_k^T \\ Q_{k,\perp}^T \end{bmatrix} b \right\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{H}_k \\ 0 \end{bmatrix} y - \|b\|_2 e_1 \right\|_2, \quad e_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^n
\end{aligned}$$

(where $[Q_k, Q_{k,\perp}]$ orthogonal; same trick as in least-squares)

- Minimised when $\|\tilde{T}_k y - \tilde{Q}_k^T b\| \to \min$; Hessenberg least-squares problem
- Solve via QR (k Givens rotations)+triangular solve, $O(k^2)$ in addition to Arnoldi

MINRES: symmetric (indefinite) version of GMRES (nonexaminable)

MINRES (minimum-residual method) for $A = A^T$ (but not necessarily $A \succ 0$)

$$x = \text{argmin}_{x \in \mathcal{K}_k(A,b)} \|Ax - b\|_2$$

Algorithm: Given $AQ_k = Q_{k+1}\tilde{T}_k$ and writing $x = Q_k y$, rewrite as

$$\begin{aligned}
\min_y \|AQ_k y - b\|_2 &= \min_y \|Q_{k+1}\tilde{T}_k y - b\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{T}_k \\ 0 \end{bmatrix} y - \begin{bmatrix} Q_k^T \\ Q_{k,\perp}^T \end{bmatrix} b \right\|_2 \\
&= \min_y \left\| \begin{bmatrix} \tilde{T}_k \\ 0 \end{bmatrix} y - \|b\|_2 e_1 \right\|_2, \quad e_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^n
\end{aligned}$$

(where $[Q_k, Q_{k,\perp}]$ orthogonal; same trick as in least-squares)

- Minimised when $\|\tilde{T}_k y - \tilde{Q}_k^T b\| \to \min$; tridiagonal least-squares problem
- Solve via QR (k Givens rotations)+tridiagonal solve, $O(k)$ in addition to Lanczos

MINRES convergence (nonexaminable)

As in GMRES,

$$\min_{x\in\mathcal{K}_k(A,b)}\|Ax-b\|_2=\min_{p_{k-1}\in\mathcal{P}_{k-1}}\|Ap_{k-1}(A)b-b\|_2=\min_{\tilde{p}\in\mathcal{P}_k,\tilde{p}(0)=0}\|(\tilde{p}(A)-I)b\|_2$$
$$=\min_{p\in\mathcal{P}_k,p(0)=1}\|p(A)b\|_2$$

Since $A=A^T$, A is diagonalisable $A=Q\Lambda Q^T$ with Q orthogonal, so

$$\|p(A)\|_2=\|Qp(\Lambda)Q^T\|_2\le\|Q\|_2\|Q^T\|_2\|p(\Lambda)\|_2$$
$$=\max_{z\in\lambda(A)}|p(z)|$$

Interpretation: (again) find polynomial s.t. $p(0)=1$ and $|p(\lambda_i)|$ small

MINRES convergence cont'd (nonexaminable)

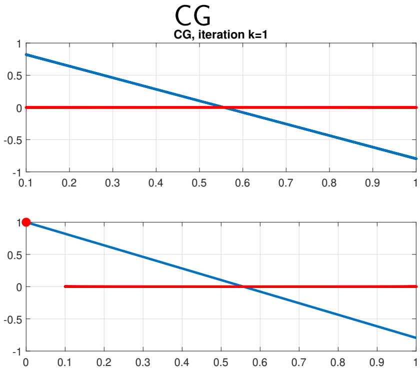
$$\frac{\|Ax-b\|_2}{\|b\|_2} \le \min_{p\in\mathcal{P}_k, p(0)=1} \max |p(\lambda_i)|$$

One can prove (nonexaminable)

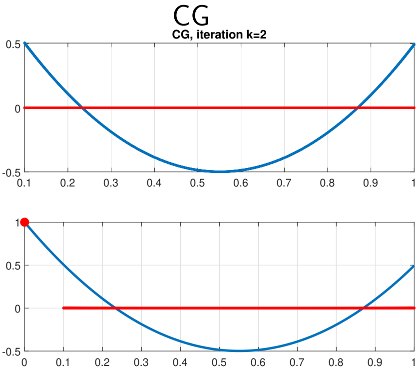
$$\min_{p\in\mathcal{P}_k, p(0)=1} \max |p(\lambda_i)| \le 2\left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^{k/2}$$

- obtained by Chebyshev+Möbius change of variables [Greenbaum's book 97]
- minimisation needed on positive **and** negative sides, hence slower convergence when A indefinite

CG and MINRES, optimal polynomials

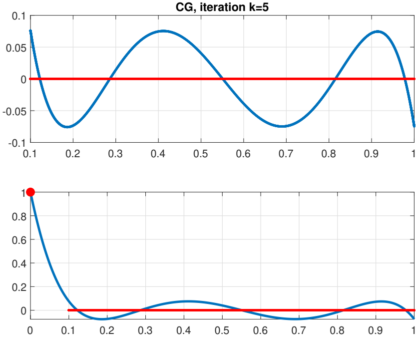


CG and MINRES, optimal polynomials

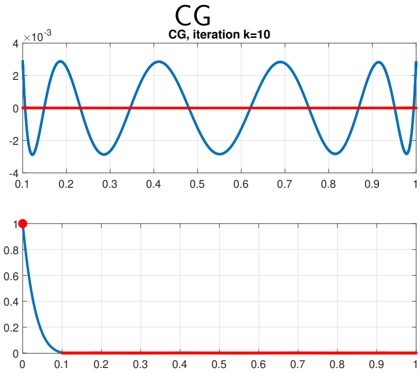


CG and MINRES, optimal polynomials

CG

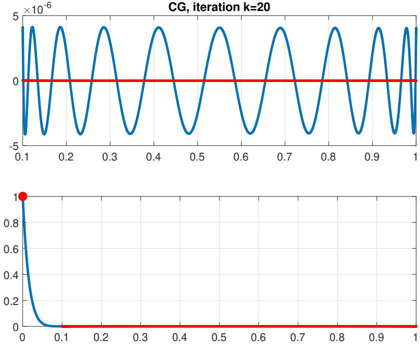


CG and MINRES, optimal polynomials

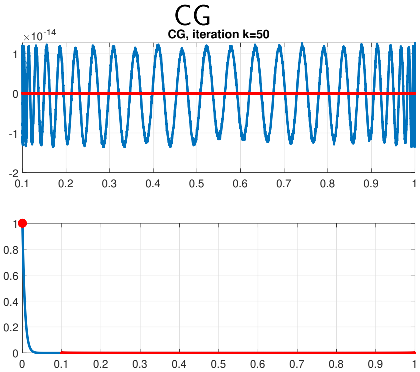


CG and MINRES, optimal polynomials

CG

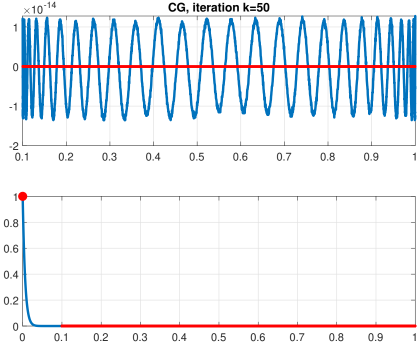


CG and MINRES, optimal polynomials

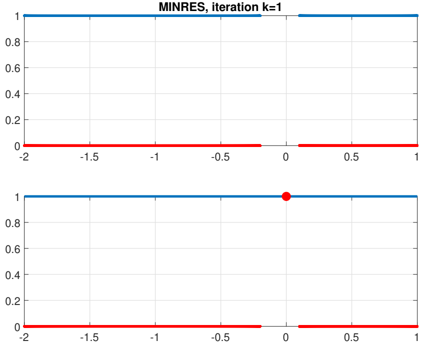


CG and MINRES, optimal polynomials

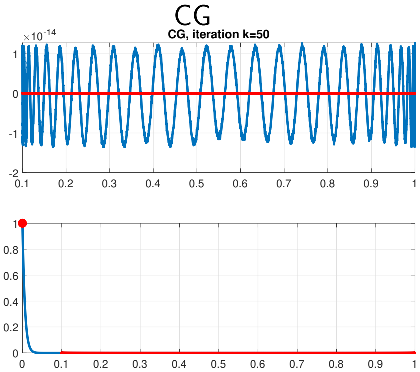
CG

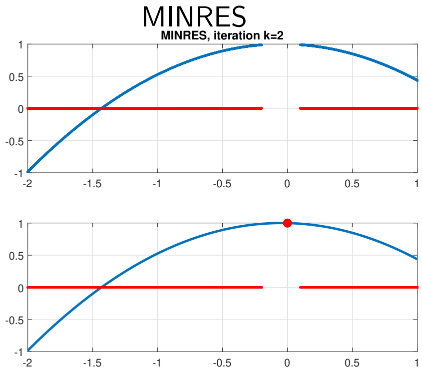


MINRES



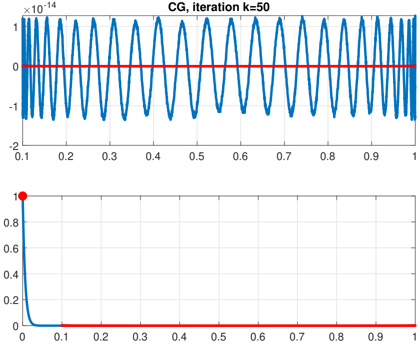
CG and MINRES, optimal polynomials



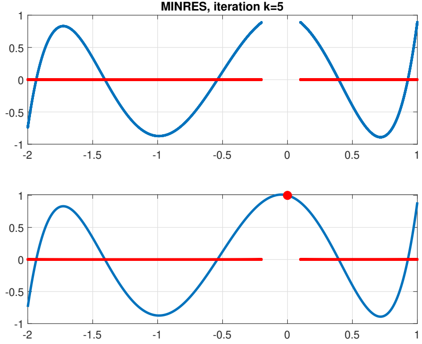


CG and MINRES, optimal polynomials

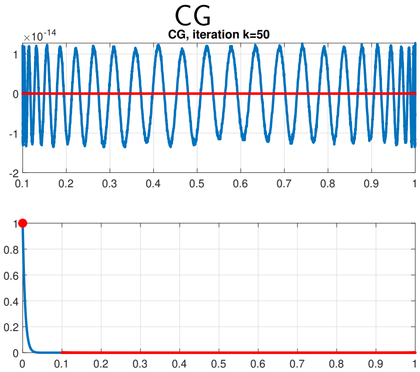
CG

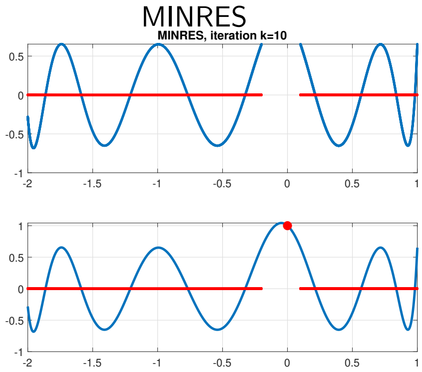


MINRES

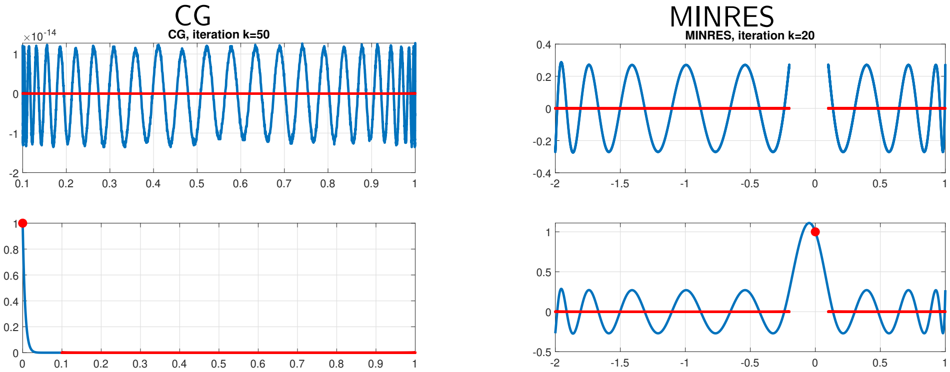


CG and MINRES, optimal polynomials

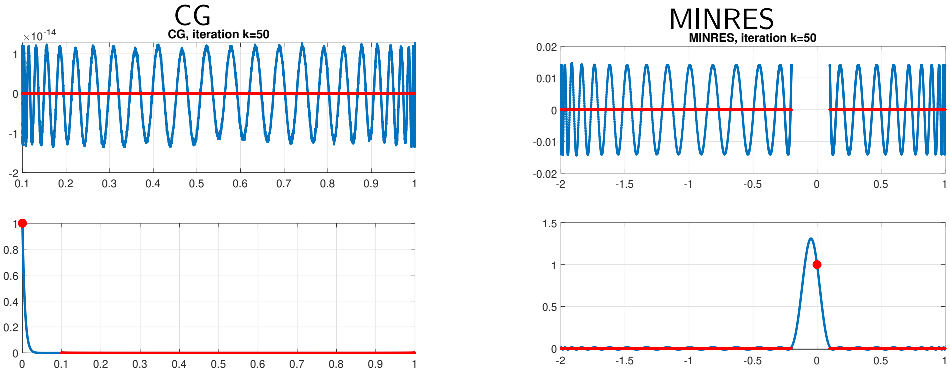




CG and MINRES, optimal polynomials



CG and MINRES, optimal polynomials



- CG employs Chebyshev polynomials
- MINRES is more complicated+slower convergence

Preconditioned CG/MINRES

$$Ax = b, \quad A \succ 0$$

Find preconditioner M s.t. "$M^T M \approx A^{-1}$" and solve

$$M^T A M y = M^T b, \quad My = x$$

As before, desiderata of M:

- $M^T A M$ simple to apply
- $M^T A M$ has clustered eigenvalues

Note that reducing $\kappa_2(M^T A M)$ directly implies rapid convergence

- Possible to implement with just $M^T M$ (no need to find M)

The Lanczos algorithm for symmetric eigenproblem (nonexaminable)

Rayleigh-Ritz: given symmetric A and orthonormal Q, find approximate eigenpairs

1. Compute $Q^T AQ$
2. Eigenvalue decomposition $Q^T AQ = V\hat{\Lambda}V^T$
3. Approximate eigenvalues diag$(\hat{\Lambda})$ (Ritz values) and eigenvectors QV (Ritz vectors)

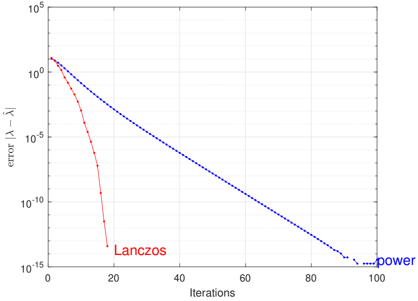
This is a **projection** method (similar alg. available for SVD)

Lanczos algorithm=Lanczos iteration+Rayleigh-Ritz

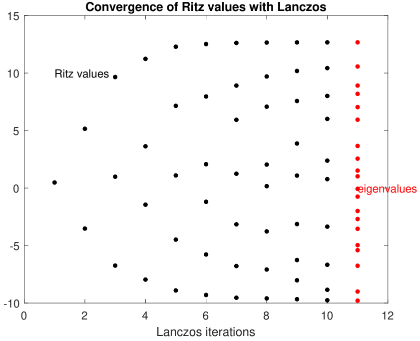
- In this case $Q = Q_k$, so simply $Q_k^T AQ_k = T_k$ (tridiagonal eigenproblem)
- Very good convergence to extremal eigenpairs
  - Recall from Courant-Fisher $\lambda_{\max}(A) = \max_x \frac{x^T Ax}{x^T x}$
  - Hence $\lambda_{\max}(A) \geq \underbrace{\max_{x\in\mathcal{K}_k(A,b)} \frac{x^T Ax}{x^T x}}_{\text{Lanczos output}} \geq \underbrace{\frac{v^T Av}{v^T v}, \quad v = A^{k-1}b}_{k-1 \text{ power method}}$, as $v \in \mathcal{K}_k(A,b)$
  - Same for $\lambda_{\min}$, similar for e.g. λ_2

Experiments with Lanczos (nonexaminable)

Symmetric $A \in \mathbb{R}^{n \times n}, n = 100$, Lanczos/power method with random initial vector b



Convergence to dominant eigenvalue



Convergence of all eigenvalues

Arnoldi for nonsymmetric eigenvalue problems (nonexaminable)

Arnoldi for eigenvalue problems: Arnoldi iteration+Rayleigh-Ritz (just like Lanczos alg)

1. Compute $Q^T AQ$
2. Eigenvalue decomposition $Q^T AQ = X\hat{\Lambda}X^{-1}$
3. Approximate eigenvalues diag$(\hat{\Lambda})$ (Ritz values) and eigenvectors QX (Ritz vectors)

As in Lanczos, $Q = Q_k = \mathcal{K}_k(A, b)$, so simply $Q_k^T AQ_k = H_k$ (Hessenberg eigenproblem, ideal for QRalg)

Which eigenvalues are found by Arnoldi?

- Krylov subspace is invariant under shift: $\mathcal{K}_k(A, b) = \mathcal{K}_k(A - sI, b)$
- Thus any eigenvector that power method applied to $A - sI$ converges to should be contained in $\mathcal{K}_k(A, b)$
- To find other (e.g. interior) eigvals, shift-invert Arnoldi: $Q = \mathcal{K}_k((A - sI)^{-1}, b)$

Randomised algorithms in NLA

So far, all algorithms have been deterministic (always same output)

- Direct methods (LU for $Ax = b$, QRalg for $Ax = \lambda x$ or $A = U\Sigma V^T$):
  - Incredibly reliable, backward stable
  - Works like magic if $n \lesssim 10000$
  - But not beyond; cubic complexity $O(n^3)$ or $O(mn^2)$
- Iterative methods (GMRES, CG, Arnoldi, Lanczos)
  - Very fast when it works (nice spectrum etc)
  - Otherwise, not so much; need for preconditioning

Randomised algorithms in NLA

So far, all algorithms have been deterministic (always same output)

- Direct methods (LU for $Ax = b$, QRalg for $Ax = \lambda x$ or $A = U\Sigma V^T$):
  - Incredibly reliable, backward stable
  - Works like magic if $n \lesssim 10000$
  - But not beyond; cubic complexity $O(n^3)$ or $O(mn^2)$
- Iterative methods (GMRES, CG, Arnoldi, Lanczos)
  - Very fast when it works (nice spectrum etc)
  - Otherwise, not so much; need for preconditioning
- Randomised algorithms
  - Output differs at every run
  - Ideally succeed with enormous probability, e.g. $1 - \exp(-cn)$
  - Often by far the fastest&only feasible approach
  - Not for all problems—active field of research

We'll cover two NLA topics where randomisation very successful: **low-rank approximation (randomised SVD)**, and overdetermined **least-squares problems**

Gaussian random matrices

Gaussian $G \in \mathbb{R}^{m \times n}$: Takes iid (independent identically distributed) entries drawn from the standard normal (Gaussian) distribution $G_{ij} \sim N(0, 1)$.

Key properties of Gaussian matrices:

Gaussian random matrices

Gaussian $G \in \mathbb{R}^{m \times n}$: Takes iid (independent identically distributed) entries drawn from the standard normal (Gaussian) distribution $G_{ij} \sim N(0,1)$.

Key properties of Gaussian matrices:

- Orthogonal invariance: If G Gaussian, QGV is also Gaussian for any fixed orthogonal Q, V (independent of G).

Gaussian random matrices

Gaussian $G \in \mathbb{R}^{m \times n}$: Takes iid (independent identically distributed) entries drawn from the standard normal (Gaussian) distribution $G_{ij} \sim N(0,1)$.

Key properties of Gaussian matrices:

- Orthogonal invariance: If G Gaussian, QGV is also Gaussian for any fixed orthogonal Q, V (independent of G).
  1. Linear combination of Gaussian random variables is Gaussian.
  2. The distribution of a Gaussian r.v. is determined by its mean and variance.
  3. $\mathbb{E}[(Qg_i)] = Q\mathbb{E}[g_i] = 0$ (g_i: ith column of G), and $\mathbb{E}[(Qg_i)^T(Qg_i)] = Q\mathbb{E}[g_i^T g_i]Q^T = I$, so each Qg_i is multivariate Gaussian with the same distribution as g_i. Independence of Qg_i, Qg_j is immediate.

Gaussian random matrices

Gaussian $G \in \mathbb{R}^{m\times n}$: Takes iid (independent identically distributed) entries drawn from the standard normal (Gaussian) distribution $G_{ij} \sim N(0,1)$.

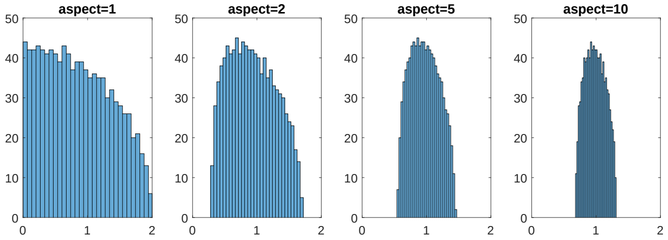
Key properties of Gaussian matrices:

- Orthogonal invariance: If G Gaussian, QGV is also Gaussian for any fixed orthogonal Q, V (independent of G).
  1. Linear combination of Gaussian random variables is Gaussian.
  2. The distribution of a Gaussian r.v. is determined by its mean and variance.
  3. $\mathbb{E}[(Qg_i)] = Q\mathbb{E}[g_i] = 0$ (g_i: ith column of G), and $\mathbb{E}[(Qg_i)^T(Qg_i)] = Q\mathbb{E}[g_i^T g_i]Q^T = I$, so each Qg_i is multivariate Gaussian with the same distribution as g_i. Independence of Qg_i, Qg_j is immediate.

  Alternatively: joint pdf of $g_i = [g_{11}, \ldots, g_{n1}]^T$ is $\frac{1}{(2\pi)^{n/2}}\exp(-\frac{1}{2}(g_{11}^2 + \cdots + g_{n1}^2))$, and that of $Qg_i = [\tilde{g}_{11}, \ldots, \tilde{g}_{n1}]^T$ is (change of variables, note $\det Q = 1$) is $\frac{1}{(2\pi)^{n/2}}\exp(-\frac{1}{2}(\tilde{g}_{11}^2 + \cdots + \tilde{g}_{n1}^2))$
- Marchenko-Pastur rule: "Rectangular random matrices are well conditioned"

Tool from RMT: Rectangular random matrices are well conditioned

Singvals of random matrix $G \in \mathbb{R}^{m\times n}$ $(m \geq n)$ with iid G_{ij} (mean 0, variance 1) follow Marchenko-Pastur (M-P) distribution (proof nonexaminable)



density $\sim \frac{1}{x}\sqrt{((1+\sqrt{\frac{m}{n}})-x)(x-(1-\sqrt{\frac{m}{n}}))}$, support $[\sqrt{m}-\sqrt{n}, \sqrt{m}+\sqrt{n}]$

$\sigma_{\max}(G) \approx \sqrt{m}+\sqrt{n}$, $\sigma_{\min}(G) \approx \sqrt{m}-\sqrt{n}$, hence $\kappa_2(G) \approx \frac{1+\sqrt{m/n}}{1-\sqrt{m/n}} = O(1)$,

Key fact in many breakthroughs in computational maths!

- Randomised SVD, Blendenpik (randomised least-squares)
- (nonexaminable:) Compressed sensing (RIP) [Donoho 06, Candes-Tao 06], Matrix concentration inequalities [Tropp 11], Function approx. by least-squares [Cohen-Davenport-Leviatan 13]

'Fast' (but fragile) alg for $\min_x \|Ax-b\|_2$

$$\min_x \|Ax-b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \ \ m \gg n$$

'Fast' (but fragile) alg for $\min_x \|Ax-b\|_2$

$$\min_x \|Ax-b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \ m \gg n$$

Consider 'row-subselection' algorithm: select $s(>n)$ rows A_1, b_1, and solve $\hat{x} := \operatorname{argmin}_x \|A_1x-b_1\|_2$

- $\hat{x}$ exact solution if $Ax_* = b$ (consistent LS) and A_1 full rank
- If $Ax_* \neq b$, $\hat{x}$ can be terrible: e.g. $A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$ where $A_1 = \epsilon I_n (\epsilon \ll 1)$, and $A_i = I_n$ for $i \geq 2$, and $b_i = b_j$ if $i,j \geq 2$. Then $x_* \approx b_2$, but $\hat{x} = \operatorname{argmin}_x \|A_1x-b_1\|_2$ has $\hat{x} = \frac{1}{\epsilon}b_1$.

'Fast' (but fragile) alg for $\min_x \|Ax - b\|_2$

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \;\; m \gg n$$

Consider 'row-subselection' algorithm: select $s(> n)$ rows A_1, b_1, and solve $\hat{x} := \text{argmin}_x \|A_1 x - b_1\|_2$

- $\hat{x}$ exact solution if $Ax_* = b$ (consistent LS) and A_1 full rank
- If $Ax_* \neq b$, $\hat{x}$ can be terrible: e.g. $A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$ where $A_1 = \epsilon I_n (\epsilon \ll 1)$, and $A_i = I_n$ for $i \geq 2$, and $b_i = b_j$ if $i, j \geq 2$. Then $x_* \approx b_2$, but $\hat{x} = \text{argmin}_x \|A_1 x - b_1\|_2$ has $\hat{x} = \frac{1}{\epsilon} b_1$.

  How to avoid such choices? **Randomisation**

Sketch and solve for $\min_x \|Ax-b\|_2$

A simple randomised algorithm for $\min_x \|Ax-b\|_2$,: *sketch and solve*; draw Gaussian $G \in \mathbb{R}^{s\times m}$ ($n < s \ll m$, e.g. $s = 4n$) and

$$\underset{x}{\text{minimize}} \|G(Ax-b)\|_2.$$

Consider QR fact. $[A\ b] = QR \in \mathbb{C}^{m\times(n+1)}$.

- Note $\boxed{GQ}$ is $s\times n$ Gaussian (by orth. invariance); so $\sigma_i(GQ) \in [\sqrt{s}-\sqrt{n+1}, \sqrt{s}+\sqrt{n+1}]$
- $\|G(Av-b)\|_2 = \|G[A,b]\begin{bmatrix} v \\ -1 \end{bmatrix}\|_2 \leq (\sqrt{s}+\sqrt{n+1})\|R\begin{bmatrix} v \\ -1 \end{bmatrix}\|_2 = (\sqrt{s}+\sqrt{n+1})\|Av-b\|_2$,
  $\forall v$, and similarly $\|G(Av-b)\|_2 \geq (\sqrt{s}-\sqrt{n+1})\|Av-b\|_2$.
- Since by definition $\|G(A\hat{x}-b)\|_2 \leq \|G(Ax-b)\|_2$, it follows that

$$\|A\hat{x}-b\|_2 \leq \frac{1}{\sqrt{s}-\sqrt{n+1}}\|G(Ax-b)\|_2 \leq \frac{\sqrt{s}+\sqrt{n+1}}{\sqrt{s}-\sqrt{n+1}}\|Ax-b\|_2.$$

If $s = 4(n+1)$, we have $\frac{\sqrt{s}+\sqrt{n+1}}{\sqrt{s}-\sqrt{n+1}} = 3$, so $\|Ax_*-b\|_2 = 10^{-10} \Rightarrow \|A\hat{x}-b\|_2 \leq 3\cdot 10^{-10}$

Randomised least-squares: Blendenpik

[Avron-Maymounkov-Toledo 2010]

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \;\; m \gg n$$

- Traditional method: normal eqn $x = (A^TA)^{-1}A^Tb$ or $A = QR, x = R^{-1}(Q^Tb)$, both $O(mn^2)$ cost
- Randomised: generate random $G \in \mathbb{R}^{4n\times m}$, and $GA = \hat{Q}\hat{R}$

  (QR factorisation), then solve $\min_y \|(A\hat{R}^{-1})y - b\|_2$'s normal eqn via Krylov
  - $O(mn\log m + n^3)$ cost using fast FFT-type transforms for G
  - Successful because $A\hat{R}^{-1}$ is well-conditioned

Explaining Blendenpik via Marchenko-Pastur

Claim: $A\hat{R}^{-1}$ is well-conditioned with $G \; A = \hat{Q} \; \hat{R}$ (QR)

Show this for $G \in \mathbb{R}^{4n \times m}$ Gaussian:

Proof: Let $A = QR$. Then $GA = (GQ)R =: \tilde{G}R$

- $\tilde{G}$ is $4n \times n$ rectangular Gaussian, hence well-cond
- So by M-P, $\kappa_2(\tilde{R}^{-1}) = O(1)$ where $\tilde{G} = \tilde{Q}\tilde{R}$ is QR
- Thus $\tilde{G}R = (\tilde{Q}\tilde{R})R = \tilde{Q}(\tilde{R}R) = \tilde{Q}\hat{R}$, so $\hat{R}^{-1} = R^{-1}\tilde{R}^{-1}$
- Hence $A\hat{R}^{-1} = Q\tilde{R}^{-1}$, $\kappa_2(A\hat{R}^{-1}) = \kappa_2(\tilde{R}^{-1}) = O(1)$

Blendenpik: solving $\min_x \|Ax-b\|_2$ using $\hat{R}$

We have $\kappa_2(A\hat{R}^{-1}) =: \kappa_2(B) = O(1)$;
defining $\hat{R}x = y$, $\min_x \|Ax-b\|_2 = \min_y \|(A\hat{R}^{-1})y - b\|_2 = \min_y \|By-b\|_2$

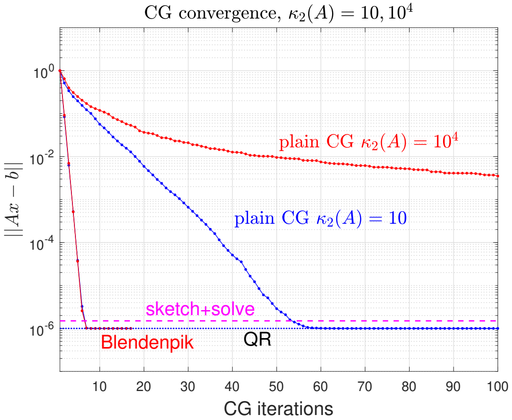
- B well-conditioned⇒in normal equation

$$B^TBy = B^Tb \tag{1}$$

  B well-conditioned $\kappa_2(B) = O(1)$;

- solve (1) via CG (or a tailor-made method LSQR; nonexaminable)
  - exponential convergence, $O(1)$ iterations! (or $O(\log\frac{1}{\epsilon})$ iterations for ϵ accuracy)
  - each iteration requires $w \leftarrow Bw$, consisting of $w \leftarrow \hat{R}^{-1}w$ ($n \times n$ triangular solve) and $w \leftarrow Aw$ ($m \times n$ mat-vec multiplication); $O(mn)$ cost overall
  - In total, $O(mn\log m)$ (fast FFT-based sketching) plus $O(mn\log\frac{1}{\epsilon})$ (CG) cost

Blendenpik experiments



Solving $\min_x \|Ax - b\|_2$ via CG for $A^T Ax = A^T b$ vs. Blendenpik $(AR^{-1})^T (AR^{-1})x = (AR^{-1})^T b$, $m = 10000, n = 100$

In practice, Blendenpik gets $\approx \times 5$ speedup over classical (Householder-QR based) method when $m \gg n$

SVD: the most important matrix decomposition

- **Symmetric eigenvalue decomposition**: $A = V\Lambda V^T$
  for symmetric $A \in \mathbb{R}^{n\times n}$, where $V^TV = I_n$, $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$.
- **Singular Value Decomposition (SVD)**: $A = U\Sigma V^T$
  for any $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Here $U^TU = V^TV = I_n$, $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_n)$,
  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

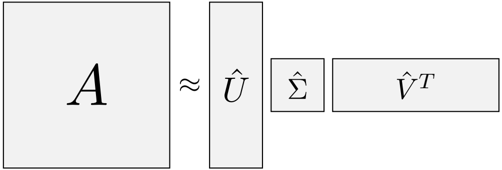
SVD proof: Take Gram matrix A^TA and its eigendecomposition $A^TA = V\Lambda V^T$. Λ is nonnegative, and $(AV)^T(AV)$ is diagonal, so $AV = U\Sigma$ for some orthonormal U. Right-multiply V^T.

SVD useful for

- Finding column space, row space, null space, rank, ...
- Matrix analysis, polar decomposition, ...
- **Low-rank approximation**

(Most) important result in Numerical Linear Algebra

Given $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$, find low-rank (rank r) approximation



, $\hat{\Sigma} \in \mathbb{R}^{r\times r}$

- Optimal solution $A_r = U_r \Sigma_r V_r^T$ via truncated SVD
  $U_r = U(:, 1:r), \Sigma_r = \Sigma(1:r, 1:r), V_r = V(:, 1:r)$, giving

$$\|A - A_r\| = \|\mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_n)\|$$

  in any unitarily invariant norm [Horn-Johnson 1985]
- But that costs $O(mn^2)$ (bidiagonalisation+QR); look for cheaper approximation

Pseudoinverse

Given $M \in \mathbb{R}^{m \times n}$ with economical SVD $M = U_r \Sigma_r V_r^T$ ($U_r \in \mathbb{R}^{m \times r}, \Sigma_r \in \mathbb{R}^{r \times r}, V_r \in \mathbb{R}^{n \times r}$ where $r = \text{rank}(M)$ so that $\Sigma_r \succ 0$), the **pseudoinverse** $M^\dagger$ is

$$M^\dagger = V_r \Sigma_r^{-1} U_r^T \in \mathbb{R}^{n \times m}$$

- satisfies $MM^\dagger M = M$, $M^\dagger M M^\dagger = M^\dagger$, $MM^\dagger = (MM^\dagger)^T$, $M^\dagger M = (M^\dagger M)^T$ (which are often taken to be the definition—above is much simpler IMO)
- $M^\dagger = M^{-1}$ if M nonsingular, $M^\dagger M = I_n (MM^\dagger = I_m)$ if $m \geq n (m \geq n)$ and M full rank
- Given a full-rank underdetermined system $Ax = b$, general solution is $x = A^\dagger b + V_{r,\perp} z$ for arbitrary z, and minimum-norm soln is $x = A^\dagger b$

Randomised SVD by HMT

[Halko-Martinsson-Tropp, SIAM Review 2011]

1. Form a random (Gaussian) matrix $G \in \mathbb{R}^{n \times r}$, usually $r \ll n$.
2. Compute AG.
3. QR factorisation $AG = QR$.
4. $A \approx Q \; Q^T A \; (= (QU_0)\Sigma_0 V_0^T)$ is rank-r approximation.

- $O(mnr)$ cost for dense A
- Near-optimal approximation guarantee: for any $\hat{r} < r$,

$$\mathbb{E}\|A - \hat{A}\|_F \leq \left(1 + \frac{r}{r - \hat{r} - 1}\right) \|A - A_{\hat{r}}\|_F$$

where $A_{\hat{r}}$ is the rank $\hat{r}$-truncated SVD (expectation w.r.t. random matrix X)

Goal: understand this, or at least why $\mathbb{E}\|A - \hat{A}\| = O(1)\|A - A_{\hat{r}}\|$

HMT approximant: analysis (down from 70 pages!)

$\hat{A} = QQ^TA$, where $AG = QR$. Goal: $\|A - \hat{A}\| = \|(I_m - QQ^T)A\| = O(\|A - A_{\hat{r}}\|)$.

1. $QQ^TAG = AG$ (QQ^T is orthogonal projector onto span(AG)). Hence $(I_m - QQ^T)AG = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I_n - GM^T)$ for any $M \in \mathbb{R}^{n\times r}$.

HMT approximant: analysis (down from 70 pages!)

$\hat{A} = QQ^TA$, where $AG = QR$. Goal: $\|A - \hat{A}\| = \|(I_m - QQ^T)A\| = O(\|A - A_{\hat{r}}\|)$.

1. $QQ^TAG = AG$ (QQ^T is orthogonal projector onto span(AG)). Hence $(I_m - QQ^T)AG = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I_n - GM^T)$ for any $M \in \mathbb{R}^{n\times r}$.
2. Set $M^T = (V_1^TG)^\dagger V_1^T$ where $V_1 = [v_1, \ldots, v_{\hat{r}}] \in \mathbb{R}^{n\times\hat{r}}$ top sing vecs of A ($\hat{r} \le r$).

HMT approximant: analysis (down from 70 pages!)

$\hat{A} = QQ^TA$, where $AG = QR$. Goal: $\|A - \hat{A}\| = \|(I_m - QQ^T)A\| = O(\|A - A_{\hat{r}}\|)$.

1. $QQ^TAG = AG$ (QQ^T is orthogonal projector onto span(AG)). Hence $(I_m - QQ^T)AG = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I_n - GM^T)$ for any $M \in \mathbb{R}^{n\times r}$.
2. Set $M^T = (V_1^TG)^\dagger V_1^T$ where $V_1 = [v_1, \dots, v_{\hat{r}}] \in \mathbb{R}^{n\times\hat{r}}$ top sing vecs of A ($\hat{r} \le r$).
3. Then $V_1V_1^T(I - GM^T) = V_1V_1^T(I - G(V_1^TG)^\dagger V_1^T) = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I - V_1V_1^T)(I_n - GM^T)$.

HMT approximant: analysis (down from 70 pages!)

$\hat{A} = QQ^T A$, where $AG = QR$. Goal: $\|A - \hat{A}\| = \|(I_m - QQ^T)A\| = O(\|A - A_{\hat{r}}\|)$.

1. $QQ^T AG = AG$ (QQ^T is orthogonal projector onto span(AG)). Hence $(I_m - QQ^T)AG = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I_n - GM^T)$ for any $M \in \mathbb{R}^{n\times r}$.
2. Set $M^T = (V_1^T G)^\dagger V_1^T$ where $V_1 = [v_1, \dots, v_{\hat{r}}] \in \mathbb{R}^{n\times\hat{r}}$ top sing vecs of A ($\hat{r} \le r$).
3. Then $V_1 V_1^T(I - GM^T) = V_1 V_1^T(I - G(V_1^T G)^\dagger V_1^T) = 0$, so $A - \hat{A} = (I_m - QQ^T)A(I - V_1 V_1^T)(I_n - GM^T)$.
4. Taking norms, $\|A - \hat{A}\|_2 = \|(I_m - QQ^T)A(I - V_1 V_1^T)(I_n - GM^T)\|_2 = \|(I_m - QQ^T)U_2\Sigma_2 V_2^T(I_n - GM^T)\|_2$ where $[V_1, V_2]$ is orthogonal, so

$$\|A - \hat{A}\|_2 \le \|\Sigma_2\|_2 \|(I_n - GM^T)\|_2 = \underbrace{\|\Sigma_2\|_2}_{\text{optimal rank-}\hat{r}} \quad \|GM^T\|_2$$

To see why $\|GM^T\|_2 = O(1)$ (with high probability), we need random matrix theory

$\|GM^T\|_2 = O(1)$

Recall we've shown for $M^T = (V_1^T G)^\dagger V_1^T \; G \in \mathbb{R}^{n \times r}$

$$\|A - \hat{A}\|_2 \le \|\Sigma_2\|_2 \|(I_n - GM^T)\|_2 = \underbrace{\|\Sigma_2\|_2}_{\text{optimal rank-}\hat{r}} \quad \|GM^T\|_2$$

Now $\|GM^T\|_2 = \|G(V_1^T G)^\dagger V_1^T\|_2 = \|G(V_1^T G)^\dagger\|_2 \le \|G\|_2 \|(V_1^T G)^\dagger\|_2$.
Assume G is random Gaussian $G_{ij} \sim \mathcal{N}(0,1)$. Then

- $V_1^T G$ is a Gaussian matrix (orthogonal invariance), hence $\|(V_1^T G)^\dagger\| = 1/\sigma_{\min}(V_1^T G) \lesssim 1/(\sqrt{r} - \sqrt{\hat{r}})$ by M-P
- $\|G\|_2 \lesssim \sqrt{m} + \sqrt{r}$ by M-P

Together we get $\|GM^T\|_2 \lesssim \frac{\sqrt{m}+\sqrt{r}}{\sqrt{r}-\sqrt{\hat{r}}} = \text{"}O(1)\text{"}$

- When G non-Gaussian random matrix, perform similarly, harder to analyze

Precise analysis for HMT (nonexaminable)

A square matrix $P \in \mathbb{R}^{n\times n}$ is called a **projector** if $P^2 = P$

- P diagonalisable and all eigenvalues 1 or 0
- $\|P\|_2 \geq 1$ and $\|P\|_2 = 1$ iff $P = P^T$; in this case P is called orthogonal projector
- $I - P$ is another projector, and unless $P = 0$ or $P = I$, $\|I-P\|_2 = \|P\|_2$:
  Schur form $QPQ^* = \left[\begin{smallmatrix} I & B \\ 0 & 0 \end{smallmatrix}\right]$, $Q(I-P)Q^* = \left[\begin{smallmatrix} 0 & -B \\ 0 & I \end{smallmatrix}\right]$; see [Szyld 2006]

Theorem (Reproduces HMT 2011 Thm.10.5)

If G Gaussian, for any $\hat{r} < r$, $\mathbb{E}\|E_{\mathrm{HMT}}\|_F \leq \sqrt{\mathbb{E}\|E_{\mathrm{HMT}}\|_F^2} = \sqrt{1 + \frac{r}{r-\hat{r}-1}}\|A - A_{\hat{r}}\|_F$.

PROOF. First ineq: Cauchy-Schwarz. Defining $G(V_1^TG)^\dagger V_1^T =: \mathcal{P}_{G,V_1}$ (projector),

$$\begin{aligned}\|E_{\mathrm{HMT}}\|_F^2 =& \|A(I - V_1V_1^T)(I - \mathcal{P}_{G,V_1})\|_F^2 = \|A(I - V_1V_1^T)\|_F^2 + \|A(I - V_1V_1^T)\mathcal{P}_{G,V_1}\|_F^2 \\ &= \|\Sigma_2\|_F^2 + \|\Sigma_2\mathcal{P}_{G,V_1}\|_F^2 = \|\Sigma_2\|_F^2 + \|\Sigma_2(V_2^TG)(V_1^TG)^\dagger V_1^T\|_F^2.\end{aligned}$$

Now if G is Gaussian then $V_2^TG \in \mathbb{R}^{(n-\hat{r})\times r}$ and $V_1^TG \in \mathbb{R}^{\hat{r}\times r}$ are independent Gaussian. Hence by [HMT Prop. 10.1] $\mathbb{E}\|\Sigma_2(V_2^TG)(V_1^TG)^\dagger\|_F^2 = \frac{r}{r-\hat{r}-1}\|\Sigma_2\|_F^2$, so

$$\mathbb{E}\|E_{\mathrm{HMT}}\|_F^2 = \left(1 + \frac{r}{r-\hat{r}-1}\right)\|\Sigma_2\|_F^2.$$

Generalized Nyström (nonexaminable)

$X \in \mathbb{R}^{n\times r}$ Gaussian; set $Y \in \mathbb{R}^{n\times(r+\ell)}$ another Gaussian, and [N. 2020]

$$\hat{A} = (AX(Y^TAX)^\dagger Y^T)A = \mathcal{P}_{AX,Y}A$$

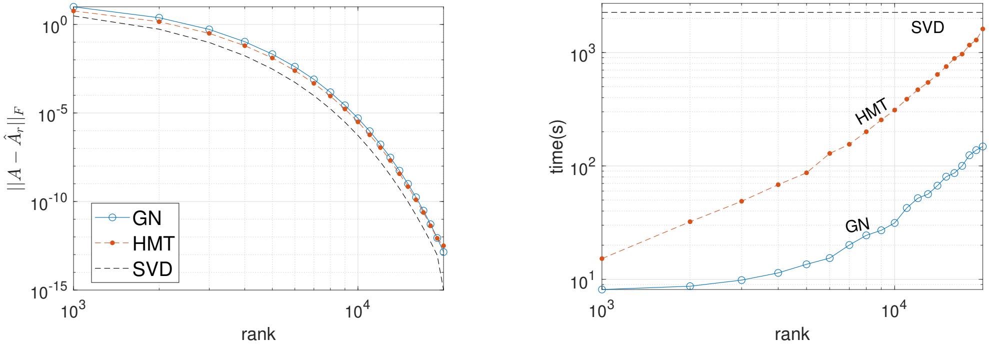
Then $A-\hat{A} = (I-\mathcal{P}_{AX,Y})A = (I-\mathcal{P}_{AX,Y})A(I-XM^T)$; choose M s.t. $XM^T = X(V_1^TX)^\dagger V_1^T = \mathcal{P}_{X,V_1}$. Then $\mathcal{P}_{AX,Y}, \mathcal{P}_{X,V_1}$ projections, and

$$\begin{aligned}\|A-\hat{A}\| &= \|(I-\mathcal{P}_{AX,Y})A(I-\mathcal{P}_{X,V_1})\| \\ &\le \|(I-\mathcal{P}_{AX,Y})A(I-V_1V_1^T)(I-\mathcal{P}_{X,V_1})\| \\ &\le \|A(I-V_1V_1^T)(I-\mathcal{P}_{X,V_1})\| + \|\mathcal{P}_{AX,Y}A(I-V_1V_1^T)(I-\mathcal{P}_{X,V_1})\|.\end{aligned}$$

- Note $\|A(I-V_1V_1^T)(I-\mathcal{P}_{X,V_1})\|$ exact same as HMT error
- Extra term $\|\mathcal{P}_{AX,Y}\|_2 = O(1)$ as before if $c>1$ in $Y \in \mathbb{R}^{m\times cr}$
- Overall, about $(1+\|\mathcal{P}_{AX,Y}\|_2) \approx (1+\frac{\sqrt{n}+\sqrt{r+\ell}}{\sqrt{r+\ell}-\sqrt{r}})$ times bigger expected error than HMT, **still near-optimal** and **much faster** $O(mn\log n + r^3)$

Experiments: dense matrix

Dense $30,000 \times 30,000$ matrix w/ geometrically decaying σ_i



HMT: Halko-Martinsson-Tropp 11, GN: generalized Nyström , SVD: full svd

- Randomised algorithms are very competitive until $r \approx n$
- error $\|A - \hat{A}_r\| = O(\|A - A_{\hat{r}}\|)$, as theory predicts

MATLAB codes

Setup:

```
n = 1000; % size
A = gallery('randsvd',n,1e100); % geometrically decaying singvals
r = 200;  % rank
```

Then

HMT:

```
X = randn(n,r);
AX = A*X;
[Q,R] = qr(AX,0); % QR fact.
At = Q*(Q'*A);

norm(At-A,'fro')/norm(A,'fro')
ans = 1.2832e-15
```

Generalized Nyström :

```
X = randn(n,r); Y = randn(n,1.5*r);
AX = A*X;  YA = Y'*A;  YAX = YA*X;
[Q,R] = qr(YAX,0);  % stable p-inv
At = (AX/R)*(Q'*YA);

norm(At-A,'fro')/norm(A,'fro')
ans = 2.8138e-15
```

Important (N)LA topics not treated

- tensors [Kolda-Bader 2009]
- FFT (values$\leftrightarrow$coefficients map for polynomials) [e.g. Golub and Van Loan 2012]
- sparse direct solvers [Duff, Erisman, Reid 2017]
- multigrid [e.g. Elman-Silvester-Wathen 2014]
- functions of matrices [Higham 2008]
- generalised, polynomial eigenvalue problems [Guttel-Tisseur 2017]
- perturbation theory (Davis-Kahan etc) [Stewart-Sun 1990]
- compressed sensing [Foucart-Rauhut 2013]
- model order reduction [Benner-Gugercin-Willcox 2015]
- communication-avoiding algorithms [e.g. Ballard-Demmel-Holtz-Schwartz 2011]

Numerical Linear Algebra, summary and topics for revision

1st half

- Norms, SVD and its properties (Courant-Fisher etc), applications (low-rank)
- Direct methods (using LU) for linear systems
- Direct methods (using QR fact) least-squares problems
- Stability of algorithms, conditioning

2nd half

- Direct method (QR algorithm) for eigenvalue problems, SVD
- Krylov decomposition (Arnoldi, Lanczos) and Krylov subspace methods for linear systems (GMRES, CG)
- Randomised algorithms for least-squares and SVD

Where does this course lead to?

Courses with significant intersection

- C6.3 Approximation of Functions (Prof. Nick Trefethen, MT): Chebyshev polynomials/approximation theory
- C7.7 Random Matrix Theory (Prof. Jon Keating): for theoretical underpinnings of Randomised NLA
- C6.4 Finite Element Method for PDEs (Prof. Endre Suli): NLA arising in solutions of PDEs
- C6.2 Continuous Optimisation (YN this year): NLA in optimisation problems

and many more: differential equations, data science, optimisation, machine learning,...
NLA is everywhere in computational maths

Thank you for your interest in NLA!