Lecture 2: Numerical linear algebra

- QR factorization
- Eigenvalue decomposition
- Singular value decomposition
- Conditioning of a problem
- Floating point arithmetic and stability of an algorithm

Orthonormal set of vectors

Consider a finite set of vectors $\mathscr{Q} := \{q_1, \ldots, q_k\} \subset \mathbb{R}^n$

- $\mathscr{Q}$ is normalized $:\iff \|q_i\| = 1$, $i = 1, \ldots, k$
- $\mathscr{Q}$ is orthogonal $:\iff q_i \perp q_j$, for all $i \neq j$
- $\mathscr{Q}$ is orthonormal $:\iff \mathscr{Q}$ is orthogonal and normalized

with $Q := \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}$, $\quad \mathscr{Q}$ orthonormal $\iff Q^\top Q = I_k$

Properties:

- orthonormal vectors are independent (show this)
- multiplication with Q preserves norm, $\|Qz\|^2 = z^\top Q^\top Q z = \|z\|^2$
- multiplication with Q preserves inner product, $\langle Qz, Qy \rangle = \langle z, y \rangle$

Orthogonal projectors

Consider an orthonormal matrix $Q \in \mathbb{R}^{n \times k}$ and $\mathscr{L} := \text{span}(Q) \subseteq \mathbb{R}^n$.

The columns of Q form an orthonormal basis for $\mathscr{L}$.

$Q^\top Q = I_k$, however, for $k < n$, $QQ^\top \neq I_n$.

$\Pi_{\text{span}(Q)} := QQ^\top$ is an orthogonal projector on $\text{span}(Q)$, *i.e.*,

$$\Pi_{\mathscr{L}} x = \arg\min_y \|x - y\|_2 \quad \text{subject to} \quad y \in \mathscr{L}$$

Properties: $\Pi = \Pi^2$, $\Pi = \Pi^\top$ (necessary and sufficient for Π orth. proj.)

$\Pi^\perp := (I - \Pi)$ is also an orth. proj., it projects on

$$(\text{span}(\Pi))^\perp \subseteq \mathbb{R}^n \quad \text{— the orthogonal complement of } \text{span}(\Pi)$$

Orthonormal basis for $\mathbb{R}^n$

orthonormal set $\mathcal{Q} := \{q_1, \ldots, q_k\} \subset \mathbb{R}^n$ of $k = n$ vectors

then $Q := \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ is called orthogonal and satisfies $Q^\top Q = I_n$

It follows that $Q^{-1} = Q^\top$ and

$$QQ^\top = \sum_{i=1}^{n} q_i q_i^\top = I_n$$

Expansion in orthonormal basis $x = QQ^\top x$

- $a := Q^\top x$ coordinates of x in the basis $\mathcal{Q}$
- $x = Qa$ reconstruct x from the coordinates a

Geometrically multiplication by Q (and $Q^\top$) is rotation.

Gram-Schmidt (G-S) procedure

Given independent set $\{a_1, \dots, a_k\} \subset \mathbb{R}^n$, G-S produces orthonormal set $\{q_1, \dots, q_k\} \subset \mathbb{R}^n$ such that

$$\operatorname{span}(a_1, \dots, a_r) = \operatorname{span}(q_1, \dots, q_r), \qquad \text{for all } r \le k$$

G-S procedure: Let $q_1 := a_1 / \|a_1\|$. At the ith step $i = 2, \dots, k$

- orthogonalized a_i w.r.t. $q_1, \dots, q_{i-1}$:

$$v_i := \underbrace{(I - \Pi_{\operatorname{span}(q_1,\dots,q_{i-1})}) a_i}_{\text{projection of } a_i \text{ on } (\operatorname{span}(q_1,\dots,q_{i-1}))^\perp}$$

- normalize the result: $q_i := v_i / \|v_i\|$

(A modified version of the G-S procedure is used in practice.)

QR factorization

G-S procedure gives as a byproduct scalars r_{ji}, $j \le i$, $i = 1, \dots, k$, s.t.

$$\begin{aligned} a_i &= (q_1^\top a_i) q_1 + \cdots + (q_{i-1}^\top a_i) q_{i-1} + \|q_i\| q_i \\ &= r_{1i} q_1 + \cdots + r_{ii} q_i \end{aligned}$$

in a matrix form G-S produces the matrix factorization

$$\underbrace{\begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} q_1 & q_1 & \cdots & q_k \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ 0 & r_{22} & \cdots & r_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{kk} \end{bmatrix}}_{R}$$

with orthonormal $Q \in \mathbb{R}^{n \times k}$ and upper triangular $R \in \mathbb{R}^{k \times k}$

If $\{a_1, \ldots, a_k\}$ are dependent, $v_i := (I - \Pi_{\text{span}(q_1,\ldots,q_{i-1})})a_i = 0$ for some i

Conversely, if $v_i = 0$ for some i, a_i is linearly dependent on $\{a_1, \ldots, a_{i-1}\}$

Modified G-S procedure: when $v_i = 0$, skip to the next input vector a_{i+1}

$\implies$ R is in upper staircase form, *e.g.*,

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

(empty elements are zeros)

Which vectors a_i are dependent on $\{a_1, \ldots, a_{i-1}\}$ in this example?

Full QR

$$A = \underbrace{\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}}_{\text{orthogonal}} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \qquad \begin{aligned} \operatorname{span}(A) &= \operatorname{span}(Q_1) \\ \left(\operatorname{span}(A)\right)^{\perp} &= \operatorname{span}(Q_2) \end{aligned}$$

Procedure for finding Q_2:

complete A to a full rank matrix, *e.g.*, $A_{\mathrm{m}} := \begin{bmatrix} A & I \end{bmatrix}$, and apply G-S on A_{m}

In MATLAB:

```
» [Q ,R ] = qr(A) % full QR

» [Q1,R1] = qr(A,0) % reduced QR
```

Eigenvalue decomposition (EVD)

Suppose $\{v_1, \ldots, v_n\}$ is a lin. indep. set of eigenvectors of $A \in \mathbb{R}^{n\times n}$

$$Av_i = \lambda_i v_i, \qquad \text{for } i = 1, \ldots, n$$

written in a matrix form, we have the matrix factorization

$$A \underbrace{\begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}}_{V} = \underbrace{\begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}}_{\Lambda}$$

V is nonsingular, so that

$$AV = V\Lambda \quad \Longrightarrow \quad V^{-1}AV = \Lambda$$

Three applications of EVD

- Compute matrix power A^k, more generally a fun. $f(A)$ of a matrix

$$f(A) = V f(\Lambda) V^{-1} \qquad \text{(assuming } A \text{ diagonalizable)}$$

Example:

$$\begin{bmatrix} 1/3 & 1 \\ 0 & 1/2 \end{bmatrix}^{100} = \; ?$$

Three applications of EVD

- Compute matrix power A^k, more generally a fun. $f(A)$ of a matrix

$$f(A) = V f(\Lambda) V^{-1} \qquad \text{(assuming } A \text{ diagonalizable)}$$

Example:

$$\begin{bmatrix} 1/3 & 1 \\ 0 & 1/2 \end{bmatrix}^{100} = \ ?$$

Eigenvalues: $\lambda_1 = 1/3$, $\lambda_2 = 1/2$, Eigenvectors: $v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $v_2 = \begin{bmatrix} 6 \\ 1 \end{bmatrix}$

Three applications of EVD

- Compute matrix power A^k, more generally a fun. $f(A)$ of a matrix

$$f(A) = Vf(\Lambda)V^{-1} \qquad \text{(assuming } A \text{ diagonalizable)}$$

Example:

$$\begin{bmatrix} 1/3 & 1 \\ 0 & 1/2 \end{bmatrix}^{100} = \; ?$$

Eigenvalues: $\lambda_1 = 1/3$, $\lambda_2 = 1/2$, Eigenvectors: $v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $v_2 = \begin{bmatrix} 6 \\ 1 \end{bmatrix}$

$$\begin{bmatrix} 1/3 & 1 \\ 0 & 1/2 \end{bmatrix}^{100} = \begin{bmatrix} 1 & 6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3^{-100} & \\ & 2^{-100} \end{bmatrix} \begin{bmatrix} 1 & -6 \\ 0 & 1 \end{bmatrix} \approx 0$$

- First order vector linear constant coef. differential/difference eqns

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = Ax(t),\ t \in \mathbb{R}_+ \qquad \text{and} \qquad x(t+1) = Ax(t), t \in \mathbb{Z}_+$$

Given $x(0) \in \mathbb{R}^n$, the equation has a unique solution x.

Qualitative properties of the set of solutions, such as, stability

$$x(t) \to 0 \quad \text{as} \quad t \to \infty$$

are determined by the location of the eigenvalues of A.

- In continuous-time: stability holds $\iff \Re\lambda_i < 0$ for all i
- In discrete-time: stability holds $\iff |\lambda_i| < 1$ for all i

- Principal component analysis (PCA)

given a set of vectors $\{a_1, \dots, a_n\}$,

find an orthonormal set $\{v_1, \dots, v_n\}$, such that

$$\text{span}(a_1, \dots, a_n) \approx \text{span}(v_1, \dots, v_k), \qquad \text{for } k = 1, \dots, n$$

If "$\approx$" means

$$\text{maximize} \quad \Big\| \underbrace{\Pi_{\text{span}(v_1,\dots,v_k)}}_{\text{projection}} \underbrace{\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}}_{A} \Big\|_{\mathrm{F}}$$

the solution $\{v_1, \dots, v_n\}$ is an orthonormal set of eigenvectors of $A^\top A$ ordered according to the magnitude of the eigenvalues.

Used for data compression/recognition (eigenfaces, eigengenes, ...)

Overview of eigenvalue algorithms

- the best ways of computing eigenvalues are not obvious
- bad strategy: rooting the characteristic polynomial
- the power iteration $(\frac{x}{\|x\|}, \frac{Ax}{\|Ax\|}, \frac{A^2x}{\|A^2x\|}, \ldots)$ is not effective in general
- modern general purpose algorithms are based on eigenvalue revealing factorizations
- two stages: Hessenberg form (finite), Schur form (iterative)

Any eigenvalue solver must be iterative

$$p(z) = p_0 + p_1 z + \cdots + z^n \quad \leftrightarrow \quad A = \begin{bmatrix} -p_{n-1} & -p_{n-2} & \cdots & -p_1 & -p_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

roots of p $\quad \leftrightarrow \quad$ eigenvalues of A

Eigenvalue computation is a more general problem than root finding.

No analogue of quadratic formula exists for polynomials of degree ≥ 5.
(Abel 1824)

The aim of eigenvalue solvers is to produce

sequence of numbers that converges rapidly towards eigenvalues.

Rayleigh quotient

Symmetric $A \in \mathbb{R}^{n \times n}$ has n real eigenvalues, which we index as follows

$$\lambda_{\min} := \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n =: \lambda_{\max}$$

Corresponding to $\lambda_1, \ldots, \lambda_n$, we choose orthonormal set of eigenvectors

$$v_1, \ldots, v_n$$

Rayleigh quotient of $v \in \mathbb{R}^n$ (w.r.t. A) is a mapping $r : \mathbb{R}^n \to \mathbb{R}$ defined by

$$r(v) := \frac{v^\top A v}{v^\top v}$$

Note that $r(\alpha v_i) = \lambda_i$, for all $\alpha \in \mathbb{R}$ and $i = 1, \ldots, n$.

Fact: $\min_v r(v) = \lambda_{\min}$ and $\max_v r(v) = \lambda_{\max}$.

Power iteration

- Given: unit norm vector $v^{(0)}$ and symmetric matrix A
- For $k = 1, 2, \dots$ (till convergence)
- Apply A: $w = Av^{(k-1)}$
- Normalize: $v^{(k)} := w/\|w\|$
- Output: eigenvalue/eigenvector of A — $\left((v^{(k)})^\top A v^{(k)}, v^{(k)}\right)$

If $|\lambda_1| > |\lambda_2|$ and $v_1^\top v^{(0)} \neq 0$,

$$v^{(k)} \to \pm v_1$$

with linear convergence rate $O(|\lambda_2/\lambda_1|)$.

Inverse iteration

- Given: unit norm vector $v^{(0)}$, symmetric matrix A, and $\mu \geq 0$
- For $k = 1, 2, \ldots$ (till convergence)
-     Apply $(A - \mu I)^{-1}$: solve $(A - \mu I)w = v^{(k-1)}$
-     Normalize: $v^{(k)} := w / \|w\|$
- Output: eigenvalue/eigenvector of A — $\left((v^{(k)})^\top A v^{(k)}, v^{(k)}\right)$

Let λ be the closest eigenvalue to μ and λ' be the second closest.

Let v be the unit norm eigenvector corresponding to λ. If $v^\top v^{(0)} \neq 0$,

$$v^{(k)} \to \pm v$$

with linear convergence rate $O(|(\mu - \lambda')/(\mu - \lambda)|)$.

Rayleigh quotient iteration

- Given: unit norm vector $v^{(0)}$ and symmetric matrix A
- Let $\lambda^{(0)} := (v^{(0)})^\top A v^{(0)}$
- For $k = 1, 2, \ldots$ (till convergence)
- Apply $(A - \lambda^{(k-1)} I)^{-1}$: solve $(A - \lambda^{(k-1)} I)w = v^{(k-1)}$
- Normalize: $v^{(k)} := w / \|w\|$
- Let $\lambda^{(k)} := (v^{(k)})^\top A v^{(k)}$
- Output: eigenvalue/eigenvector of A — $\left(\lambda^{(k)}, v^{(k)}\right)$

Let λ be the closest eigenvalue to μ and v be the corresponding eigenvector. If $v^\top v^{(0)} \neq 0$,

$$v^{(k)} \to \pm v \qquad \text{with cubic convergence rate.}$$

Exercise

- Implement the power, inverse power, and Rayleigh quotient methods
- Apply them on examples and observe their convergence properties
- Comment on the results

Simultaneous power iteration

Take a set of initial vectors $\{v_1^{(0)}, \ldots, v_p^{(0)}\}$ and consider the iteration:

$$\underbrace{\begin{bmatrix} v_1^{(k+1)} & \cdots & v_p^{(k+1)} \end{bmatrix}}_{V^{(k+1)}} = A \underbrace{\begin{bmatrix} v_1^{(k)} & \cdots & v_p^{(k)} \end{bmatrix}}_{V^{(k)}}, \qquad k = 0, 1, \ldots$$

One can expect that under suitable assumptions

$$\text{span}(v_1^{(k)}, \ldots, v_p^{(k)}) \to \text{span}(v_1, \ldots, v_p), \qquad \text{as} \quad k \to \infty$$

However,

$$v_i^{(k)} \to v_1, \qquad \text{as} \quad k \to \infty, \quad \text{for all } i$$

so $V^{(k+1)}$ becomes increasingly ill-conditioned as $k \to \infty$.

Normalized simultaneous power iteration

- Given: orthonormal matrix $Q^{(0)} \in \mathbb{R}^{n \times p}$ and symmetric matrix A
- For $k = 1, 2, \dots$ (till convergence)
- Apply A: solve $Z = AQ^{(k-1)}$
- Compute orthonormal basis for image(Z):

$$\text{QR factorization:} \quad Q^{(k)} R^{(k)} = Z$$

- Output: orthonormal eigenvectors of A — $Q^{(k)}$

Under suitable assumptions

$$\text{image}(Q^{(k)}) \to \text{span}(v_1, \dots, v_p), \qquad \text{as} \quad k \to \infty.$$

Hessenberg and Schur forms

Every square matrix has a Hessenberg form

$$A = Q \underbrace{\begin{bmatrix} \times & \times & \cdots & \times & \times \\ \times & \times & \cdots & \times & \times \\ & \times & \ddots & \ddots & \times \\ & & \ddots & \ddots & \vdots \\ & & & \times & \times \end{bmatrix}}_{H} Q^\top$$

Q — orthogonal

H — upper Hessenberg

and a Schur form

$$A = UTU^\top$$

U — unitary (complex orthogonal)

T — upper triangular

In MATLAB: `[Q,H] = hess(A)`, `[U,T] = schur(A)`
`[V,L] = eig(A)`

QR algorithm

The basic QR algorithm is normalized simultaneous power iteration with a full set $p = n$ vectors and initial condition $Q^{(0)} = I_n$.

- Given: a symmetric matrix $A^{(0)} = A$
- For $k = 1, 2, \ldots$ (till convergence)
- QR factorization: $A^{(k-1)} = Q^{(k)} R^{(k)}$
- Recombine in reverse order: $A^{(k)} = R^{(k)} Q^{(k)}$
- Output: a Schur decomposition of A — $Q^{(k)}, R^{(k)}$.

$$A^{(k)} = R^{(k)} Q^{(k)} = Q^{(k)\top} A^{(k-1)} Q^{(k)} \quad \Longrightarrow \quad A^{(k)} \text{ is similar to } A^{(k-1)}$$

Additional features of a practical QR algorithm

- Pre-processing: reduce A to tridiagonal form before the iteration
- Shifts: factor $A^{(k)} - \lambda^{(k)} I$, $\lambda^{(k)}$ — eigenvalue estimate
- Deflations: reduce the size of A when and eigenvalue is found

QR algorithm with shifts $\leftrightarrow$ Rayleigh quotient iteration

Generalized eigenvalues

Consider $n \times n$ matrices A and B; the pair (A,B) is called pencil

(v,λ) is a generalized eigenvector/eigenvalue of the pencil (A,B) if

$$Av = \lambda Bv$$

For nonsingular B, the generalized eigenvalue problem is equivalent to

$$B^{-1}Av = \lambda v$$

standard eigenvalue problem

Generalized Rayleigh quotient:

$$\lambda_{\min} = \min_{v \in \mathbb{R}^n} \frac{v^\top Av}{v^\top Bv} \quad , \quad \lambda_{\max} = \max_{v \in \mathbb{R}^n} \frac{v^\top Av}{v^\top Bv}$$

Singular value decomposition

The SVD is used as both computational and analytical tool.

Any $m \times n$ matrix A has an SVD

$$A = \underbrace{\begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}^\top}_{V^\top}$$

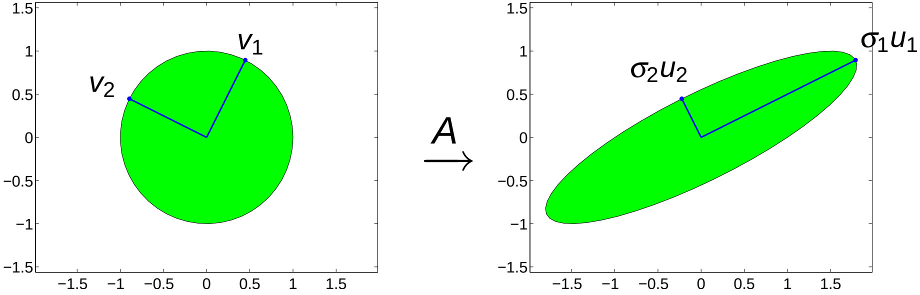
where U and V are orthonormal

- $\sigma_1, \ldots, \sigma_r$ are called singular values
- $u_1, \ldots, u_r$ are called left singular vectors
- $v_1, \ldots, v_r$ are called right singular vectors

Geometric fact motivating the SVD

The image of a unit ball under linear map is a hyperellips.

$$\underbrace{\begin{bmatrix} 1.00 & 1.50 \\ 0 & 1.00 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0.89 & -0.45 \\ 0.45 & 0.89 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 2.00 & 0 \\ 0 & 0.50 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0.45 & -0.89 \\ 0.89 & 0.45 \end{bmatrix}}_{V^\top}$$



Full SVD

Reduced SVD of a matrix $A \in \mathbb{R}^{m\times n}$ of rank r

$$A = U_1 \Sigma_1 U_1 = \begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \begin{bmatrix} v_1^\top \\ \vdots \\ v_r^\top \end{bmatrix}$$

Full SVD: find $U_2 \in \mathbb{R}^{m\times(m-r)}$ and $V_2 \in \mathbb{R}^{n\times(n-r)}$ such that

$$U := \begin{bmatrix} U_1 & U_2 \end{bmatrix} \quad \text{and} \quad V := \begin{bmatrix} V_1 & V_2 \end{bmatrix} \qquad \text{are orthogonal}$$

and add zero rows/columns to Σ_1 to form $\Sigma \in \mathbb{R}^{m\times n}$

Warning: The singular values are $\sigma_1, \dots, \sigma_r$ plus $\min(m-r, n-r)$ zeros

In MATLAB:

```
[U,S,V] = svd(A)      —  full SVD
[U,S,V] = svd(A,0)    —  reduced SVD
```

Link between SVD and EVD

- both SVD and EVD diagonalize a matrix A
- left singular vectors of A are eigenvectors of $AA^\top$
- right singular vectors of A are eigenvectors of $A^\top A$
- the nonzero singular values of A are the square roots of the nonzero eigenvalues of $AA^\top$ or $A^\top A$

  Q: What are the eigenvalues of $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$?
- for a symmetric A, $|\lambda_i| = \sigma_i$

Differences between SVD and EVD

- SVD exists for any matrix

  EVD exist for some square matrices

- SVD applies two orthogonal similarity transformations

  EVD applies one (in general not orthonormal) similarity transf.

- EVD is useful in problems where A is repeatedly applied

  SVD is used to analyse a single application of A on a vector

Conditioning of a problem

Problem: $f : \mathscr{X} \to \mathscr{Y}$, where $\mathscr{X}$ is the data space, $\mathscr{Y}$ is the solutions space

Usually f is a continuous nonlinear function.

Consider a particular data instance $X_0 \in \mathscr{X}$.

The problem f is called well conditioned at the data X_0 if

small perturbations in X lead to small changes in $f(X)$

Absolute condition number: $$\lim_{\delta \to 0} \sup_{\|\widetilde{X}\| < \delta} \frac{\|f(X_0 + \widetilde{X}) - f(X_0)\|}{\|\widetilde{X}\|}$$

Relative condition number: $$\lim_{\delta \to 0} \sup_{\|\widetilde{X}\| < \delta} \frac{\|f(X_0 + \widetilde{X}) - f(X_0)\| / \|f(X_0)\|}{\|\widetilde{X}\| / \|X_0\|}$$

Conditioning of root finding

Given polynomial coefficients $\{p_0, p_1, \ldots, p_n\}$, find its roots $\{\lambda_1, \ldots, \lambda_n\}$

$$p(\lambda) = p_0 + p_1\lambda^1 + \cdots + p_n\lambda^n = c(\lambda - \lambda_1)\cdots(\lambda - \lambda_n)$$

The relative condition number of λ_j w.r.t. perturbation $\widetilde{a}_i$ of a_i is

$$\kappa_{i,j} = |a_i\lambda_j^{i-1}| \Big/ |\tfrac{\mathrm{d}}{\mathrm{d}\lambda}p(\lambda_j)|$$

Example: For $p(\lambda) = \Pi_1^{20}(\lambda - i)$, $\arg\max_{i,j} \kappa_{i,j} = (15, 15)$

```
» roots(poly([1:20]))
ans = 1.0000 ... 14.0684 14.9319 16.0509 ... 20.0003
```

Check the computed roots of $(\lambda - 1)^4$ (`roots(poly([1 1 1 1]))`).

Condition number of matrix–vector product

Theorem: The problem of computing $y = Ax$, given nonsingular matrix $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$ has relative cond. number w.r.t. perturbations in x

$$\kappa = \|A\| \frac{\|x\|}{\|y\|} \leq \|A\| \|A^{-1}\|$$

Condition number of a matrix: $\kappa(A) := \|A\| \|A^{-1}\|$

for nonsquare matrices and 2-norm $\|\cdot\|$, $\kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A)$

A is ill-conditioned if $\kappa(A)$ is large, A is well-conditioned if $\kappa(A)$ is small

$\kappa(A)$ is related to perturbations in the worst case

For an ill-conditioned A, $y = Ax$ may be well-conditioned, for certain x

Condition number of solving systems of equations

Theorem: The problem of computing $x = A^{-1}y$, given $A \in \mathbb{R}^{n\times n}$ and $y \in \mathbb{R}^n$ has relative cond. number $\kappa(A)$ w.r.t. perturbations in A.

Proof: The perturbation $\widetilde{A}$ in A leads to a perturbation $\widetilde{x}$ in x, such that

$$(A+\widetilde{A})(x+\widetilde{x}) = y \quad \Longrightarrow \quad \widetilde{A}x + A\widetilde{x} \stackrel{1}{=} 0$$

"$\stackrel{1}{=}$" means equal up to first order terms in $\widetilde{A}$ and $\widetilde{x}$.

($\kappa(A)$ describes the effect of infinitesimal perturbations.)

$$\widetilde{x} \stackrel{1}{=} -A^{-1}\widetilde{A}x \quad \Longrightarrow \quad \|\widetilde{x}\| \leq \|A^{-1}\|\|\widetilde{A}\|\|x\|$$

$$\Longrightarrow \quad \frac{\|\widetilde{x}\|/\|x\|}{\|\widetilde{A}\|/\|A\|} \leq \|A^{-1}\|\|A\| = \kappa(A) \qquad \square$$

Digital representation of real numbers

IEEE double precision arithmetic:

- Range: $[-2.23 \times 10^{-308}, 1.79 \times 10^{308}]$ overflow/underflow
- Discretization: $[2^i, 2^{i+1}] \mapsto 2^i\{1, 1+2^{-52}, 1+2\times 2^{-52}, \ldots, 2\}$

The gaps between adjacent numbers are in a relative scale at most

$$\varepsilon := 2^{-52} \approx 2.22 \times 10^{-16} \qquad \text{machine precision}$$

- fixed point: the position of the decimal point is fixed
- floating point: its position is stored together with the digits

fixed point leads to uniform absolute errors
floating point leads to uniform relative errors

Rounding: let $\mathrm{fl}(x)$ be the digital representation of $x \in \mathbb{R}$, $|\mathrm{fl}(x) - x| \leq \varepsilon$

Stability of an algorithm

Problem: $f : \mathscr{X} \to \mathscr{Y}$, Algorithm: $\widehat{f} : \mathscr{X} \to \mathscr{Y}$

$\widehat{f}$ is backward stable if for each $X \in \mathscr{X}$ there is $\widehat{X} \in \mathscr{X}$, such that

$$\frac{\|X - \widehat{X}\|}{\|X\|} = O(\varepsilon) \quad \text{and} \quad \widehat{f}(X) = f(\widehat{X})$$

in words:

backward stable algorithm gives the right answer to a nearby problem

$e(\widetilde{X}) := \|\widetilde{X}\| / \|X\| = O(\varepsilon)$ means that there is $c, \delta > 0$ such that

$$\|\widetilde{X}\| < \delta \quad \Longrightarrow \quad |e(\widetilde{X})| \leq c\varepsilon$$

Computational complexity of an algorithm

measured by # of flops (floating point operations) or execution time

1 flop — one addition, subtraction, multiplication, or division

the flops count is usually simplified to leading order terms, *e.g.*, $O(n)$

useful in theoretical comparison of algorithms but it is not an accurate predictor of the computation time

- n vector–vector operations — $O(n)$ flops
  e.g., vector sum, scalar–vector multiplication, inner product
- $m \times n$ matrix–vector product — $O(mn)$ flops
- $m \times n$ matrix – $n \times p$ matrix product — $O(mnp)$ flops

Example: solving $Ax = b$ with general $A \in \mathbb{R}^{n \times n}$ requires $O(n^3)$ flops

Linear equations with special structure

- diagonal — n flops ($x_i = y_i / a_{ii}$ for $i = 1, \ldots, n$)
- lower/upper triangular: n^2 flops (via forward/backward substitution)
- banded — $O(nk)$, where k is the bandwidth
- symmetric — $O(n^3/3)$ (via Cholesky decomposition)
- orthogonal — $O(n^2)$ ($x = A^T y$)
- permutation — 0 flops
- Toeplitz — $O(n^2)$ flops
- combination of banded, symmetric, and Toeplitz

Numerical linear algebra software

Matlab uses as its computational kernel LAPACK

LAPACK is a freely available FORTRAN library

currently the state-of-the-art software for numerical linear algebra

MATLAB provides simple and convenient interface to LAPACK

it is an excellent tool for research; free alternatives to MATLAB are

- octave
- scilab

BLAS and LAPACK

- BLAS — Basic Linear Algebra Subroutines, and
  ATLAS — Automatically Tunable Linear Algebra Subroutines
  - Level 1 BLAS: vector–vector operations
  - Level 2 BLAS: matrix–vector products
  - Level 3 BLAS: matrix–matrix products
- LAPACK — Linear Algebra PACKage
  - Matrix factorizations; exploit triangular, banded, diagonal structures
  - Solvers for linear systems, LS, LN problems; provide error bounds

ScaLAPACK — version for parallel computers.

References

Introductory texts:

- N. Trefethen & Bau, Numerical linear algebra
- G. Stewart, Introduction to matrix computations
- Overton, Numerical computing with IEEE floating point arithmetic

Advanced texts:

- G. Golub & Van Loan, Matrix computations
- N. Higham, Accuracy and stability of numerical methods
- J. Demmel, Applied numerical linear algebra
- LAPACK Users' Guide