Chapter 2

Numerical linear algebra

- Projectors, Gram-Schmidt, and QR factorization
- Computation of eigenvalues and eigenvectors
- Singular value decomposition
- Conditioning of a problem
- Floating point arithmetic and stability of an algorithm

2.1 Projectors, Gram-Schmidt, and QR factorization

Projectors

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

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

Exercise problem 23. Show that an orthonormal set of vectors is independent.

□

Define the matrix $Q := \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}$ and note that $\mathscr{Q}$ is orthonormal if and only if $Q^\top Q = I_k$. A matrix Q, such that $Q^\top Q = I_k$, is sometimes called *orthonormal* (this is not a completely standard terminology).

Exercise problem 24. Show that

- multiplication with an orthonormal matrix Q *preserves norm*, i.e., $\|Qz\|^2 = z^\top Q^\top Qz = \|z\|^2$, and
- multiplication with an orthonormal matrix Q *preserves inner product*, i.e., $\langle Qz, Qy \rangle = \langle z, y \rangle$.

□

Consider an orthonormal matrix $Q \in \mathbb{R}^{n \times k}$ and the subspace $\mathscr{L}$ spanned by the columns of Q, i.e.,

$$\mathscr{L} := \text{image}(Q) \subseteq \mathbb{R}^n.$$

The columns of Q form an *orthonormal basis* for $\mathscr{L}$. Since Q is orthonormal, $Q^\top Q = I_k$, however, for $k < n$,

$$QQ^\top \neq I_n.$$

Exercise problem 25. The matrix $\Pi_{\text{image}(Q)} := QQ^\top$ is an *orthogonal projector* onto $\mathscr{L}$, i.e.,

$$\Pi_{\mathscr{L}} x = \arg\min_y \|x - y\|_2 \quad \text{subject to} \quad y \in \mathscr{L} \tag{2.1}$$

□

Exercise problem 26. Show that necessary and sufficient conditions for Π to be a projector are

1. $\Pi = \Pi^2$ (Π is idempotent), and
2. $\Pi = \Pi^\top$ (Π is symmetric).

□

Exercise problem 27. Show that the matrix

$$\Pi^\perp := I - \Pi$$

is also an orthogonal projector. The projector $\Pi^\perp$ is called the *complementary projector* to Π. Define for a set $\mathscr{S} \subset \mathbb{R}^n$, its *orthogonal complement*

$$\mathscr{S}^\perp := \{ x \in \mathbb{R}^n \mid x^\top y = 0 \text{ for all } y \in \mathscr{S} \}.$$

Show that for any set $\mathscr{S} \subset \mathbb{R}^n$, its orthogonal complement $\mathscr{S}^\perp$ is a subspace and that $\Pi^\perp$ projects onto the orthogonal complement $\big(\text{image}(\Pi)\big)^\perp$ of $\text{image}(\Pi)$.

□

If $\mathscr{Q} := \{ q_1, \ldots, q_k \} \subset \mathbb{R}^n$ is an *orthonormal set* of $k = n$ vectors, then the matrix $Q := \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ is called *orthogonal*. A matrix $Q \in \mathbb{R}^n$ is orthogonal if and only if it satisfies the identities

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

It follows that for an orthogonal matrix Q, $Q^{-1} = Q^\top$. The identity $x = QQ^\top x$ is an expansion of x in the orthonormal basis given by the columns of Q.

- $\widetilde{x} := Q^\top x$ is the vector of the coordinates of x in the basis $\mathscr{Q}$, and
- $x = Q\widetilde{x}$ reconstructs x in the standard basis $\{ e_1, \ldots, e_n \}$.

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

Exercise problem 28. Verify that a matrix representation of rotation in $\mathbb{R}^2$ is an orthogonal matrix.

□

Gram-Schmidt procedure and QR factorization

Given an independent set of vectors $\{ a_1, \ldots, a_k \} \subset \mathbb{R}^n$, the Gram-Schmidt procedure, see Algorithm 1, produces an orthonormal set $\{ q_1, \ldots, q_k \} \subset \mathbb{R}^n$, such that

$$\text{span}(a_1, \ldots, a_r) = \text{span}(q_1, \ldots, q_r), \qquad \text{for all } r \leq k.$$

Algorithm 1 Gram-Schmidt

Input: $\{ a_1, \ldots, a_k \} \subset \mathbb{R}^n$

1: $q_1 := a_1 / \|a_1\|$
2: **for** $i = 2, \ldots, k$ **do**
3: $\quad v_i := (I - \Pi_{\text{image}(q_1, \ldots, q_{i-1})}) a_i$ {project a_i onto $\big(\text{span}(q_1, \ldots, q_{i-1})\big)^\perp$}
4: $\quad q_i := v_i / \|v_i\|$ {normalize}
5: **end for**

Output: $\{ q_1, \ldots, q_k \} \subset \mathbb{R}^n$

From $a_i \in \text{span}(q_1, \ldots, q_k)$, it follows that

$$a_i = r_{1i} q_1 + \cdots + r_{ii} q_i \tag{2.2}$$

for some scalars r_{ij}, for $i \leq j$ and $j = 1, \ldots, k$.

Exercise problem 29. Show that $r_{ij} = q_i^\top a_j$.

□

Written in a matrix form (2.2) is what is called the QR matrix factorization

$$\underbrace{\begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} q_1 & q_1 & \cdots & q_k \end{bmatrix}}_{Q_1} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ 0 & r_{22} & \cdots & r_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{kk} \end{bmatrix}}_{R_1}.$$

Here $Q_1 \in \mathbb{R}^{n\times k}$ is orthonormal and $R_1 \in \mathbb{R}^{k\times k}$ is upper triangular.

The Gram-Schmidt procedure applies a sequence of linear operations on the columns of A, such that the resulting matrix has orthonormal columns. In a matrix form this is expressed by $A\widetilde{R}_1 = Q_1$, where the matrix $\widetilde{R}_1$ encodes the operations of the Gram-Schmidt procedure. Note that $\widetilde{R}_1 = R_1^{-1}$, where R_1 is the upper triangular matrix of the QR factorization $A = Q_1R_1$.

Exercise problem 30. Show that the inverse $\widetilde{R}_1$ of a nonsingular upper triangular matrix R_1 is again upper triangular

□

Therefore, the Gram-Schmidt procedure can be termed "triangular orthogonalization".

There is an alternative procedure for computing the QR factorization that applies a sequence of orthogonal transformations on the columns of A, aiming to produce an upper triangular matrix. In a matrix form this is expressed by $\widetilde{Q}_1 A = R_1$, where Q_1

$$\widetilde{Q}_1 = \widetilde{Q}^{(1)} \cdots \widetilde{Q}^{(n)}$$

encodes the sequence of orthogonal transformations.

Exercise problem 31. Show that the product of orthonormal matrices $\widetilde{Q}^{(1)}, \ldots, \widetilde{Q}^{(n)}$ is orthonormal.

□

$\widetilde{Q} = Q_1^\top$ is the orthogonal matrix of the QR factorization $A = Q_1R_1$. The alternative process of "orthogonal triangularization" turns our to be numerically more stable than the Gram-Schmidt procedure and is the preferred way of computing the QR factorization

If $\{a_1, \ldots, a_k\}$ is a set of dependent vectors, $v_i := (I - \Pi_{\text{span}(q_1,\ldots,q_{i-1})})a_i = 0$ for some i. Conversely, if $v_i = 0$ for some i, a_i is linearly dependent on $a_1, \ldots, a_{i-1}$. The Gram-Schmidt produces can handle linearly dependent sets by skipping to the next input vector a_{i+1} whenever $v_i = 0$. As a result, the matrix R_1 is in *upper staircase form*, e.g.,

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & & & \times \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix}, \tag{2.3}$$

with all empty elements being zeros.

Exercise problem 32. Which vectors are linearly dependent in the example of (2.3)?

□

The factorization

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = QR$$

with $Q := \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ orthogonal and R_1 upper triangular is called *full QR factorization*.

Exercise problem 33. Show that

- $\text{image}(A) = \text{image}(Q_1)$, and

- $\big(\text{image}(A)\big)^{\perp} = \text{image}(Q_2)$.

□

A method for finding the full QR factorization of A is to complete A to a full rank matrix, e.g., $A_{\text{m}} := \begin{bmatrix} A & I \end{bmatrix}$, and apply the Gram-Schmidt procedure on A_{m}. In MATLAB, `[Q,R] = qr(A)` procedures the full QR and `[Q1,R1] = qr(A,0)` procedures the reduced QR.

2.2 Computation of eigenvalues and eigenvectors

Three applications of the eigenvalue decomposition

- *Evaluation of a function $f : \mathbb{R} \to \mathbb{R}$ at a square matrix.* Consider a real analytic function $x \mapsto f(x)$ with a Taylor series expansion

$$f(x) = \frac{1}{0!} f(0)x^0 + \frac{1}{1!}\big(\tfrac{\mathrm{d}}{\mathrm{d}t} f\big)(0)x^1 + \frac{1}{2!}\big(\tfrac{\mathrm{d}^2}{\mathrm{d}t^2} f\big)(0)x^2 + \cdots$$

A matrix valued matrix function $X \mapsto f(X)$, where $X \in \mathbb{R}^{n\times n}$, corresponding to the scalar valued scalar function $x \mapsto f(x)$, where $x \in \mathbb{R}$ is defined by the series

$$f(X) := \frac{1}{0!} f(0)X^0 + \frac{1}{1!}\big(\tfrac{\mathrm{d}}{\mathrm{d}t} f\big)(0)X^1 + \frac{1}{2!}\big(\tfrac{\mathrm{d}^2}{\mathrm{d}t^2} f\big)(0)X^2 + \cdots$$

Exercise problem 34. Assuming that X is diagonalizable, i.e., there is a nonsingular matrix V and a diagonal matrix Λ, such that

$$X = V^{-1}\Lambda V, \qquad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n),$$

show that

$$f(X) = V \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{bmatrix} V^{-1} =: V f(\Lambda) V^{-1}.$$

□

- *Stability of linear time-invariant systems.* Consider the first order vector linear constant coefficients differential and difference equations

$$\tfrac{\mathrm{d}}{\mathrm{d}t}x(t) = Ax(t), \text{ for } t \in \mathbb{R}_+ \qquad \text{and} \qquad x(t+1) = Ax(t), \text{ for } t \in \mathbb{Z}_+. \tag{2.4}$$

Given $x(0) \in \mathbb{R}^n$, the equations (2.4) have unique solutions x. Qualitative properties of the set of solutions, such as, *stability*, i.e.,

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

are determined by the *location of the eigenvalues of A.*

Exercise problem 35. Show that for the differential equation (continuous-time system), stability holds if and only if $\Re(\lambda_i) < 0$ for all i and for the difference equation (discrete-time system), stability holds if $|\lambda_i| < 1$ for all i.

□

Stability of (2.4), however, can be checked without computing the eigenvalues $\lambda_1, \ldots, \lambda_n$, cf., the Routh–Hurwitz (continuous-time systems) and Schur–Cohn (discrete-time systems) tests [Jur74].

- *Principal component analysis (PCA).* Principal component analysis is a statistical technique. Here we give an alternative deterministic formulation. Given a set of vectors $\{a_1, \ldots, a_n\}$, find

$$\{b_1^*, \ldots, b_j^*\} := \arg\max_{b_1,\ldots,b_j} \left\| \Pi_{\text{span}(b_1,\ldots,b_j)} \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \right\|_{\text{F}} \qquad \text{subject to} \qquad \begin{bmatrix} b_1 & \cdots & b_j^* \end{bmatrix}^{\top} \begin{bmatrix} b_1 & \cdots & b_j^* \end{bmatrix} = I \tag{2.5}$$

(Recall that $\Pi_{\text{span}(b_1,\ldots,b_j)}$ is the orthogonal projector onto $\text{span}(b_1, \ldots, b_j)$.)

Exercise problem 36. Shown that the solution $\{b_1^*,\dots,b_n^*\}$ to (2.5) is the orthonormal set of eigenvectors $\{v_1,\dots,v_j\}$ of the matrix $A^\top A$, corresponding to the largest eigenvalues.

□

Eigenvalue/eigenvector computation algorithms

Eigenvalue computation is closely related to rooting a polynomial.

Exercise problem 37. Show that the set of eigenvalues of the *companion matrix*

$$C_p := \begin{bmatrix} -p_{n-1} & -p_{n-2} & \cdots & -p_1 & -p_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

coincides with the set of roots of a polynomial

$$p(z) = p_0 + p_1 z + \cdots + p_n z^n.$$

□

A classical results in mathematics due to Abel, is that there is no analogue of the formula for the roots of a quadratic polynomial for a general polynomial of degree more than 4. By the above link between roots of a polynomial and eigenvalues of a matrix, we conclude that the same must be true for the eigenvalues of a general matrix of dimension more than 4×4.

> There is no algorithm that can compute the eigenvalues of a general matrix of dimension more than 4×4 in a finite number of operations.

Eigenvalue algorithms must be *iterative* and in finite time produce only an *approximation* of the eigenvalues. The aim in the design of eigenvalue algorithms is to increase the converges speed and reduce the number of operations per iteration, so that the sequence of the eigenvalue approximations produced by the algorithm converges rapidly to the eigenvalues.

The power iteration, inverse power iteration, and Rayleigh quotient iteration are basic methods for computing an eigenvalue/eigenvector pair of a matrix and are ingredients of the modern algorithms for eigenvalue computation. Next we outline these algorithms and for simplicity we restrict to the symmetric case. A symmetric $A\in\mathbb{R}^{n\times n}$ has n real eigenvalues, which we index as follows

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

Corresponding to $\lambda_1,\dots,\lambda_n$, we choose an orthonormal set of eigenvectors $q_1,\dots,q_n$. The Rayleigh quotient of $v\in\mathbb{R}^n$ (with respect to 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 $r(\alpha q_i)=\lambda_i$, for all $\alpha\in\mathbb{R}$ and $i=1,\dots,n$.

Exercise problem 38. Show that $\min_v r(v)=\lambda_{\min}$ and $\max_v r(v)=\lambda_{\max}$.

□

Exercise problem 39. Show that if $|\lambda_1|>|\lambda_2|$ and $v_1^\top v^{(0)}\neq 0$, then $v^{(k)}\to\pm v_1$ with *linear convergence* rate $O(|\lambda_2/\lambda_1|)$.

□

Let λ be the closest eigenvalue to μ and λ' be the second closest. Let v be the unit norm eigenvector corresponding to λ. Then if $v^\top v^{(0)}\neq 0$, then $v^{(k)}\to\pm v$ with *linear convergence* rate $O(|(\mu-\lambda')/(\mu-\lambda)|)$.

Let λ be the closest eigenvalue to μ and v be the corresponding eigenvector. Then, if $v^\top v^{(0)}\neq 0$, then $v^{(k)}\to\pm v$ with *cubic convergence rate.*

Algorithm 2 Power iteration.

Input: unit norm vector $v^{(0)}$ and a symmetric matrix A.

1: **for** $k = 1,2,\ldots$ (till convergence) **do**
2: $w := Av^{(k-1)}$ {apply A}
3: $v^{(k)} := w/\|w\|$ {normalize}
4: **end for**

Output: eigenvalue/eigenvector of A — $\left((v^{(k)})^\top Av^{(k)}, v^{(k)}\right)$

Algorithm 3 Inverse iteration.

Input: unit norm vector $v^{(0)}$, a symmetric matrix A, and $\mu \geq 0$.

1: **for** $k = 1,2,\ldots$ (till convergence) **do**
2: $(A-\mu I)w = v^{(k-1)}$ {apply $(A-\mu I)^{-1}$}
3: $v^{(k)} := w/\|w\|$ {normalize}
4: **end for**

Output: eigenvalue/eigenvector of A — $\left((v^{(k)})^\top Av^{(k)}, v^{(k)}\right)$

Exercise problem 40. Implement the power, inverse power, and Rayleigh quotient methods. Apply them on examples and observe their convergence properties. Comment on the results.

□

Normalized 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

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

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 $k \to \infty$. This problem is resolved by changing the computed basis on each iteration step to an equivalent orthonormal basis.

Under suitable assumptions

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

Algorithm 4 Rayleigh quotient iteration

Input: unit norm vector $v^{(0)}$ and symmetric matrix A

1: **for** $k = 1,2,\ldots$ (till convergence) **do**
2: $(A-\lambda^{(k-1)}I)w = v^{(k-1)}$ {apply $(A-\lambda^{(k-1)}I)^{-1}$}
3: $v^{(k)} := w/\|w\|$ {normalize}
4: $\lambda^{(k)} := (v^{(k)})^\top Av^{(k)}$ {eigenvalue estimate}
5: **end for**

Output: eigenvalue/eigenvector of A — $\left(\lambda^{(k)}, v^{(k)}\right)$

Algorithm 5 Normalized simultaneous power iteration.

Input: orthonormal matrix $Q^{(0)} \in \mathbb{R}^{n\times p}$ and symmetric matrix A.

1: **for** $k = 1,2,\ldots$ (till convergence) **do**
2: $Z = AQ^{(k-1)}$ {apply A}
3: QR factorization $Q^{(k)}R^{(k)} = Z$ {compute orthonormal basis for image(Z)}
4: **end for**

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

QR algorithm

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

$$A^{(k)} = R^{(k)}Q^{(k)} = {Q^{(k)}}^{\top}A^{(k-1)}Q^{(k)},$$

so that $A^{(k)}$ is similar to $A^{(k-1)}$.

Additional features of a practical QR algorithm are

- *pre-processing:* reduce A to tridiagonal form before the iteration,
- *shifts:* factor $A^{(k)} - \lambda^{(k)}I$ instead of $A^{(k)}$, where $\lambda^{(k)}$ is an eigenvalue estimate, and
- *deflations:* reduce the size of A when and eigenvalue is found.

Algorithm 6 QR algorithm.

Input: symmetric matrix $A^{(0)} = A$.

1: **for** $k = 1,2,\ldots$ (till convergence) **do**
2: $A^{(k-1)} = Q^{(k)}R^{(k)}$ {QR factorization}
3: $A^{(k)} = R^{(k)}Q^{(k)}$ {recombine in reverse order}
4: **end for**

Output: a Schur decomposition of A — $Q^{(k)}, R^{(k)}$.

The QR algorithm with shifts corresponds to the Rayleigh quotient iteration.

Note 41 (Hessenberg and Schur forms). The analogue of the tridiagonal form for nonsymmetric matrices is the Hessenberg form

$$H := \begin{bmatrix} \times & \times & \cdots & \times & \times \\ \times & \times & \cdots & \times & \times \\ & \times & \ddots & \ddots & \times \\ & & \ddots & \ddots & \vdots \\ & & & \times & \times \end{bmatrix}$$

There is an orthogonal similarity transformation that brings any square matrix to a Hessenberg form. Modern algorithms for eigenvalue factorization of a general matrix A consist of two distinct stages:

1. reduction of A by an orthogonal similarity transformation to *Hessenberg form*, and
2. iterative convergence by a sequence of unitary similarity transformation to a *Schur form*.

Let U^* be the conjugate transposed of $U \in \mathbb{C}^{m\times n}$. The matrix U is unitary if $U = U^*$. The Schur form R of A is an upper triangular matrix, such that $R = U^*AU$, where U is a unitary matrix. The Schur form is *eigenvalue revealing* because the eigenvalues of A appear on the diagonal of R. The first stage requires a finite number of operations and is introduced for the purpose of speeding up the convergence on the second stage, which requires an infinite number of operations.

Generalized eigenvalues

A pair (A,B) of square matrices $A,B \in \mathbb{R}^{n\times n}$ is called a *pencil*. A generalized eigenvector/eigenvalue (v,λ) of the pencil (A,B) is a vector $v \in \mathbb{C}^n$ and a scalar $\lambda \in \mathbb{C}$, such that

$$Av = \lambda Bv.$$

For a nonsingular matrix B, the generalized eigenvalue problem is equivalent to a standard eigenvalue problem

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

If A and B are symmetric matrices, the pencil (A,B) is called symmetric and has real generalized eigenvalues. The generalized Rayleigh quotient is a mapping $r : \mathbb{R}^n \to \mathbb{R}$ defined by

$$r_{A,B}(v) := \frac{v^\top Av}{v^\top Bv}$$

It has analogous properties to the Rayleigh quotient.

Exercise problem 42. Consider a symmetric pencil (A,B) and let $\lambda_{\min}/\lambda_{\max}$ be the minimal/maximal generalized eigenvalue of (A,B). Show that

$$\lambda_{\min} = \min_{v\in\mathbb{R}^n} r_{A,B}(v) \quad \text{and} \quad \lambda_{\max} = \max_{v\in\mathbb{R}^n} r_{A,B}(v).$$

□

2.3 Singular value decomposition

The singular value decomposition is used as both computational and analytical tool. Any matrix $A \in \mathbb{R}^{m\times n}$ has a singular value decomposition

$$A = \underbrace{\begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix}}_{U_1} \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}}_{\Sigma_1} \underbrace{\begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}^\top}_{V_1^\top}, \tag{2.6}$$

where U_1 and V_1 are orthonormal and $\sigma_1,\ldots,\sigma_r$ are positive scalars, called *singular values* of A. The columns of U, $u_1,\ldots,u_r$ are called *left singular vectors* and the columns of V, $v_1,\ldots,v_r$ are called *right singular vectors* of A.

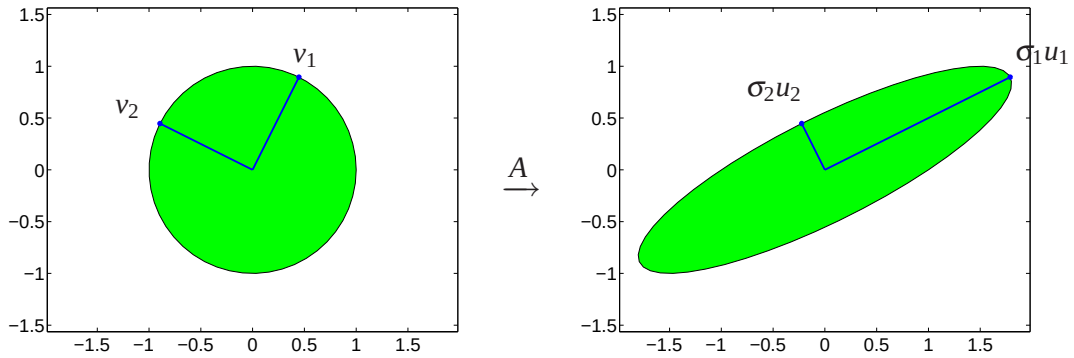
A geometric fact motivating the singular value decomposition is:

the image $\mathscr{E} = A\mathscr{U}$ of a unit ball $\mathscr{U}$ under a linear map $x \mapsto Ax$ is a hyperellips.

Example 43. Consider the following example

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

The image of the unit ball under the linear map defined by A is



The vector v_1 is mapped by A to the vector $\sigma_1 u_1$, and the vector v_2 is mapped by A to the vector $\sigma_2 u_2$. From all unit norm inputs vectors, the vector v_1 achieves the maximum 2-norm output vector $\sigma_1 u_1$, and v_2 achieves the minimum 2-norm output vector $\sigma_2 u_2$. Note that $\|\sigma_i u_i\| = \sigma_i$.

The decomposition (2.6) is sometimes called the *reduced SVD* of a matrix $A \in \mathbb{R}^{m\times n}$ in order to distinguish it from the *full SVD*

$$A = U\Sigma V,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal and

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{m\times n}.$$

Compared with the reduced SVD, the full SVD computes matrices $U_2 \in \mathbb{R}^{m\times(m-r)}$ and $V_2 \in \mathbb{R}^{n\times(n-r)}$, such that

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

are orthogonal and adds zero rows/columns to Σ_1 to form $\Sigma \in \mathbb{R}^{m\times n}$. Note that the singular values of A are $\sigma_1, \ldots, \sigma_r$ *and* $\min(m-r, n-r)$ zeros. In MATLAB the full SVD is computed via `[U,S,V] = svd(A)` and the reduced SVD via `[U,S,V] = svd(A,0)`.

Similarities between SVD and EVD: Both the SVD and EVD diagonalize a matrix A. The left singular vectors of A are eigenvectors of $AA^\top$, the 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 $A^\top A$. In particular, for a symmetric A, $|\lambda_i| = \sigma_i$ and for a positive definite matrix $\lambda_i = \sigma_i$.

Exercise problem 44. Using the fact that $AA^\top$ or $A^\top A$ have the same nonzero eigenvalues, argue (without doing computations) that the eigenvalues of $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ are $3, 0, 0$.

□

Differences between SVD and EVD: Despite the above similarities, there are important differences between the SVD and EVD. The SVD exists for *any matrix*, while the EVD exist for nondefective square matrices. The SVD applies *two orthogonal similarity transformations*, while the EVD applies one (in general not orthogonal) similarity transformation. The SVD is used to analyse a *single application of A* on a vector, while the EVD is useful in problems where *A is repeatedly applied*.

2.4 Conditioning and stability

Abstractly a computational problem is a mapping $f : \mathscr{X} \to \mathscr{Y}$ from a data space $\mathscr{X}$ to a solutions space $\mathscr{Y}$. Usually f is a continuous nonlinear function (even though the problem is in th realm of the linear algebra). A particular data instance is an element $X_0 \in \mathscr{X}$. The problem f is called *well conditioned* at the data X_0 if

small perturbations in X_0 lead to small changes in $f(X_0)$

The *absolute condition number* of the problem f at the data instance X_0 is the derivative of f with respect to X at X_0

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

The *relative condition number* is defined as

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

Example 45 (Conditioning of root finding). The roots finding problem is: Given polynomial coefficients $\{p_0, p_1, \ldots, p_n\}$, find its roots $\{\lambda_1, \ldots, \lambda_n\}$, i.e.,

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

The relative condition number of λ_j with respect to perturbation $\widetilde{a}_i$ of a_i is

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

For the polynomial $p(\lambda) = (\lambda - 1)\cdots(\lambda - 20)$, $\arg\max_{i,j} \kappa_{i,j} = (15, 15)$ and in MATLAB we obtain

```
>> roots(poly([1:20]))

ans = 1.0000   ...    14.0684   14.9319   16.0509   ...   20.0003
```

The function `poly` gives the coefficients of a monic polynomial with roots, specified by the input argument. (This operation is recursive multiplication of a polynomial by monomials or equivalently convolution of a sequence by a sequence with length two.) The function `roots` solves the roots finding problem, i.e., its argument is a vector of polynomial coefficients and the results is a vector of the roots. In exact arithmetic `roots(poly([1:20]))` should return the answer $\{1, \ldots, 20\}$, however, due to the round-off errors in the finite precision arithmetic the answer is not exact. The experiment shows that the 15th root is the one that has the largest perturbation.

Exercise problem 46. Check the computed roots of $(\lambda - 1)^4$ (`roots(poly([1 1 1 1]))`).

Condition number of matrix–vector product

Theorem 47. *The problem of computing $y = Ax$ for given nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and a vector $x \in \mathbb{R}^n$ has relative condition number with respect to perturbations in x*

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

For a nonsingular matrix A, the number

$$\kappa(A) := \|A\|\|A^{-1}\|$$

is called the *condition number of A*. For a general (nonsquare) matrix and 2-norm $\|\cdot\|$,

$$\kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A).$$

The matrix A is called *ill-conditioned* if $\kappa(A)$ is "large", and A is called *well-conditioned* if $\kappa(A)$ is "small". (Here "large" and "small" depend on the size of the expected perturbations, so that they depend on the application.) The number $\kappa(A)$ is related to perturbations in the worst case. For an ill-conditioned A, the problem $y = Ax$ may still be well-conditioned at certain x's.

Condition number of solving systems of equations

The relative condition number of the computational problem "solve a system of equations $y = Ax$, $A \in \mathbb{R}^{n\times n}$, with respect to perturbation in y" is given by Theorem 47. Indeed, provided A is nonsingular, $x = A^{-1}y$, which is a matrix–vector product problem. (The matrix now is A^{-1}.) If A is singular, the problem is infinitely ill-conditioned. Another term for this case is *ill-posed* problem.

Theorem 48. *The problem of computing $x = A^{-1}y$, given $A \in \mathbb{R}^{n\times n}$ and $y \in \mathbb{R}^n$ has relative condition number $\kappa(A)$ with respect to perturbations in A.*

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

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

Here "$\stackrel{1}{=}$" means equal up to first order terms in $\widetilde{A}$ and $\widetilde{x}$. ($\kappa(A)$ describes the effect of infinitesimal perturbations.)

$$\begin{aligned} \widetilde{x} \stackrel{1}{=} -A^{-1}\widetilde{A}x \quad &\Longrightarrow \quad \|\widetilde{x}\| \le \|A^{-1}\|\|\widetilde{A}\|\|x\| \\ &\Longrightarrow \quad \frac{\|\widetilde{x}\|/\|x\|}{\|\widetilde{A}\|/\|A\|} \le \|A^{-1}\|\|A\| = \kappa(A). \end{aligned}$$

□

Digital representation of real numbers

The IEEE double precision arithmetic is a widely used standard for digital representation of real numbers. It is characterized by

- range: $[-2.23 \times 10^{-308}, 1.79 \times 10^{308}]$, and
- discretization: $[2^i, 2^{i+1}] \mapsto 2^i\{1, 1+2^{-52}, 1+2\times 2^{-52}, \ldots, 2\}$.

If a number large than 10^{308} is produced during computations, an exception called an *overflow* occurs. Conversely, if a number smaller than 10^{-308} is produced, an *underflow* occurs. In the case of underflow, the number is rounded to zero. Different systems react in different ways to overflow, which is a more serious exception.

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

$$\varepsilon := 2^{-52} \approx 2.22 \times 10^{-16}.$$

The number $\varepsilon := 2^{-52}$ is called the *machine precision*.

In *fixed point* arithmetic the position of the decimal point is fixed. In *floating point* arithmetic the position of the decimal point is stored together with the digits. Fixed point arithmetic leads to uniform absolute errors, while floating point arithmetic leads to uniform relative errors.

Let $\mathrm{fl}(x)$ be the digital representation of $x \in \mathbb{R}$, the error $|\mathrm{fl}(x) - x| \le \varepsilon$ is called *rounding error*.

Stability of an algorithm

Recall the general definition of a computational problem as a mapping $f : \mathscr{X} \to \mathscr{Y}$. A computational algorithm, aiming to solve the problem f, is another mapping $\widehat{f} : \mathscr{X} \to \mathscr{Y}$ from the data to the solution space. The algorithm $\widehat{f}$ is called *backward stable* if for each $X \in \mathscr{X}$ there is $\widehat{X} \in \mathscr{X}$, such that

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

i.e.,

backward stable algorithm gives the right answer to a nearby problem

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

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

Computational complexity of an algorithm

The computational complexity of an algorithm is measured by the number of floating point operations (flops) or by the execution time. One flop is defined as one addition, subtraction, multiplication, or division. The flops count is usually simplified to leading order terms, e.g., $O(n)$. It is useful in theoretical comparison of algorithms but on modern computers it is not an accurate predictor of the computation time.

Standard linear algebra operations have the following computational complexities:

- n vector–vector operations (vector sum, scalar–vector multiplication, inner product) — $O(n)$ flops

- $m \times n$ matrix–vector product — $O(mn)$ flops
- $m \times n$ matrix – $n \times p$ matrix product — $O(mnp)$ flops
- solving $Ax = b$ with general $A \in \mathbb{R}^{n \times n}$ — $O(n^3)$ flops

However if the matrix A has special structure and this structure is exploited, the computational complexities may be lower. The following are typical examples of matrix structures and the corresponding computational complexities for solving systems of equations with such matrices:

- 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

Currently the state-of-the-art software for numerical linear algebra are the Basic Linear Algebra Subroutines (BLAS) and Linear Algebra PACKage (LAPACK) libraries. BLAS has functions for the lower-level linear algebra operations, such as vector–vector operations (Level 1 BLAS), matrix–vector products (Level 2 BLAS), and matrix–matrix products (Level 3 BLAS). Although these operations are conceptually simple and algorithmically straightforward, their implementation on a computer with distributed memory, is a highly nontrivial task.

The LAPACK library contains functions for higher level operations such as matrix factorizations and solvers for linear systems, least-squares, and least-norm problems. There are special functions for structured problems involving triangular, banded, diagonal matrices and the solvers provide error bounds based on perturbation analysis.

Notes and references

A excellent concise introduction to numerical linear algebra is given by Trefethen and Bau [TB97]. Classic reference on this subject is the book of Golub and Van Loan [GV96]. The book of Demmel [Dem97] is with a particular emphasis on writing numerical software. Perturbation theory is treated in [SS90] and stability analysis is extensively treated by Higham [Hig91]. The IEEE floating point arithmetic is presented in [Ove01].

Bibliography

[Dem97] J Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[GV96] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[Hig91] N. Higham. *Accuracy and Stability of Numerical Methods*. SIAM, 1991.

[Jur74] E. Jury. *Inners and stability of dynamic systems*. Wiley, 1974.

[Ove01] M. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM, 2001.

[SS90] G. Stewart and J. Sun. *Matrix perturbation theory*. Academic Press, Boston, 1990.

[TB97] L. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.