

Daniel Leitner

CO-MAT2

Lecture Notes

Wintersemester 2013/14

Computational Science Center
Universität Wien
Oskar-Morgenstern-Platz 1
A-1090 Wien

Contents

1	Eigenvalues	1
1.1	Eigenvalue problems	1
1.1.1	Estimation of eigenvalues	2
1.1.2	Power iteration	4
1.1.3	QR algorithm	6
1.2	Generalized eigenvalue problem	6
1.3	Singular values	7
2	Iteration methods	9
2.1	Fixed point iteration	9
2.1.1	Jacobi method	10
2.1.2	Gauss-Seidel method	10
2.1.3	Successive over-relaxation (SOR)	11
2.2	Krylov subspace methods	11
2.2.1	Arnoldi iteration	12
2.2.2	Lanzcos iteration	14
2.2.3	Generalized minimal residuals (GMRES)	14
2.2.4	Conjugate gradients (CG)	15
3	Nonlinear systems of equations	19
3.1	Newton's method	19
3.1.1	One-dimensional geometric motivation	19
3.1.2	Higher-dimensional generalisation	20
3.2	Quasi-Newton methods	21
3.2.1	One-dimensional motivation: Secant method	21
3.2.2	Higher-dimensional generalisation	22
3.3	Basic line search concepts	23
3.3.1	Armijo	23
3.3.2	Goldstein and Price	23
3.3.3	Wolfe	23
4	Stochastic Processes	25
4.1	Basic concepts	25
4.2	Markov chains	25

5	Random numbers and Monte Carlo Simulation	27
5.1	Pseudorandom number generators	27
5.2	Tests of pseudorandom numbers	27
5.3	Basic concepts of Monte Carlo Simulation	27
6	Multivariate Integration	29
6.1	Monte Carlo method	29
6.2	Quasi-Monte Carlo method	29

Chapter 1

Eigenvalues

In this chapter we will consider a matrix $A \in \mathbb{C}^{n \times n}$ and find its eigenvalues and eigenvectors. We use the more general case of complex matrices because even for real matrices eigenvalues and eigenvectors can be complex.

1.1 Eigenvalue problems

For a given matrix A we search for pairs of *eigenvalues* $\lambda \in \mathbb{C}$ and *eigenvectors* $x \in \mathbb{C}^n$ that satisfy

$$Ax = \lambda x \quad x \in \mathbb{C}^n \setminus \{0\}, \lambda \in \mathbb{C}.$$

We can find the eigenvalues by the roots λ_i of the *characteristic polynomial*

$$\chi(z) = \det(A - zI) = (z - \lambda_1)(z - \lambda_2) \dots (z - \lambda_n),$$

which is a non-linear problem. We define the *algebraic multiplicity* of an eigenvalue λ to be its multiplicity as a root of $\chi(z)$.

For each eigenvalue λ we can calculate the corresponding eigenvectors solving the linear equation

$$(A - \lambda I) v_j = 0.$$

The eigenvectors v_j together with the zero vector form the *eigenspace* denoted with E_λ . The number of linear independent eigenvectors (i.e. the dimension of E_λ) is called the *geometric multiplicity* of the eigenvalue λ .

Example 1.1.1. Consider the two matrices

$$A = \begin{bmatrix} 2 & & \\ & 2 & \\ & & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{bmatrix}.$$

What is the characteristic polynomial, eigenvalues, algebraic and geometric multiplicities?

An eigenvalue whose algebraic multiplicity is greater than its geometric multiplicity is called a *defective eigenvalue*. A matrix with at least one defective eigenvalue is called a *defective matrix*.

We call the matrices A and B *similar* if there exists a non-singular $X \in \mathbb{C}^{n \times n}$ such that $B = XAX^{-1}$. Similar matrices have the same characteristic polynomial, eigenvalues, and algebraic and geometric multiplicities.

Numerical algorithms to find eigenvalues are **not** based on finding roots of the characteristic polynomial, since this is a highly unstable problem. Eigenvalues are either computed by power iteration (see Section 1.1.2) or by eigenvalue revealing factorizations (see Section 1.1.3).

Lemma 1.1.2. *The most important matrix factorizations are:*

1. *If A is nondefective a diagonalisation $A = X\Lambda X^{-1}$ exists, where Λ is a diagonal matrix.*
2. *If A is normal (i.e. $AA^* = A^*A$) an unitary diagonalisation $A = Q\Lambda Q^*$ exists, where Q is an unitary matrix (i.e. $Q^* = Q^{-1}$), and Λ is a diagonal matrix. Note that all hermitian matrices are normal.*
3. *An unitary triangulation (Schur factorization) $A = QTQ^*$ always exists, where Q is unitary and T is upper triangular.*

Typical applications of eigenvalues are situations where we are interested in A^k or e^{At} .

1.1.1 Estimation of eigenvalues

For a quick estimation of the eigenvalues we can use

Theorem 1.1.3. (Gershgorin circle theorem) *Given a matrix $A = [a_{ij}] \in \mathbb{C}^{n \times n}$, then for every eigenvalue λ of A , holds*

$$\lambda \in \bigcup_{i=1}^n \mathcal{K}_i,$$

with the Gershgorin discs

$$\mathcal{K}_i := \{\zeta \in \mathbb{C} : |\zeta - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}.$$

Proof. Let $Ax = \lambda x$ with $x = [x_i] \neq 0$. Then there exists an index i , so that $|x_j| \leq |x_i|$ for all $j \neq i$. $(Ax)_i$ denotes the i -th component of Ax , then

$$\lambda x_i = (Ax)_i = \sum_{j=1}^n a_{ij} x_j$$

and therefore,

$$|\lambda - a_{ii}| = \left| \sum_{j \neq i} a_{ij} \frac{x_j}{x_i} \right| \leq \sum_{j \neq i} |a_{ij}|.$$

Therefore, $\lambda \in \mathcal{K}_i \subseteq \bigcup_{j=1}^n \mathcal{K}_j$. □

Example 1.1.4. What are the Gershgorin discs of

$$A = \begin{bmatrix} 4 & 0 & -3 \\ 0 & -1 & 1 \\ -1 & 1 & 0 \end{bmatrix} ?$$

We can further confine the range in which the eigenvalues are using the *Rayleigh quotient*. The Rayleigh quotient of a vector x is given by

$$r(x) := \frac{x^* A x}{x^* x}, \quad x \in \mathbb{C}^n \setminus \{0\}.$$

Note that $r(v_i) = \lambda_i$, and for each vector x the Rayleigh quotient gives the value $\alpha := r(x)$ which acts most like an eigenvalue (i.e. minimizes $\|Ax - \alpha x\|_2$).

The range of $r(x)$

$$\mathcal{W}(A) := \left\{ \frac{x^* A x}{x^* x} : x \in \mathbb{C}^n \setminus \{0\} \right\} \subseteq \mathbb{C}.$$

is called the *numerical range* of the matrix A . Notably, the eigenvalues of A lie in the numerical range of A .

Further properties of $\mathcal{W}(A)$ are

1. $\mathcal{W}(A)$ is connected.
2. If A is hermitian, then $\mathcal{W}(A)$ is the real interval $[\lambda_{\min}, \lambda_{\max}]$.
3. If A is *skew-symmetric* (i.e. $A^* = -A$), then $\mathcal{W}(A)$ is the convex hull ($\subseteq \mathbb{C}$) of all eigenvalues of A .

Every matrix $A \in \mathbb{C}^{n \times n}$ can be split into an hermitian part (first term) and a skew symmetric part (second term):

$$A = \frac{A + A^*}{2} + \frac{A - A^*}{2}.$$

From this split and the above properties we directly derive:

Theorem 1.1.5. (Theorem of Bendixon)

$$\sigma(A) \subseteq \mathcal{W}\left(\frac{A + A^*}{2}\right) \oplus \mathcal{W}\left(\frac{A - A^*}{2}\right),$$

where $\sigma(A)$ is the spectrum of $A \in \mathbb{C}^{n \times n}$ (i.e. the set containing all eigenvalues of A).

Example 1.1.6. We use the theorem to further confine the range of Example 1.1.4. First we calculate the hermitian and skew-symmetric part

$$H = \frac{A + A^t}{2} = \begin{bmatrix} 4 & 0 & -2 \\ 0 & -1 & 1 \\ -2 & 1 & 0 \end{bmatrix}, \quad S = \frac{A - A^t}{2} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

The spectra of H and S can be estimated by the theorem of Gerschgorin: This yields

$$\mathcal{R} = [-3, 6] \times [-i, i] \supset \mathcal{W}(A) \supset \sigma(A).$$

The spectrum of A must lie in the intersection of $\mathcal{R}$ and the Gerschgorin discs of Example 1.1.4. The actual spectrum of A is

$$\sigma(A) = \{-1.7878, 0.1198, 4.6679\}.$$

1.1.2 Power iteration

In this section we numerically calculate eigenvalues and eigenvectors by a method called *power iteration*. For simplicity, we only consider symmetric real matrices. Such matrices are diagonalisable and their eigenvectors form an orthonormal basis. Furthermore, we sort the absolute values of the eigenvalues in descending orders $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n| \geq 0$.

We consider the sequence $z^{(k+1)} = Az^{(k)} / \|Az^{(k)}\|$. Under certain assumptions the sequence converges to the eigenvector corresponding to the largest eigenvalue of A . The approach is outlined in Algorithm 1.

Data: Matrix A , initial eigenvector $z^{(0)}$ with $\|z^{(0)}\| = 1$
Initialisation: set $k = 1$
while *convergence criterion not satisfied* **do**
 $\tilde{z}^{(k)} := Az^{(k-1)}$ apply A
 $z^{(k)} := \frac{\tilde{z}^{(k)}}{\|\tilde{z}^{(k)}\|}$ normalise
 $k \leftarrow k + 1$
end

Algorithm 1: Power iteration

The estimation for the eigenvalue can be found by applying the Rayleigh quotient to the resulting $z^{(k)}$,

$$\lambda^{(k)} := z^{(k)*} A z^{(k)}.$$

We analyse power iteration by writing the initial guess $z^{(0)}$ as a linear combination of the orthonormal eigenvectors q_i :

$$z^{(0)} = a_1 q_1 + a_2 q_2 + \dots + a_n q_n \quad (1.1)$$

The value $z^{(k)}$ is a multiple of $A^k z^{(0)}$, therefore we write

$$z^{(k)} = c_k A^k z^{(0)},$$

where c_k is some scalar constant. Note that $Aq_1 = \lambda_1 q_1$, thus inserting Eqn (1.1) yields

$$\begin{aligned} z^{(k)} &= c_k (a_1 \lambda_1^k q_1 + a_2 \lambda_2^k q_2 + \dots + a_n \lambda_n^k q_n) \\ z^{(k)} &= c_k \lambda_1^k (a_1 q_1 + a_2 (\lambda_2/\lambda_1)^k q_2 + \dots + (\lambda_n/\lambda_1)^k q_n) \end{aligned}$$

If k becomes big, the terms (λ_j/λ_1) for $j = 2 \dots n$ become small if $|\lambda_1| > |\lambda_j|$, and therefore $z^{(k)} \rightarrow (c_k \lambda_1^k a_1) q_1$. Thus for all $z^{(0)}$, where $a_1 \neq 0$ the proposed Algorithm 1 will converge to the eigenvector corresponding to the largest eigenvalue of A .

The algorithm is very simple, but of limited use, since it only finds the eigenvector corresponding to the largest eigenvalue, and if λ_1 is not much larger

than λ_2 convergence is very slow. To speed things up we would need a large large eigenvalue $|\lambda_1|$.

We improve the algorithm using the following idea: For any $\mu \in \mathbb{R}$ that is not an eigenvalue of A , the eigenvectors of A are the same as of $(A - \mu I)^{-1}$, and if λ_i is eigenvalue of A then $(\lambda_i - \mu)^{-1}$ is eigenvalue of $(A - \mu I)^{-1}$.

Consequently, if we choose a value μ that is close to an eigenvalue λ_j . Then $(\lambda_i - \mu)^{-1}$ will be large, and therefore the eigenvector of $A - \mu I$ can be computed fast. This leads to the method called *inverse iteration* described in Algorithm 2.

The algorithm computes the eigenvector corresponding to the eigenvalue nearest to μ . Inverse iteration is the standard method for calculating eigenvectors if the eigenvalues are already known.

Data: Matrix A , initial μ close to the desired eigenvalue, initial vector $z^{(0)}$ with $\|z^{(0)}\| = 1$
Initialisation: set $k = 1$
while *convergence criterion not satisfied* **do**
 solve $(A - \mu I)w = z^{(k-1)}$ for w apply $(A - \mu I)^{-1}$
 $z^{(k)} := \frac{w}{\|w\|}$ normalise
 $k \leftarrow k + 1$
end

Algorithm 2: Inverse iteration

We can further improve inverse iteration by continuously improving the eigenvalue estimate μ in each step to increase the rate of convergence. Therefore, we use the Rayleigh quotient

$$r(z) := \frac{z^* A z}{z^* z}, \quad z \in \mathbb{C}^n \setminus \{0\}.$$

to estimate the eigenvalue λ from the estimated (normed) eigenvector z , thus $\lambda = z^T A z$ (this minimizes $\|Ax - \lambda x\|$). This leads to the *Rayleigh quotient iteration* outlined in Algorithm 3. This algorithm is very fast and shows a cubic convergence rate.

Data: Matrix A , initial vector $z^{(0)}$ with $\|z^{(0)}\| = 1$
Initialisation: set $k = 1$, let $\lambda^{(0)} := (z^{(0)})^T A z^{(0)}$
while *convergence criterion not satisfied* **do**
 solve $(A - \lambda^{(k-1)} I)w = z^{(k-1)}$ for w apply $(A - \lambda^{(k-1)} I)^{-1}$
 $z^{(k)} := \frac{w}{\|w\|}$ normalise
 $\lambda^{(k)} := (z^{(k)})^T A z^{(k)}$ Rayleigh quotient
 $k \leftarrow k + 1$
end

Algorithm 3: Rayleigh quotient iteration

1.1.3 QR algorithm

The QR algorithm is a stable and simple procedure for calculating all eigenvalues and eigenvectors (see Algorithm 4).

First, we note that in each iteration of the algorithm we use only (stable) similarity transformation. Thus $A^{(k)}$ and $A^{(k-1)}$ are similar, since $R^{(k)} = (Q^{(k)})^* A^{(k-1)}$, and therefore $A^{(k)} = (Q^{(k)})^* A^{(k-1)} Q^{(k)}$. Thus, if $A^{(k)}$ converges, this matrix has the same eigenvalues and eigenvectors as $A^{(0)}$.

Second, QR algorithm can be seen as a more sophisticated variation of power iteration (Algorithm 1). Instead of using one single vector, QR algorithm works with a complete orthonormal basis of vectors, using the QR decomposition to orthogonalize. For a hermitian matrix A the algorithm converges to $AQ = Q\Lambda$, where Λ is the diagonal matrix of eigenvalues to which A converged. Q is an orthogonal matrix, where the columns of Q are the eigenvectors. For the non-hermitian case, the algorithm converges to the a Schur factorisation of the matrix (see Lemma 1.1.2).

Data: Matrix A Initialisation: set $k = 1$, $A^{(0)} = A$ while <i>convergence criterion not satisfied</i> do $Q^{(k)} R^{(k)} = A^{(k-1)}$ QR factorization of $A^{(k-1)}$ $A^{(k)} = R^{(k)} Q^{(k)}$ recombine in reverse order $k \leftarrow k + 1$ end

Algorithm 4: QR algorithm

QR algorithm is the standard algorithm for computing all eigenvalues of a matrix. The algorithm can achieve cubic convergence for hermitian matrices, given the following enhancements:

1. Initially, A is reduced to tridiagonal form using Householder transformations.
2. Instead of $A^{(k)}$, the shifted matrix $A^{(k)} - \mu_k I$ is factored, where μ_k is an eigenvalue estimate (same idea as Rayleigh quotient iteration, Algorithm 3).
3. When possibly $A^{(k)}$ is split into submatrices (a divide and conquer strategy).

1.2 Generalized eigenvalue problem

The problem of finding a vector $v_i \in \mathbb{C}^n \setminus \{0\}$ that obeys

$$Av_i = \lambda_i Bv_i$$

for $A, B \in \mathbb{C}^{n \times n}$ is called a *general eigenvalue problem* and the *generalised eigenvalue* $\lambda_i \in \mathbb{C}$ obeys the equation

$$\det(A - \lambda_i B) = 0.$$

In this case we can find n linearly independent vectors v_i so that $Av_i = \lambda_i Bv_i$ the following equality holds

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

where X is the matrix composed of the eigenvectors v_i and Λ is a diagonal matrix of the eigenvalues λ_i .

For generalised eigenvalue problem the Rayleigh quotient is defined as

$$r(x) := \frac{x^* Ax}{x^* Bx}.$$

With this definition it is easy to generalize the Rayleigh quotient iteration (Algorithm 3) to calculate generalized eigenvectors.

Example 1.2.1. Consider the matrices

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Then, the characteristic polynomial takes the form $\chi(z) = z^2 + 1$, which results to the set of the eigenvalues $\sigma(A, B) = \pm i$. Even though both matrices are symmetric the pair of them exhibits complex conjugate eigenvalues.

1.3 Singular values

Let $A \in \mathbb{C}^{m \times n}$ with $\text{range}(A) = p \leq \min\{m, n\}$, then there exists a unique factorisation called *singular value decomposition* (SVD) such that

$$A = U\Sigma V^*,$$

where $U := [u_1, \dots, u_m] \in \mathbb{C}^{m \times m}$ is unitary ($U^*U = UU^* = I$), and $V := [v_1, \dots, v_n] \in \mathbb{C}^{n \times n}$ is unitary, with $\{u_j\}_{j=1}^m$ and $\{v_j\}_{j=1}^n$ being orthonormal bases on $\mathbb{C}^m$ and $\mathbb{C}^n$, respectively. $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal

$$\Sigma := \left[\begin{array}{cc|c} \sigma_1 & 0 & 0 \\ & \ddots & \vdots \\ 0 & \sigma_p & 0 \\ \hline 0 & \dots & 0 \end{array} \right],$$

with real values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p > 0$.

A geometrical interpretation of the SVD is that the image of a unit sphere under any $m \times n$ matrix is a hyperellipse, where the vectors $\sigma_i u_i$ are the principal semiaxis. Note that

$$Av_j = \sigma_j u_j \quad \text{for } j = 1 \dots p,$$

and

$$U^*AV = \Sigma.$$

The SVD shows that all matrices are diagonal under the proper bases for the domain and range spaces.

There are fundamental differences between singular values and eigenvalue decompositions (compare with Lemma 1.1.2):

- SVD uses different bases, eigenvalue decomposition just one.
- SVD uses orthonormal bases, in eigenvalue decomposition this is generally not the case (only if A is normal).
- Not all matrices have an eigenvalue decomposition, but all (even rectangular) matrices have a SVD.

Theorem 1.3.1. *Nonzero singular values of A are the square roots of the nonzero eigenvalues of A^*A and AA^* .*

Theorem 1.3.2. *If A is hermitian (i.e. $A = A^*$) the singular values of A are the absolute values of the eigenvalues of A .*

Thus, numerical computation of singular values could be based on calculating the eigenvalues of AA^* . However, this method is unstable, since the condition number of AA^* might be much bigger than that of A .

Stable SVD algorithms are based on finding the eigenvalues of the self-adjoint block matrix

$$M = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix},$$

which are the singular values of A with positive and negative sign.

Chapter 2

Iteration methods

Iterative methods try to find an approximate solution $x \in \mathbb{R}^n$ to the linear equation $Ax = b$ or the eigenvalue problem $Ax = \lambda x$, where $A \in \mathbb{R}^{n \times n}$ is very large and typically sparse. Such matrices appear frequently in the applied sciences e.g. as stiffness matrices from partial differential equations.

2.1 Fixed point iteration

We will construct iterative methods based on an important result from analysis:

Theorem 2.1.1. (Banach fixed-point theorem) *Let $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a contraction, i.e.*

$$\|\Phi(x) - \Phi(y)\| \leq q\|x - y\| \text{ for } q < 1 \text{ for all } x, y \in \mathbb{R}^n .$$

The fix point equation $x = \Phi(x)$ has exactly one solution $\hat{x} \in \mathbb{R}^n$, and the iteration $\{x^{(k)}\}$ with $x^{(0)} \in \mathbb{R}^n$, $x^{(k+1)} = \Phi(x^{(k)})$ for $k = 0, 1, 2, \dots$, converges to the solution $\hat{x}$ for $k \rightarrow \infty$. Furthermore, for $k \geq 1$

1. $\|x^{(k)} - \hat{x}\| \leq q\|x^{(k-1)} - \hat{x}\|$ (monotony)
2. $\|x^{(k)} - \hat{x}\| \leq \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\|$ (a-priori bound)
3. $\|x^{(k)} - \hat{x}\| \leq \frac{q}{1-q}\|x^{(k)} - x^{(k-1)}\|$ (a-posteriori bound)

The basic idea is that we choose a splitting

$$A = M - N,$$

with a matrix M that can be easily inverted. With this splitting we write $Ax = b$ as fix point equation

$$\begin{aligned} Mx &= Nx + b, \text{ or} \\ x &= Tx + c, \end{aligned}$$

where $T = M^{-1}N$ and $c = M^{-1}b$. Thus, the iteration $x^{(k+1)} = \Phi(x^{(k)})$ is given by $\Phi(x) = Tx + c$.

By choosing M we can construct different iterative methods to solve $Ax = b$, and we can use Theorem 2.1.1 to analyse its behaviour. Note that it is essential that M^{-1} is easy to compute.

The framework is outlined in Algorithm 5. Choices of M and N are presented in the following sections.

Data: Matrix $A = [a_{ij}]$, initial vector $x^{(0)}$
Initialisation: set $k = 0$
while *convergence criterion not satisfied* **do**
 for $i = 1 : n$ **do**
 $\tilde{x}_i^{(k+1)} := \Phi(x^k)$ with Φ according to Eqns (2.1), (2.2), or (2.3)
 end
 $k \leftarrow k + 1$
end

Algorithm 5: Stationary iterative methods

2.1.1 Jacobi method

Jacobi methods splits the matrix $A = [a_{ij}]$ into its diagonal elements $M = \text{diag}(a_{11}, \dots, a_{nn})$ and $N = M - A$, thus the inverse can be easily calculated: $M^{-1} = \text{diag}(\frac{1}{a_{11}}, \dots, \frac{1}{a_{nn}})$. Therefore, we write

$$x^{(k+1)} = M^{-1}(b + Nx^{(k)})$$

or component-wise

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right). \quad (2.1)$$

From fix-point theorem 2.1.1 we know that the iteration converges if we can find a $q < 1$ such that $\|\Phi(x) - \Phi(y)\| = \|M^{-1}N(x - y)\| \leq q\|x - y\|$. From $\|M^{-1}N(x - y)\| \leq \|M^{-1}N\|\|x - y\|$ we conclude that $\|M^{-1}N\| \leq q$. Therefore, if $\|M^{-1}N\| < 1$ the method will converge. Especially, this is the case when A is a diagonal dominant matrix. Note that Jacobi iteration is very simple to parallelise.

2.1.2 Gauss-Seidel method

Gauss-Seidel method accelerates convergence of the Jacobi method by using the already computed vector components $x_j^{(k+1)}$ for $j < i$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right). \quad (2.2)$$

We rewrite this equation as

$$a_{ii}x_i^{(k+1)} + \sum_{j<i} a_{ij}x_j^{(k+1)} = b_i - \sum_{j>i} a_{ij}x_j^{(k)}$$

and in matrix form

$$(D - L)x^{(k+1)} = b + Ux^{(k)},$$

where $D = \text{diag}(a_{11}, \dots, a_{nn})$ and L a strictly lower triangular matrix, and U a strictly upper triangular matrix. Thus, we choose $M = D - L$ and $N = M - A = U$, and again can use the fix-point theorem 2.1.1 to analyse convergence.

2.1.3 Successive over-relaxation (SOR)

The SOR is a variant of Gauss-Seidel method that improves convergence rate by using a linear interpolation between the last iterate $x^{(k)}$ and the Gauss-Seidel iterate (Eqn 2.2). The method uses the relaxation parameter $\omega \in (0, 2)$, which is often chosen heuristically in dependence of the specific problem. The iteration is given by

$$x_i^{(k+1)} = x_i^{(k)}(1 - \omega) + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)} \right), \quad (2.3)$$

and in matrix form

$$x^{(k+1)} = (1 - \omega)x^{(k)} + \omega D^{-1} (b + Lx^{(k+1)} + Ux^{(k)}).$$

The same approach can be applied to the Jacobi method. This is sometimes called *JOR method*.

2.2 Krylov subspace methods

The idea of all Krylov subspace methods is to project a n -dimensional problem into a lower-dimensional m -Krylov subspace.

Definition 2.2.1. Let $b \in \mathbb{R}^n$. The m^{th} *Krylov-space* of A with respect to b is defined as

$$\mathcal{K}_m(A, b) := \langle b, Ab, A^2b, \dots, A^{m-1}b \rangle.$$

That is, $\mathcal{K}_m(A, b)$ is the space spanned by the vectors that one obtains by iteratively applying the matrix A to b . In particular, we set $\mathcal{K}_1(A, b) := \mathbb{R}b$ and $\mathcal{K}_0(A, b) := \{0\}$.

Furthermore, we define the *Krylov matrix* by

$$K_n(A, b) = [b, Ab, A^2b, \dots, A^{n-1}b].$$

In the following subsections we describe methods for solving the eigenvalue problem and linear equations for hermitian and non-hermitian matrices $A \in \mathbb{C}^{n \times n}$.

	$Ax = \lambda x$	$Ax = b$
$A \neq A^*$	Arnoldi	GMRES
$A = A^*$	Lanczos	CG

2.2.1 Arnoldi iteration

We first consider the Arnoldi iteration which is a method of computing a *Hessenberg reduction* $A = QHQ^*$, where H has the structure

$$H = \begin{bmatrix} \times & & \dots & \times \\ \times & \times & & \\ & \ddots & \ddots & \vdots \\ & & \times & \times \end{bmatrix}.$$

Arnoldi iteration is an important ingredient in many numerical methods (e.g. GMRES in Section 2.2.3). In the end of this section we will show how we can use Arnoldi iteration to approximate eigenvalues.

In principle a Hessenberg reduction can be easily achieved by Housholder transformations. The advantage of Arnoldi iteration is that it can be stopped part way, which yields a reduced Hessenberg reduction of the first m columns of A .

Hessenberg reduction can be written in the form

$$AQ = QH.$$

If we consider the first m columns of this matrix equation we obtain

$$AQ_m = Q_{m+1}\hat{H}_m, \quad (2.4)$$

where $Q_m \in \mathbb{C}^{n \times m}$ is the matrix with the first m columns of Q :

$$Q_m = [q_1, q_2, \dots, q_m],$$

and $\hat{H}_m \in \mathbb{C}^{(m+1) \times m}$ is the reduced matrix with the first m columns of H (i.e. the rows at the bottom containing only zeros are removed):

$$\hat{H}_m = \begin{bmatrix} h_{11} & & \dots & h_{1m} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{m(m-1)} & h_{mm} \\ & & 0 & h_{(m+1)m} \end{bmatrix}.$$

Considering the last column of Eqn (2.4) we obtain the following recurrence relation:

$$Aq_m = h_{1m}q_1 + h_{2m}q_2 + \dots + h_{mm}q_m + h_{(m+1)m}q_{m+1}. \quad (2.5)$$

Progressively solving this equation for q_{m+1} yields the Arnoldi iteration (see Algorithm 6).

Theorem 2.2.2. *The matrices Q_m generated by the Arnoldi iteration have the following properties*

1. Q_m are the reduced QR factors of the Krylov matrix $K_m(A, b)$:

$$K_m = Q_m R_m.$$

Therefore, the Arnoldi process offers a systematic construction of orthonormal bases for successive Krylov subspaces.

Data: Matrix A , vector b , number n

Initialisation: $q_1 = b/\|b\|$

```

for  $m = 1 \dots n$  do
   $v = Aq_m$ 
  for  $i = 1 \dots m$  do
     $h_{im} = q_i^* v$ 
     $v = v - h_{im} q_i$ 
  end
   $h_{(m+1)m} = \|v\|$ 
   $q_{m+1} = v/h_{(m+1)m}$ 
end

```

Algorithm 6: Arnoldi iteration

2. The Hessenberg matrices H_m are the projections of A onto the m -dimensional Krylov subspace $\mathcal{K}_m(A, b)$

$$H_m = Q_m^* A Q_m.$$

Since H_m is a projection of A the eigenvalues of H_m are related to those of A . The eigenvalues θ_j of the matrix H_m are called Arnoldi estimates or Ritz values.

3. The successive iterates are related by Eqn (2.4).

Arnoldi estimates are most accurate approximations of eigenvalues. Therefore, an obvious way of approximating the eigenvalues of $A \in \mathbb{C}^{n \times n}$ where n is large is to

1. Calculate the matrix H_m using Algorithm 6 for some $m \ll n$.
2. Calculate the eigenvalues of H_m ($\in \mathbb{C}^{m \times m}$) with a standard method (e.g. QR algorithm 4).

The Arnoldi process has interesting connection to polynomial approximation theory. This can help us to analyse in which way Arnoldi approximates eigenvalues, and which eigenvalues it will find (typically, it will detect only a few eigenvalues, since $m \ll n$). The Arnoldi process has interesting connection to polynomial approximation theory. This can help us to analyse in which way Arnoldi approximates eigenvalues, and which eigenvalues it will find (typically, it will detect only a few eigenvalues, since $m \ll n$).

If a vector is in a Krylov subspace of dimension $m + 1$, i.e. $x \in \mathcal{K}_{m+1}(A, b)$, it can be expressed as

$$x = c_0 b + c_1 A b + \dots c_m A^m b,$$

or if $q(z) = c_0 + c_1 z + \dots c_m z^m$ in polynomial form as

$$x = q(A)b.$$

Lets define P_m as the set of all monic (i.e the polynomials, where the highest coefficient $c_m = 1$) polynomials of degree m :

Lemma 2.2.3. *The characteristic polynomial of the Hessenberg matrix H_m is the unique solution of the minimization problem to find $p_m \in P_m$ such that*

$$\|p_m(A)b\|$$

is minimal.

This relates the characteristic polynomial of H_m , which acts as a 'pseudo minimal polynomial', to the characteristic polynomial of A which is the exact minimal polynomial.

2.2.2 Lanczos iteration

Arnoldi iteration is simplified if the matrix A is symmetric or hermitian. In this case, the matrix H will not only have Hessenberg structure but can be tridiagonal.

Therefore the recurrence equation (Eqn 2.5) turns into the simpler recurrence

$$Aq_m = h_{(m-1)m}q_{m-1} + h_{mm}q_m + h_{(m+1)m}q_{m+1}. \quad (2.6)$$

2.2.3 Generalized minimal residuals (GMRES)

The Arnoldi process can be used to approximately solve linear equations $Ax = b$. Lets assume $A \in \mathbb{R}^{n \times n}$ to be a regular matrix and consider the m -th Krylov subspaces $\mathcal{K}_m(A, b)$.

The idea of generalized minimal residuals (GMRES) is to choose each iterate $x^{(m)} \in \mathcal{K}_m(A, b)$ in order to minimize the norm of the residual $r^{(m)} = b - Ax^{(m)}$. The iterate $x^{(m)} \in \mathbb{R}^n$ can be written as $x^{(m)} = K_m c$ where $K_m \in \mathbb{R}^{n \times m}$ is the m -th Krylov matrix and $c \in \mathbb{R}^m$. Finding the minimal residuum is a least square problem:

$$c = \arg \min \|AK_m c - b\|, \quad x^{(m)} = K_m c.$$

This method is numerically unstable, but we can easily fix by taking Q_m from Section 2.2.1 as basis for the Krylov subspace $\mathcal{K}_m(A, b)$ instead of K_m . Arnoldi process iteratively creates these matrices Q_m (see Theorem 2.2.2). Therefore, we write $x^{(m)} = Q_m y$ and minimize

$$y = \arg \min \|AQ_m y - b\|, \quad x^{(m)} = Q_m y.$$

Applying Eqn (2.4) yields

$$y = \arg \min \|Q_{m+1} \hat{H}_m y - b\|, \quad x^{(m)} = Q_m y,$$

and multiplying the least square problem by Q_{m+1}^* gives

$$y = \arg \min \|\hat{H}_m y - Q_{m+1}^* b\|, \quad x^{(m)} = Q_m y.$$

Finally, we arrive at Algorithm 7 after inserting $\|b\|e_1 = Q_{m+1}^* b$. This can be easily verified, $q_1 = b/\|b\|$ (see Algorithm 6) and q_j are orthonormal vectors.

Data: Matrix A , vector b , number n
Initialisation: $k = 1$
while *convergence criterion not satisfied* **do**
 $Q_k, \hat{H}_k = \text{Arnoldi iteration}$ one step from Alg. 6
 $y = \arg \min \|\hat{H}_k y - \|b\|e_1\|$ least square problem
 $x^{(k)} = Q_k y$
 $k \leftarrow k + 1$
end

Algorithm 7: GMRES algorithm

2.2.4 Conjugate gradients (CG)

The conjugate gradient (CG) method is a very effective way to solve the linear equation $Ax = b$, if $A \in \mathbb{R}^{n \times n}$ is a symmetric and positive definite matrix.

Because of the positive definiteness of A , this equation can be reformulated as a quadratic optimisation problem. We define the mapping $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$ as

$$\Phi(x) := \frac{1}{2}x^T Ax - x^T b. \quad (2.7)$$

Then $\nabla \Phi(x) = Ax - b$, which shows that $\hat{x} \in \mathbb{R}^n$ is a critical point of Φ , if and only if $\hat{x}$ satisfies $A\hat{x} = b$. Moreover, the Hessian of Φ is simply the positive definite matrix A , showing that Φ is strictly convex and therefore $\hat{x} := A^{-1}b$ is its unique minimiser.

Furthermore, because A is positive definite, the mapping $\langle \cdot, \cdot \rangle_A: \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\langle x, y \rangle_A := x^T Ay = \langle x, Ay \rangle$$

defines a scalar product on $\mathbb{R}^n$. The corresponding norm

$$\|x\|_A := \sqrt{\langle x, x \rangle_A} = \sqrt{x^T Ax}$$

is called the *energy norm* induced by A . Rewriting the functional Φ , one can show that

$$\begin{aligned} \Phi(x) - \Phi(\hat{x}) &= \frac{1}{2}x^T Ax - x^T b - \frac{1}{2}\hat{x}^T A\hat{x} + \hat{x}^T b \\ &= \frac{1}{2}(x - \hat{x})^T A(x - \hat{x}) + \frac{1}{2}(x^T A\hat{x} + \hat{x}^T Ax) - \hat{x}^T A\hat{x} - x^T b + \hat{x}^T b \\ &= \frac{1}{2}(x - \hat{x})^T A(x - \hat{x}) \\ &= \frac{1}{2}\|x - \hat{x}\|_A^2 \end{aligned}$$

Thus, minimising Φ is equivalent to minimising the squared distance to the solution of the equation $Ax = b$ measured in the energy norm.

We construct a method to approximate $\hat{x}$ by successively minimizing the functional Φ . Let $x^{(k)}$ the actual approximation, then we find the next iterate by

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)}, \quad (2.8)$$

where $d^{(k)} \neq 0$ is the search direction and $\alpha^{(k)}$ is magnitude we 'move' into this direction.

For a given $d^{(k)}$ we determine α by minimizing along the direction $d^{(k)}$. First we define the mapping $\alpha \rightarrow F(\alpha)$ by

$$\begin{aligned} F(\alpha) &:= \Phi(x^{(k)} + \alpha d^{(k)}) \\ &= \Phi(x^{(k)}) + \alpha d^{(k)T} A x^{(k)} + \frac{1}{2} \alpha^2 d^{(k)T} A d^{(k)} - \alpha d^{(k)T} b. \end{aligned}$$

Then, $F(\alpha)$ is minimal if $F'(\alpha^{(k)}) = 0$, thus

$$\alpha^{(k)} = \frac{(b - A x^{(k)})^T d^{(k)}}{d^{(k)T} A d^{(k)}} =: \frac{r^{(k)T} d^{(k)}}{d^{(k)T} A d^{(k)}}, \quad (2.9)$$

where $r^{(k)} = b - A x^{(k)}$ is the residuum at the k -th iterate.

In the CG method, the direction $d^{(k+1)}$ is chosen to be a linear combination of the previous direction $d^{(k)}$ and the residual $r^{(k+1)}$ in such a way that $d^{(k+1)}$ and $d^{(k)}$ are orthogonal with respect to A

$$d^{(k+1)} := r^{(k+1)} + \beta_k d^{(k)},$$

with β_k such that

$$\langle d^{(k)}, d^{(k-1)} \rangle_A = d^{(k)T} A d^{(k-1)} = 0.$$

We calculate

$$0 = d^{(k+1)T} A d^{(k)} = r^{(k+1)T} A d^{(k)} + \beta_k d^{(k)T} A d^{(k)},$$

and therefore

$$\beta_k = -\frac{r^{(k+1)T} A d^{(k)}}{d^{(k)T} A d^{(k)}} = -\frac{\langle r^{(k+1)}, d^{(k)} \rangle_A}{\langle d^{(k)}, d^{(k)} \rangle_A}. \quad (2.10)$$

With Eqns (2.9) and (2.10) the CG iteration of Eqn 2.8 is well defined. We summarize:

Theorem 2.2.4. *Let $A \in \mathbb{R}^{n \times n}$ be symmetric and positive definite and let $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$ be as in (2.7). Let $x^{(0)} \in \mathbb{R}^n$ be arbitrary and consider the iteration (the CG method)*

$$x^{(k+1)} = x^{(k)} - \frac{\langle r^{(k)}, d^{(k)} \rangle}{\langle d^{(k)}, d^{(k)} \rangle_A} d^{(k)}$$

with $r^{(k)} := b - A x^{(k)}$, $d^{(0)} = r^{(0)}$, and

$$d^{(k)} = r^{(k)} - \frac{\langle r^{(k)}, d^{(k-1)} \rangle_A}{\langle d^{(k-1)}, d^{(k-1)} \rangle_A} d^{(k-1)},$$

where we assume that the iteration is stopped if $r^{(k)} = 0$.

If $r^{(k)} = 0$, then $x^{(k)} = \hat{x} = A^{-1}b$. Else, the Krylov space $\mathcal{K}_k(A, r^{(0)})$ has dimension k and $x^{(k)}$ minimises Φ on the affine space $x^{(0)} + \mathcal{K}_k(A, r^{(0)})$.

Data: a positive definite and symmetric matrix $A \in \mathbb{R}^{n \times n}$;
some $b \in \mathbb{R}^n$;
an initial guess x_{init} ;
Result: the solution $x^* \in \mathbb{R}^n$ of $Ax = b$;
Initialisation: set $x^{(0)} := x_{\text{init}}$, $r^{(0)} = b - Ax^{(0)}$, $d^{(0)} = r^{(0)}$, $k = 0$;
while *convergence criterion not satisfied* **do**

$$\alpha^{(k)} = \frac{\|r^{(k)}\|^2}{\langle d^{(k)}, d^{(k)} \rangle_A};$$

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)};$$

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} A d^{(k)};$$

$$\beta^{(k)} = \frac{\|r^{(k+1)}\|^2}{\|r^{(k)}\|^2};$$

$$d^{(k+1)} = r^{(k+1)} + \beta^{(k)} d^{(k)};$$

$k \leftarrow k + 1$;
end
define $\hat{x} := x^{(k)}$;

Algorithm 8: Conjugate gradient method.

Some reformulation yields the method described in Algorithm 8. The CG method described in Theorem 2.2.4 finds the solution $\hat{x}$ of the equation $Ax = b$ in at most n steps.

Theoretically, solving an equation with the conjugate gradient method is not very efficient, because it requires more operations than for instance a Cholesky or LDL decomposition. In addition, the convergence after n steps is purely theoretical, as it only holds when all the computations are performed in exact arithmetic without any rounding errors. In practise, however, it performs much better, because one can, and should, make use of the advantage of iterative methods: One need not stop only when the residual equals zero, but rather when the residual is sufficiently small.

One problem of the CG method is that the convergence speed depends heavily on the condition of the matrix A . In order to improve the performance, usually the method is not applied directly to the equation $Ax = b$, but rather to a transformed problem $M^{-1}Ax = M^{-1}b$, where the (easily invertible) matrix M is chosen in such a way that $M^{-1}A$ is much better conditioned than A . This approach is called *preconditioning*.

In theory, a slightly different approach is required, as the matrix $M^{-1}A$ will not be symmetric any more. To that end one can consider a Cholesky factorisation $M = LL^T$ of the positive definite and symmetric matrix M and then apply the CG method to the equation $L^{-1}AL^{-T}\hat{x} = L^{-1}b$ and solve $L^T x = \hat{x}$. It is possible to rewrite the resulting algorithm in a such a way that one only needs the original matrix M and never its factorisation.