Calcul scientifique pour les grands systèmes linéaires

Mi-Song Dupuy

Sorbonne Université
Master 1 Mathématiques et Applications
Parcours HPC
MU4MA053

2024 - 2025

Contents

Introduction

These notes present the classical numerical methods to solve linear systems, by direct (Chapter 2) and iterative methods (Chapters 3 and 4). A chapter is also devoted to a short introduction on eigenvalue problems (Chapter 5).

The focus in this course, is on the well-posedness, in the mathematical sense, of the algorithms and the convergence theory of the iterative methods. A classical and exhaustive text on numerical linear algebra is the book by G. Golub and C. Van Loan [GL13].

The present notes are of course inspired by other texts. Chapter 2 and Chapter 3 come from earlier notes by Xavier Claeys. The presentation of the Krylov subspace methods is heavily influenced by the book by J. Liesen and Z. Strakos [LS12]. Finally the presentation of the eigenvalue problems has been inspired by the classical text by Y. Saad [Saa11].

Although the notion of conditioning is present in these notes, preconditioning is not covered. The stability of algorithms which is crucial in the implementation of the methods is also absent. A good reference on that topic is the book by N. Higham [Hig02].

The interested reader that would like to dig deeper into the iterative methods for linear systems can also look into the other classical book by Y. Saad [Saa03]. Finally, let us mention the recent and compact monograph [CG22], where the basic methods for linear systems are covered as well as hindsight on modern methods, that are not covered in this course, like preconditioning or multigrid methods.

Chapter 1

Reminders on matrix analysis

In this chapter, we will review basic notions of matrix analysis. For a more advanced discussions, see the book by Horn and Johnson [HJ13]. The singular value decomposition will also be presented, especially its relationship with the best low-rank approximation.

1.1 Vector and matrix norms

Given an integer $n \geq 0$, we recall that a norm over $\mathbb{C}^n$ is an application $\|\cdot\|_* : \mathbb{C}^n \to \mathbb{R}_+$ satisfying: for all $v, w \in \mathbb{C}^n$ and all $\lambda \in \mathbb{C}$

- $\|v\|_* = 0 \;\Rightarrow\; v = 0$
- $\|v + w\|_* \leq \|v\|_* + \|w\|_*$
- $\|\lambda v\|_* = |\lambda| \, \|v\|_*$

Let us mention three classical norms. Given a vector $u = (u_j)_{j=1}^n \in \mathbb{C}^n$, we define

$$\|u\|_1 := \textstyle\sum_{j=1}^n |u_j|$$
$$\|u\|_2 := (\textstyle\sum_{j=1}^n |u_j|^2)^{1/2}$$
$$\|u\|_\infty := \sup_{j=1\ldots n} |u_j|$$

It can be easily verified that each of these three applications is indeed a norm. Besides, let us point that $\|\cdot\|_2$ is the norm naturally attached to the scalar product[1] $(u, v) \mapsto \overline{u}^T v = u^* v$ over $\mathbb{C}^n$. As such the 2-norm is invariant under unitary transformations $U \in \mathbb{C}^{n\times n}$: *i.e.* $\|Uv\|_2^2 = (Uv)^* Uv = v^* U^* U v = v^* v = \|v\|_2^2$.

Let us recall that these norms are equivalent as $\mathbb{C}^n$ is a finite dimensional space: we have $\|u\|_1 \leq \sqrt{n}\|u\|_2 \leq n\|u\|_\infty \leq n\|u\|_1$ for all $u \in \mathbb{C}^n$.

The choice of a vector norm $\|\cdot\|_*$ over $\mathbb{C}^n$ induces a norm over matrices $\mathbb{C}^{n\times n}$ defined by

$$\|\mathrm{A}\|_* := \sup_{u \in \mathbb{C}^n \setminus \{0\}} \frac{\|\mathrm{A}u\|_*}{\|u\|_*} \quad \text{for } \mathrm{A} \in \mathbb{C}^{n\times n}. \tag{1.1}$$

[1] Note that as opposed to the usual mathematical convention, here it is the physicists' convention that is chosen, where the left argument is complex-conjugated

A norm over $\mathbb{C}^{n\times n}$ taking the form above is said to be *induced*[2]. Induced norms possess the following elementary property.

Lemma 1.1. *If $\|\ \|_*$ is an induced norm over $\mathbb{C}^{n\times n}$ then $\|\mathrm{A}^k\|_* \leq \|\mathrm{A}\|_*^k$ for all $\mathrm{A}\in\mathbb{C}^{n\times n}, k\geq 0$.*

In the following, in an abuse of notation, we will denote $\|\cdot\|_1$ the matrix norm induced by $\|\cdot\|_1$. Similarly we will denote $\|\cdot\|_2$ and $\|\cdot\|_\infty$, the matrix norms induced by the vector norms $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

Remark 1.2. *There exist matrix norms that are not induced. Let $\mathrm{A}=(a_{j,k})\in\mathbb{C}^{n\times n}$, let $\mathrm{tr}(\mathrm{A}):=\sum_{j=1}^n a_{j,j}$ refer to its trace. In addition, we shall denote $\mathrm{A}^*:=(\overline{\mathrm{A}})^T$ its adjoint* i.e. *its hermitian transpose. Then the application $(\mathrm{A},\mathrm{B})\mapsto \mathrm{tr}(\mathrm{B}^*\mathrm{A})$ provides a scalar product over $\mathbb{C}^{n\times n}$. The norm associated to this scalar product, called the* Frobenius norm

$$\|\mathrm{A}\|_{\mathrm{F}}:=\sqrt{\mathrm{tr}(\mathrm{A}^*\mathrm{A})}=\sum_{j=1}^n\sum_{k=1}^n\|a_{j,k}\|^2$$

is not induced. Indeed if it was, one would necessarily have $\|\,\mathrm{id}\,\|_*=1$. However in the case of the Frobenius norm, a direct calculation shows that $\|\,\mathrm{id}\,\|_{\mathrm{F}}=\sqrt{n}$.

Some matrix norms have an explicit formula.

Proposition 1.3. *For any matrix $\mathrm{A}=(a_{j,k})\in\mathbb{C}^{n\times n}$, we have*

$$\begin{aligned}\|\mathrm{A}\|_1 &= \sup_{k=1\dots n}\sum_{j=1}^n|a_{j,k}|\\ \|\mathrm{A}\|_\infty &= \sup_{j=1\dots n}\sum_{k=1}^n|a_{j,k}|\end{aligned}$$

Proof. We start by proving the identity for $\|\mathrm{A}\|_1$. Given a vector $u=(u_k)_{k=1}^n\in\mathbb{C}^n\setminus\{0\}$, applying a simple triangular inequality yields

$$\begin{aligned}\|\mathrm{A}u\|_1=\sum_{j=1}^n\Big|\sum_{k=1}^n a_{j,k}u_k\Big|&\leq\sum_{j=1}^n\sum_{k=1}^n|a_{j,k}|\,|u_k|\\ &\leq\sum_{k=1}^n|u_k|\big(\sum_{j=1}^n|a_{j,k}|\,\big)\leq\|u\|_1\sup_{k=1\dots n}\sum_{j=1}^n|a_{j,k}|\end{aligned}$$

Since this holds for all $u\in\mathbb{C}^n\setminus\{0\}$, dividing the last inequality by $\|u\|_1$ and taking the supremum with respect to u, we obtain that $\|\mathrm{A}\|_1\leq\sup_{k=1\dots n}\sum_{j=1}^n|a_{j,k}|$. To conclude we can construct a $u_\star\in\mathbb{C}^n$ such that $\|\mathrm{A}u_\star\|_1/\|u_\star\|=\sup_{k=1\dots n}\sum_{j=1}^n|a_{j,k}|$. Pick $k_\star\in\{1\dots n\}$ such that $\sup_{k=1\dots n}\sum_{j=1}^n|a_{j,k}|=\sum_{j=1}^n|a_{j,k_\star}|$. It suffices then to define $u_\star=(u_k)$ by $u_k=0$ if $k\neq k_\star$ and $u_{k_\star}=1$.

Let us now examine the case of $\|\mathrm{A}\|_\infty$. Similarly to what precedes, for an arbitrary $u\in\mathbb{C}^n$ we have

$$\|\mathrm{A}u\|_\infty=\sup_{j=1\dots n}\Big\|\sum_{k=1}^n a_{j,k}u_k\Big\|\leq\sup_{j=1\dots n}\sum_{k=1}^n|a_{j,k}|\,\|u\|_\infty.$$

[2]It is also common to call induced norms, *subordinated norms* or *operator norms*

Again dividing by $\|u\|_\infty$ and taking the supremum of the left hand side with respect to u, we obtain $\|\mathrm{A}\|_\infty \leq \sup_{j=1\dots n} \sum_{k=1}^n |a_{j,k}|$. Let us prove that this upper bound is reached. Choose $j_\star$ such that $\sum_{k=1}^n |a_{j_\star,k}| = \sup_{j=1\dots n} \sum_{k=1}^n |a_{j,k}|$. We can take $u_\star = (u_k) \in \mathbb{C}^n$ with entries given by $u_k = \overline{a}_{j_\star,k}/|a_{j_\star,k}|$, and we then obtain $\|\mathrm{A}u\|_\infty/\|u\|_\infty = \sup_{j=1\dots n} \sum_{k=1}^n |a_{j,k}|$. □

1.1.1 Eigenvalues, eigenvectors, spectral radius

The set of eigenvalues is a fundamental concept for a square matrix

Definition 1.4. *Let $A \in \mathbb{C}^{n\times n}$. If $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$ a nonzero vector satisfy the equation*

$$Ax = \lambda x,$$

then λ is called an eigenvalue *of A and x is called an* eigenvector *of A associated with the eigenvalue λ.*

The eigenvalues of a matrix are thus associated to the roots of its characteristic polynomial $p(X) = \det(X\,\mathrm{id} - A)$.

We say that a matrix A is *diagonalisable* if there is an invertible matrix $P \in \mathbb{C}^{n\times n}$ and a diagonal matrix $D \in \mathbb{C}^{n\times n}$ such that $A = PDP^{-1}$. By the fundamental theorem of algebra, the characteristic polynomial can be factorised in $\mathbb{C}$, thus any matrix has at least one eigenvalue. A matrix may not be diagonalisable, although its characteristic polynomial can be factorised. A typical example is the following nilpotent matrix $N = (n_{ij})_{1\leq i,j\leq n}$ where $n_{i,i+1} = 1$ and the other entries are equal to 0. Its characteristic polynomial is X^n, thus 0 is an eigenvalue. However, there is only one eigenvector associated to 0, which is the first canonical vector $e_1 = [1, 0, \dots, 0]^T$.

Remark 1.5 (Geometric and algebraic multiplicities)**.** *An eigenvalue λ of a matrix A can be associated to multiple eigenvectors. We call the* geometric multiplicity *the dimension of the space associated to* $\mathrm{Ker}(\lambda\,\mathrm{id} - A)$*, also called the* eigenspace associated to λ.

Since λ is also the root of the characteristic polynomial, another notion of multiplicity can be defined. We call the algebraic multiplicity *the multiplicity of λ as a root of the characteristic polynomial of A.*

The algebraic multiplicity is always larger than the geometric multiplicity. Indeed, since the characteristic polynomial is invariant under similarity transform, i.e. *transformation of a matrix A into $P^{-1}AP$ for invertible matrices P, then by choosing P to be eigenvectors of A and completing the basis if needed, one can quickly check that the algebraic multiplicity is always at least as large as the geometric multiplicity. The example of the nilpotent matrix exhibited earlier shows that these two notions can differ for a given matrix. The relationship of between both have been exhaustively studied from the early days of linear algebra, and the reader is referred to [HJ13, Chapter 1] for a more thorough exposition.*

Definition 1.6. *The spectrum of $A \in \mathbb{C}^{n\times n}$ is the set of all $\lambda \in \mathbb{C}$ that are eigenvalues of A. We denote this set by $\sigma(\mathrm{A})$.*

The largest eigenvalue plays an important role in the convergence of the stationary iterative methods that will be covered in Chapter 3.

Definition 1.7 (Spectral radius)**.** *For $A \in \mathbb{C}^{n\times n}$, the spectral radius of A, denoted $\varrho(A)$ is defined by $\varrho(\mathrm{A}) = \sup\{|\lambda|,\ \lambda \in \sigma(\mathrm{A})\}$, where $\sigma(\mathrm{A})$ is the set of eigenvalues of A.*

Remark 1.8. *Note that the spectral radius is* not *a matrix norm. Indeed the nilpotent matrix* $N = (N_{ij})$ *such that* $n_{ij} = 0$ *for* $i \geq j$ *has eigenvalues 0 but the matrix is nonzero whenever* $n_{ij} \neq 0$ *for some* $i < j$*.*

For any induced matrix norm $\| \ \|_*$ *over* $\mathbb{C}^{n\times n}$ *we have* $\varrho(\mathrm{A}) \leq \|\mathrm{A}\|_*$*, for any* $\mathrm{A} \in \mathbb{C}^{n\times n}$*. The converse cannot be true, however for any matrix* $A \in \mathbb{C}^{n\times n}$*, for any* $\varepsilon > 0$*, there is a vector norm* $\| \cdot \|_*$ *such that the corresponding matrix norm satisfies* $\|A\|_* \leq \varrho(A) + \varepsilon$ *(see Proposition 3.2).*

For hermitian matrices, *i.e.* for matrices A such that $A = A^*$, their eigenvalues have an additional variational characterisation, known as the *Courant-Fischer principle* (see Proposition 1.12).

1.2 Condition number

Although to any norm is attached a condition number, most of the time one considers the so-called "quadratic" condition number attached to the norm $\| \cdot \|_2$. For a matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$, it is defined by

$$\mathrm{cond}_2(\mathrm{A}) := \|\mathrm{A}\|_2\|\mathrm{A}^{-1}\|_2. \tag{1.2}$$

This quantity is only meaningful for an invertible matrix i.e. whose kernel is trivial $\mathrm{Ker}(\mathrm{A}) = \{0\}$. The condition number is systematically greater than 1. Indeed $1 = \| \mathrm{id} \|_2 = \|\mathrm{A}\cdot\mathrm{A}^{-1}\|_2 \leq \|\mathrm{A}\|_2\|\mathrm{A}^{-1}\|_2 = \mathrm{cond}_2(\mathrm{A})$.

The next proposition shows that the condition number quantifies the sensibility of a linear system with respect to perturbations.

Theorem 1.9. *Consider* $\mathrm{A} \in \mathbb{C}^{n\times n}$ *invertible and* $b, \delta b \in \mathbb{C}^n$ *with* $b \neq 0$*. Let* $x, \delta x \in \mathbb{C}^n$ *refer to vectors such that* $\mathrm{A}x = b$ *and* $\mathrm{A}(x + \delta x) = b + \delta b$*. Then we have*

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \mathrm{cond}_2(\mathrm{A})\frac{\|\delta b\|_2}{\|b\|_2}.$$

Proof. We have $\mathrm{A}(x + \delta x) = \mathrm{A}x + \mathrm{A}\delta x = b + \delta b$ hence $\mathrm{A}\delta x = \delta b$ and thus $\delta x = \mathrm{A}^{-1}\delta b$.We then deduce $\|b\|_2 \leq \|\mathrm{A}\|_2\|x\|_2$ on the one hand, and $\|\delta x\|_2 \leq \|\mathrm{A}^{-1}\|_2\|\delta b\|_2$ on the other hand. Multiplying the last two inequalities, and dividing by $\|x\|_2\|b\|_2$, we finally obtain the required inequality. □

1.3 Matrix factorisations

We will review here important matrix factorisations, for square and nonsquare matrices.

1.3.1 Schur decomposition

Proposition 1.10. *For any square matrix* $\mathrm{A} \in \mathbb{C}^{n\times n}$*, there exists a unitary matrix* $Q \in \mathbb{C}^{n\times n}$ *such that* $Q^*\mathrm{A}Q$ *is upper triangular. Moreover the diagonal elements of* $Q^*\mathrm{A}Q$ *are eigenvalues of* A*.*

In other words, any matrix in $\mathbb{C}^{n\times n}$ is unitarily equivalent to a triangular matrix. Conceptually, this factorisation theorem is a consequence of the fundamental theorem in algebra, which asserts that any polynomial in $\mathbb{C}[X]$ can be factorised.

Proof. Let $\mathrm{A} \in \mathbb{C}^{n\times n}$ be a square matrix.

Assuming that there exists a unitary matrix Q such that $T = Q^*\mathrm{A}Q$ is upper triangular, since A and T share the same characteristic polynomial, the diagonal elements of T are the eigenvalues of A.

We now prove by induction over the dimension n that there is a unitary matrix Q such that $Q^*\mathrm{A}Q$ is upper triangular. It trivially holds for $n = 1$. Suppose that the result holds for $n-1$ and let us prove that it still holds for n. There exists $\lambda \in \mathbb{C}$ and $q_1 \in \mathbb{C}^n$, with $\|q_1\|_2 = 1$ such that $\mathrm{A}q_1 = \lambda q_1$. We can find linearly independent vectors $q_2, \dots, q_n$ such that $(q_1, \dots, q_n)$ forms an orthonormal basis of $\mathbb{C}^n$. For each $k = 2 \dots n$, there are coefficients α_k and $\beta_{j,k} \in \mathbb{C}$, $j = 2 \dots n$ such that

$$\mathrm{A}q_k = \alpha_k q_1 + \sum_{j=2}^{n} \beta_{j,k} q_j \quad k = 2 \dots n \tag{1.3}$$

Note $\mathrm{B} = (\mathrm{B}_{j,k}) \in \mathbb{C}^{(n-1)\times(n-1)}$ defined by the coefficients $\mathrm{B}_{j,k} = \beta_{j+1,k+1}$, as well as the vector $\boldsymbol{\alpha} := (\alpha_2, \dots, \alpha_n)^T \in \mathbb{C}^{n-1}$. Let us also define $\widetilde{Q} := [q_1, \dots, q_n]$ the matrix associated to basis we have just defined. Writing the matrix representation of (1.3) then leads to

$$\widetilde{Q}^*\mathrm{A}\widetilde{Q} = \begin{bmatrix} \lambda & \boldsymbol{\alpha}^T \\ 0 & \mathrm{B} \end{bmatrix}. \tag{1.4}$$

By the induction hypothesis, there exists $Q_B \in \mathbb{C}^{(n-1)\times(n-1)}$ invertible such that $T_B := Q_B^*\mathrm{B}Q_B \in \mathbb{C}^{(n-1)\times(n-1)}$ is upper triangular. Setting $\widetilde{Q}_B = \begin{bmatrix} 1 & \\ & Q_B \end{bmatrix} \in \mathbb{C}^{n\times n}$, we then obtain

$$\left(\widetilde{Q}\widetilde{Q}_B\right)^* A \widetilde{Q}\widetilde{Q}_B = \begin{bmatrix} 1 & \\ & Q_B^* \end{bmatrix} \begin{bmatrix} \lambda & \boldsymbol{\alpha}^T \\ 0 & \mathrm{B} \end{bmatrix} \begin{bmatrix} 1 & \\ & Q_B \end{bmatrix} = \begin{bmatrix} \lambda & \boldsymbol{\alpha}^T Q_B \\ & T_B \end{bmatrix}.$$

Note that $\widetilde{Q}\widetilde{Q}_B$ is unitary. As the matrix on the right hand side above is upper triangular, it concludes the induction, and hence the proof. □

The Schur decomposition is a versatile tool in linear algebra as it applies to any matrix in $\mathbb{C}^{n\times n}$.

1.3.2 Eigenvalue decomposition

Theorem 1.11 (Eigenvalue decomposition)**.** *Let A be a hermitian matrix. Then there exist a unitary matrix Q and a diagonal matrix Λ with real entries such that $A = Q\Lambda Q^*$.*

Any hermitian matrix is unitarily equivalent to a diagonal matrix with *real* entries.

Proof. Let $Q, T \in \mathbb{C}^{n\times n}$ be a Schur decomposition of A, *i.e.* such that Q is unitary, T is upper triangular and $A = QTQ^*$. Then since A is hermitian, $A = A^*$ thus $QTQ^* = QT^*Q^*$, hence $T = T^*$. Since T is upper triangular, that means that T is diagonal. We simply have to prove that the diagonal elements of $T = \mathrm{diag}(t_{11}, \dots, t_{nn})$ are real. Let $Q = \begin{bmatrix} q_1, \dots, q_n \end{bmatrix}$. Then $Aq_k = Q_k T e_k = t_{kk} q_k$ thus q_k is an eigenvector of A with eigenvalue t_{kk}. Now $\langle q_k, Aq_k \rangle = t_{kk}$ and $\langle q_k, Aq_k \rangle = \langle A^* q_k, q_k \rangle = \langle Aq_k, q_k \rangle = \langle q_k, Aq_k \rangle^*$, thus t_{kk} is real. □

From the eigenvalue decomposition, by unitary invariance of the 2-norm, we have that

$$\|A\|_2 = \|Q\Lambda Q^*\|_2 = \sup_{x\in\mathbb{C}^n} \frac{\|Q\Lambda Q^* x\|_2}{\|x\|_2} = \sup_{x\in\mathbb{C}^n} \frac{\|\Lambda Q^* x\|_2}{\|Q^* x\|_2} \leq \sup_{x\in\mathbb{C}^n} \Big(\frac{\sum_{i=1}^n |\lambda_i x_i|^2}{\sum_{i=1}^n |x_i|^2}\Big) \leq |\lambda_{\max}|,$$

where $\lambda_{\max}$ is the largest eigenvalue of A in absolute value.

This means that the condition number of a hermitian matrix A is given by

$$\mathrm{cond}_2(\mathrm{A}) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}$$

where $\lambda_{\max}$, respectively $\lambda_{\min}$, is the largest eigenvalue, respectively the smallest eigenvalue, in absolute value of A.

Proposition 1.12 (Courant-Fischer principle)**.** *Let $A \in \mathbb{C}^{n\times n}$ be a hermitian matrix. Let $\lambda_1 \leq \cdots \leq \lambda_n$ be its eigenvalues. Then for each $1 \leq k \leq n$, we have*

$$\lambda_k = \min_{\substack{S\subset\mathbb{C}^n \\ \dim S = k}} \max_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2} = \max_{\substack{S\subset\mathbb{C}^n \\ \dim S = n+1-k}} \min_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2}.$$

Proof. Let $(q_j)_{1\leq j\leq n}$ be the eigenvectors associated to $(\lambda_j)_{1\leq j\leq n}$. Let $1 \leq k \leq n$ and $S \subset \mathbb{C}^n$ be a subspace of dimension k. Since $\dim \mathrm{Span}(q_k, \ldots, q_n) = n - k + 1$, by dimension counting, we have that $S \cap \mathrm{Span}(q_k, \ldots, q_n)$ is a subspace of dimension at least 1. Let $x \in S \cap \mathrm{Span}(q_k, \ldots, q_n)$, $x \neq 0$. Then

$$\sup_{y\in S} \frac{y^*Ay}{\|y\|_2^2} \geq \frac{x^*Ax}{\|x\|_2^2} \geq \frac{\sum_{j=k}^n \lambda_j |x_j|^2}{\sum_{j=k}^n |x_j|^2} \geq \lambda_k.$$

Thus we have

$$\inf_{\substack{S\subset\mathbb{C}^n \\ \dim S = k}} \sup_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2} \geq \lambda_k.$$

Taking $S_* = \mathrm{Span}(q_1, \ldots, q_k)$, we have that $\sup_{x\in S_*} \frac{x^*Ax}{\|x\|_2^2} = \lambda_k$, thus the infinimum and the supremum are attained and

$$\lambda_k = \min_{\substack{S\subset\mathbb{C}^n \\ \dim S = k}} \max_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2}.$$

For the second equality, the previous result is applied to the matrix $-A$, noticing that $-\lambda_k$ is the $n - k + 1$ smallest eigenvalue of $-A$:

$$\begin{aligned} -\lambda_k &= \min_{\substack{S\subset\mathbb{C}^n \\ \dim S = n-k+1}} \max_{x\in S, x\neq 0} -\frac{x^*Ax}{\|x\|_2^2} \\ &= \min_{\substack{S\subset\mathbb{C}^n \\ \dim S = n-k+1}} \Big(-\min_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2}\Big) \\ &= -\max_{\substack{S\subset\mathbb{C}^n \\ \dim S = n-k+1}} \min_{x\in S, x\neq 0} \frac{x^*Ax}{\|x\|_2^2}, \end{aligned}$$

thus the result. □

For the smallest eigenvalue, this criterion simplifies to

$$\lambda_1 = \min_{x\in\mathbb{C}^n, x\neq 0} \frac{x^*Ax}{\|x\|_2^2},$$

and likewise for the largest eigenvalue

$$\lambda_n = \max_{x\in\mathbb{C}^n, x\neq 0} \frac{x^*Ax}{\|x\|_2^2}.$$

1.3.3 The QR factorisation

The QR factorisation is an important example of a matrix factorisation that can be used to factor nonsquare matrices.

Theorem 1.13. *Let $B \in \mathbb{C}^{m\times n}$. Then there exist $Q \in \mathbb{C}^{m\times m}$ unitary (*i.e. $Q^*Q = QQ^* = \mathrm{id}_m$*) and $R \in \mathbb{C}^{m\times n}$ upper-triangular such that $B = QR$. Such a factorisation is called a QR factorisation of B.*

Proof. The theorem is proved by induction on the dimension n.

For $n = 1$, $B \in \mathbb{C}^{m\times 1}$, so we can pick $Q = [\frac{B}{\|B\|} \quad Q^\perp]$ where $Q^\perp$ has columns which are an orthonormal basis of $\{B\}^\perp$. Then $B = Q\begin{bmatrix}\|B\| \\ 0\end{bmatrix}$.

Suppose that for any $G \in \mathbb{C}^{m\times n}$, we can write its QR factorisation and let $B \in \mathbb{C}^{m\times(n+1)}$. Write $B = [B_1 \quad v]$, with $B_1 \in \mathbb{C}^{m\times n}$ and $v \in \mathbb{C}^m$. By the induction hypothesis, we have $B_1 = Q_1R_1$ where $Q_1 \in \mathbb{C}^{m\times m}$ is unitary and $R_1 \in \mathbb{C}^{m\times n}$ is upper-triangular. Let

$$w = Q_1^*v \in \mathbb{C}^m, \quad \text{and} \quad w_{n+1:m} = \begin{bmatrix} w_{n+1} \\ \vdots \\ w_m \end{bmatrix} = Q_2R_2,$$

where $w_{n+1:m} = Q_2R_2$ is a QR factorisation of $w_{n+1:m}$. Then setting $Q = Q_1\begin{bmatrix}\mathrm{id}_n & 0 \\ 0 & Q_2\end{bmatrix}$ and $R = \left[\begin{array}{c|c} R_1 & \begin{matrix} w_{1:n} \\ R_2 \end{matrix}\end{array}\right]$, we check that

$$QR = Q_1\begin{bmatrix}(R_1)_{1:n} & w_{1:n} \\ 0 & Q_2R_2\end{bmatrix} = Q_1\left[\begin{array}{c|c} R_1 & \begin{matrix} w_{1:n} \\ w_{n+1:m}\end{matrix}\end{array}\right] = \begin{bmatrix} Q_1R_1 & Q_1w\end{bmatrix} = B.$$

□

In general, the QR factorisation of a matrix B is not unique. The QR factorisation is a useful tool to characterise the solution to a least-square problem.

Proposition 1.14. *Let $B \in \mathbb{C}^{m\times n}$, with $m \geq n$, be a full-rank matrix and $B = QR$ a QR factorisation of B. Let $Q_1 \in \mathbb{C}^{m\times n}$ and $Q_2 \in \mathbb{C}^{m\times(m-n)}$ such that $Q = \begin{bmatrix}Q_1 & Q_2\end{bmatrix}$. Let $R_1 \in \mathbb{C}^{n\times n}$ be such that $R = \begin{bmatrix}R_1 \\ 0\end{bmatrix}$. Then R_1 is an invertible matrix and for any $f \in \mathbb{C}^m$ then*

$$\min_{x\in\mathbb{C}^n} \|f - Bx\| = \|f - Bx_*\| = \|(Q^*f)_{n+1:m}\|,$$

where $x_* = R_1^{-1}(Q^*f)_{1:n}$ *and* $(Q^*f)_{1:n} \in \mathbb{C}^n$, $(Q^*f)_{n+1:m} \in \mathbb{C}^{m-n}$ *are such that* $Q^*f = \begin{bmatrix} (Q^*b)_{1:n} \\ (Q^*f)_{n+1:m} \end{bmatrix}$.

Proof. We have $H = Q_1R_1$, where $Q_1 \in \mathbb{C}^{m\times n}$ has independent columns and $R_1 \in \mathbb{C}^{n\times n}$. Since H is full-rank, it means that R_1 is invertible.

For the least-square problem, we have

$$\min_{x\in\mathbb{C}^n} \|f - Bx\| = \min_{x\in\mathbb{C}^n} \|f - QRx\| = \min_{x\in\mathbb{C}^n} \left\|Q^*f - \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x\right\| = \min_{x\in\mathbb{C}^n} \left\|Q^*f - \begin{bmatrix} R_1x \\ 0 \end{bmatrix}\right\|.$$

We directly have that $\min_{x\in\mathbb{C}^n} \left\|Q^*f - \begin{bmatrix} R_1x \\ 0 \end{bmatrix}\right\| \geq \|(Q^*f)_{n+1:m}\|$ and the minimum is attained for $x_* = R_1^{-1}(Q^*f)_{1:n}$. □

Remark 1.15. *With the notation of Proposition 1.14, we have* $B = Q_1R_1$*, which is called a* thin QR factorisation *of* B*. For the QR factorisation,* R *has the same shape as* B *but for the thin QR factorisation, it is* Q *which has the same shape as* B*. If* B *is full-rank, then one can prove that the thin QR factorisation is unique.*

Computing the QR factorisation

An algorithm to compute the QR factorisation is the Gram-Schmidt algorithm that is given in Algorithm 1.1. This algorithm is called the *classical Gram-Schmidt* algorithm.

Algorithm 1.1 Classical Gram-Schmidt

Input: $A = [a_1, \dots, a_n] \in \mathbb{C}^{m\times n}$ full-rank with $m \geq n$
Output: $Q = [q_1, \dots, q_m] \in \mathbb{C}^{m\times n}$ partial isometry, $R = [r_1, \dots, r_n] \in \mathbb{C}^{n\times n}$ upper triangular with $A = QR$

```
q_1 = a_1/‖a_1‖, R_11 = ‖a_1‖
for k = 2, ..., n do
    for ℓ = 1, ..., k − 1 do
        R_{ℓ,k} = ⟨q_ℓ, a_k⟩
    end for
    z = a_k − Σ_{j=1}^{k−1} R_{j,k} q_j
    q_k = z/‖z‖
end for
```

It is straightforward to notice that at each step of the algorithm, we have $A = QR$ and at the end, we retrieve the (thin) QR factorisation of the matrix A. Its complexity is equal to $mn^2 + m$ (if only the multiplications are considered).

Unfortunately, the classical Gram-Schmidt algorithm is not a numerically stable algorithm to compute accurate QR factorisations. However a simple modification of the previous algorithm enables to stabilise efficiently the Gram-Schmidt process. This algorithm is called the *Modified Gram-Schmidt algorithm* (Alg. 1.2).

Proposition 1.16. *The modified Gram Schmidt algorithm 1.2 is mathematically equivalent to the classical Gram-Schmidt algorithm 1.1.*

Algorithm 1.2 Modified Gram-Schmidt

Input: $A = [a_1, \dots, a_n] \in \mathbb{C}^{m\times n}$ full-rank with $m \geq n$
Output: $Q = [q_1, \dots, q_m] \in \mathbb{C}^{m\times n}$ partial isometry, $R = [r_1, \dots, r_n] \in \mathbb{C}^{n\times n}$ upper triangular with $A = QR$

```
for k = 1, ..., n do
    v_k^(1) = a_k
end for
for k = 1, ..., n do
    R_kk = ||v_k^(k)||, q_k = v_k^(k) / ||v_k^(k)||
    for ℓ = k+1, ..., n do
        R_{k,ℓ} = ⟨q_k, v_ℓ^(k)⟩
        v_ℓ^(k+1) = v_ℓ^(k+1) − R_{k,ℓ} q_k        ▷ can be done in place
    end for
end for
```

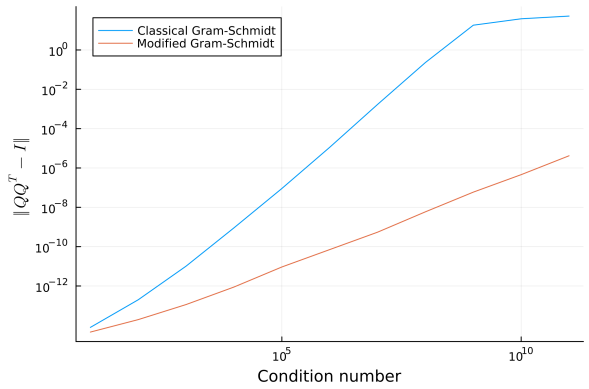


Figure 1.1: $\|QQ^T - I\|$ with respect to the conditioning of the matrix $A \in \mathbb{R}^{m\times n}$. The conditioning of a nonsquare matrix is defined here as $\frac{\sigma_1}{\sigma_n}$ where σ_1 (resp. σ_n) is the largest (resp. smallest) singular value.

Proof. We will show that for any $1 \leq k \leq n$, we have $v_k^{(k)} = a_k - \sum_{j=1}^{k-1} \langle q_j, a_k \rangle q_j$. Using the notation in Algorithm 1.2, given $1 \leq \ell \leq n$, we can prove by induction on $k \leq \ell$ that $v_\ell^{(k)} \perp q_j$ for any $1 \leq j \leq k-1$.

This means that for any $k \leq \ell - 1$, $R_{k,\ell} = \langle q_k, v_\ell^{(k)} \rangle = \langle q_k, v_\ell^{(k)} + \sum_{j=1}^{k-1} R_{j,\ell} q_j \rangle = \langle q_k, a_\ell \rangle$.

Hence, we have that $v_k^{(k)} = a_k - \sum_{j=1}^{k-1} \langle q_j, a_k \rangle q_j$, thus Algorithm 1.2 is equivalent to Algorithm 1.1. □

The modified Gram-Schmidt algorithm has the same computational complexity as the classical Gram-Schmidt algorithm, however it has a better numerical stability as illustrated in Figure 1.1.

1.3.4 Singular value decomposition

Another important matrix factorisation for nonsquare matrices is the singular value decomposition (SVD).

Theorem 1.17 (Singular value decomposition)**.** *Let $A \in \mathbb{C}^{m\times n}$. Then there exist two unitary matrices $U \in \mathbb{C}^{m\times m}, V \in \mathbb{C}^{n\times n}$ and a diagonal matrix $\Sigma = \mathrm{Diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{C}^{m\times n}$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$ such that $A = U\Sigma V^*$.*

The positive scalars $(\sigma_i)_{1\leq i\leq r}$ are called the singular values *of A. The vectors $(u_i)_{1\leq i\leq m}$, resp. $(v_i)_{1\leq i\leq n}$ are called the* left singular vectors*, resp. the* right *singular vectors.*

The SVD of A can be derived from the eigenvalue decomposition of the matrices AA^* and AA^*. Indeed, if $A = U\Sigma V^*$ is the SVD of A, then $A^* = V\Sigma U^*$ so using that U and V are unitary matrices, we have

$$AA^* = U\Sigma\Sigma^*U^* = U \begin{bmatrix} \sigma_1^2 & & & & \\ & \ddots & & & \\ & & \sigma_r^2 & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} U^*, \quad A^*A = V\Sigma^*\Sigma V^* = V \begin{bmatrix} \sigma_1^2 & & & & \\ & \ddots & & & \\ & & \sigma_r^2 & & \\ & & & 0 & \\ & & & & \ddots \end{bmatrix} V^*.$$

The singular values of A are simply the eigenvalues of the matrices AA^* and A^*A and the unitary matrices U and V the corresponding orthonormal eigenvectors.

Using the relationship between the singular values of A and the eigenvalues of A^*A, we can give a variational characterisation of the singular values.

Proposition 1.18. *Let $\mathrm{A} \in \mathbb{C}^{n\times n}$ and $\sigma_1 \geq \cdots \geq \sigma_r > 0$ its singular values. Then for each $1 \leq k \leq r$, we have*

$$\sigma_k = \max_{\substack{S\subset\mathbb{C}^n \\ \dim S=k}} \min_{x\in S} \frac{\|Ax\|_2}{\|x\|_2}. \tag{1.5}$$

Proof. This directly follows from the Courant-Fischer principle (Proposition 1.12). Noticing that the eigenvalues and the singular values are labelled in an opposite order, and that σ_k^2 is an eigenvalue of A^*A, we have that

$$\sigma_k^2 = \max_{\substack{S\subset\mathbb{C}^n \\ \dim S=k}} \min_{x\in S, x\neq 0} \frac{x^*A^*Ax}{\|x\|_2^2} = \max_{\substack{S\subset\mathbb{C}^n \\ \dim S=k}} \min_{x\in S, x\neq 0} \frac{\|Ax\|^2}{\|x\|_2^2}.$$

□

Remark 1.19. *From the proposition above, we deduce that the largest singular value σ_1 of a matrix A is equal to the induced 2-norm of the matrix A.*

The SVD enables to write any matrix as a sum of rank 1 matrices.

Proposition 1.20. *Let $A \in \mathbb{C}^{m\times n}$. There are orthonormal vectors $(u_i)_{1\leq i\leq r} \in \mathbb{C}^m$, $(v_j)_{1\leq j\leq r} \in \mathbb{C}^n$ and $(\sigma_i)_{1\leq i\leq r} \in (0,\infty)$ such that*

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^*.$$

The number of singular values hence gives the rank of the matrix.

Proof. It is a consequence of the SVD. Let (U, Σ, V) be an SVD of A, with $\Sigma = \mathrm{Diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{C}^{m\times n}$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$. Decompose U and V as

$$U = \begin{bmatrix} U_r & \widetilde{U}_r \end{bmatrix}, \quad V = \begin{bmatrix} V_r & \widetilde{V}_r \end{bmatrix},$$

where $U_r \in \mathbb{C}^{m\times r}, \widetilde{U}_r \in \mathbb{C}^{m\times(m-r)}$, $V_r \in \mathbb{C}^{n\times r}$ and $\widetilde{V}_r \in \mathbb{C}^{n\times(n-r)}$. Let $\Sigma_r = \mathrm{Diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{C}^{r\times r}$. Then we have

$$A = U\Sigma V^* = \begin{bmatrix} U_r & \widetilde{U}_r \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_r^* \\ \widetilde{V}_r^* \end{bmatrix} = U_r \Sigma_r V_r^*.$$

The proof follows by taking $\begin{bmatrix} u_1, \dots, u_r \end{bmatrix} = U_r$ and $\begin{bmatrix} v_1, \dots, v_r \end{bmatrix} = V_r$. □

Remark 1.21. *The SVD contains redundant information on the entries in the matrix, hence it is often enough to consider the* thin SVD (U_r, Σ_r, V_r) *such that* $U_r \in \mathbb{C}^{m\times r}$, $\Sigma_r = \mathrm{Diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{C}^{r\times r}$, $\sigma_1 \geq \cdots \geq \sigma_r > 0$, $V_r \in \mathbb{C}^{n\times r}$ *and* $A = U_r \Sigma_r V_r^*$.

From the SVD, it is possible to read the information on the range and the kernel of the matrix.

Proposition 1.22. *Let* (U, Σ, V) *be a SVD of* A, *where* $\Sigma = \mathrm{Diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{C}^{m\times n}$ *with* $\sigma_1 \geq \cdots \geq \sigma_r > 0$. *Let* $\begin{bmatrix} u_1, \dots, u_m \end{bmatrix} = U$ *and* $\begin{bmatrix} v_1, \dots, v_n \end{bmatrix} = V$. *Then* $\mathrm{Ker}(A) = \mathrm{Span}(v_{r+1}, \dots, v_n)$ *and* $\mathrm{Ran}(A) = \mathrm{Span}(u_1, \dots, u_r)$.

Proof. From Proposition 1.20, we directly have that $\mathrm{Ran}(A) = \mathrm{Span}(u_1, \dots, u_r)$. Since $(u_i)_{1\leq i\leq r}$ is a free family, we have that $Af = 0$ if and only if $\langle f, v_k \rangle = 0$ for all $r+1 \leq k \leq n$, thus $f \in \mathrm{Span}(v_{r+1}, \dots, v_n)$. □

Finally, an important property of the SVD is its relationship to the best low-rank approximation of a matrix.

Theorem 1.23 (Best low-rank approximation of a matrix)**.** *Let* $A \in \mathbb{C}^{m\times n}$ *be a matrix and* (U, Σ, V^*) *an SVD of* A. *The best rank-*$\tilde{r}$ *of* A *in the induced 2-norm is given by*

$$A_{\tilde{r}} = U_{\tilde{r}} \Sigma_{\tilde{r}} V_{\tilde{r}}^* = \sum_{k=1}^{\tilde{r}} \sigma_k u_k v_k^*, \tag{1.6}$$

where $U_{\tilde{r}} = \begin{bmatrix} u_1, \dots, u_{\tilde{r}} \end{bmatrix} \in \mathbb{C}^{m\times\tilde{r}}$, $\Sigma_{\tilde{r}} = \mathrm{Diag}(\sigma_1, \dots, \sigma_{\tilde{r}}) \in \mathbb{C}^{\tilde{r}\times\tilde{r}}$ *and* $V_{\tilde{r}} = \begin{bmatrix} v_1, \dots, v_{\tilde{r}} \end{bmatrix} \in \mathbb{C}^{n\times\tilde{r}}$ *are the respective truncations of* U, Σ *and* V. *The error is given by*

$$\min_{\mathrm{Rank}\,\hat{A}=\tilde{r}} \|A - \hat{A}\|_2 = \|A - A_{\tilde{r}}\|_2 = \sigma_{\tilde{r}+1}. \tag{1.7}$$

The best approximation is unique if $\sigma_{\tilde{r}} > \sigma_{\tilde{r}+1}$.

Proof. It is straightforward to check that $\|A - A_{\tilde{r}}\|_2 = \left\| \sum_{j\geq\tilde{r}+1} s_j u_j v_j^* \right\|_2 = \sigma_{\tilde{r}+1}$. Moreover for a rank-$\tilde{r}$ matrix $\widetilde{A}_{\tilde{r}}$, by definition, there is a normalised vector $x \in \mathrm{Span}(v_1, \dots, v_{\tilde{r}+1})$ such that $\widetilde{A}_{\tilde{r}} x = 0$. Thus

$$\|A - \widetilde{A}_{\tilde{r}}\|_2 \geq \|(A - \widetilde{A}_{\tilde{r}})x\|_2 \geq \|Ax\|_2 \geq \sigma_{\tilde{r}+1}.$$

□

Remark 1.24. *We also have the same result in the Frobenius norm, where the best rank-$\tilde{r}$ approximation is given by (1.6). The error is however different, in that case we have that*

$$\min_{\text{Rank } \hat{A}=\tilde{r}} \|A - \hat{A}\|_F = \|A - A_{\tilde{r}}\|_F = \sqrt{\sum_{j \geq \tilde{r}+1} \sigma_j^2}. \tag{1.8}$$

If the singular values of a given matrix decay "fast"[3], a good approximation of the matrix can be obtained by truncating its SVD to small r. This means that an accurate representation of the matrix A can be obtained by storing the vectors $(u_i)_{1 \leq i \leq \tilde{r}} \in \mathbb{C}^m$, $(v_i)_{1 \leq i \leq \tilde{r}} \in \mathbb{C}^n$ and the scalars $(\sigma_i)_{1 \leq i \leq \tilde{r}}$. This requires a storage of $(m+n+1)\tilde{r}$ numbers which can be considerably smaller than mn if $\tilde{r} \ll m, n$.

The SVD is a tool that is widely used in reduced modelling or in statistics, where it is known as the *principal component analysis*.

[3] where the notion of fast decay depends on the application

Chapter 2

Direct linear solvers

2.1 Introduction

A linear system is a collection of m linear equations involving n unknowns, with $n, m \in \mathbb{N}$, taking the form

$$\sum_{k=1}^{n} a_{j,k}\, x_k = b_j \quad \forall j = 1 \dots m. \tag{2.1}$$

In these equations $x = (x_k)_{k=1}^n$ is the unknown vector, the $a_{j,k}$ are the entries (or coefficients) of the matrix of the linear system, and $b = (b_j)_{j=1}^m$ is the right hand side. In what follows the matrix of the linear system will be denoted by $\mathrm{A} = (a_{j,k})$, so that this linear system rewrites as

$$\mathrm{A}x = b. \tag{2.2}$$

In this course we will only consider invertible linear systems, which correspond to the situations where the matrix A admits an inverse denoted by A^{-1}. According to the rank-nullity theorem, a consequence of this assumption is that linear systems are square $m = n$ so that the matrix A shall admit as many rows as columns.

Effective solution methods for problems of the form (2.1) are called linear solvers. There exists essentially two families of linear solvers

- **Direct solvers** where the solution x is computed in a finite (but potentially large) number of operations.
- **Iterative solvers** where the linear system is reformulated as a fixed point problem $x = \Phi(x)$. The solution x is then approximated by means of an algorithm of the form $x^{(p+1)} = \Phi(x^{(p)})$.

In this chapter we will focus on direct solvers. Iterative solvers are the subject of subsequent chapters of this course.

2.2 Triangular systems

First let us consider the case of a lower triangular linear system, *i.e.* when $a_{j,k} = 0$ for $j < k$. In this case the matrix A admits the form

$$\mathrm{A} = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \ddots & \vdots \\ \vdots & * & \ddots & 0 \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

The algorithm to solve the problem $\mathrm{A}x = b$, called the *forward substitution method*, is quite natural. One first observes that the j-th row (j-th equation) can be re-written

$$x_j = \frac{1}{a_{j,j}} \big(b_j - \sum_{k=1}^{j-1} a_{j,k} x_k \big) \tag{2.3}$$

From this identity, the value of x_j is deduced from the values of $x_{j-1}, x_{j-2}, \dots x_1$. One thus starts with the first row that writes $x_1 = b_1/a_{1,1}$, then $x_2, x_3, \dots x_n$ are successively obtained by means of the reccurence relations (2.3).

Case of an upper triangular system In the case of an upper triangular system, that is when $a_{j,k} = 0$ for $j > k$, the solution principle is the same. This time the method is called *backward substitution method*. The procedure starts from the last row, then the following formula is applied

$$x_j = \frac{1}{a_{j,j}} \big(b_j - \sum_{k=j+1}^{n} a_{j,k} x_k \big) \tag{2.4}$$

Algorithmic complexity Let us examine the number of operations required for the forward substitution method with respect to n, when solving a lower triangular system. If we do not make any particular assumption on the matrix A, the solution procedure that we have just discussed requires

- n divisions,
- $n - 1$ substrations,
- $\sum_{i=2}^{n} \sum_{j=1}^{i-1} 1 = \sum_{i=2}^{n} (i-1) = n(n-1)/2$ multiplications,
- $\sum_{i=3}^{n} \sum_{j=1}^{i-2} 1 = \sum_{i=2}^{n} (i-2) = (n-1)(n-2)/2$ additions.

This leads to a total cost of $n + (n-1) + n(n-1)/2 + (n-1)(n-2)/2 = n^2$ elementary operations ("floating point operation" = flop). The algorithm that we have just described is thus said to admit an algorithmic complexity of $\mathcal{O}(n^2)$.

If we assume that the matrix A admits only $\mathcal{O}(1)$ nonzero term in each row (for example if it has a band structure with a band of fixed width), then the solution to this triangular system only costs $\mathcal{O}(n)$, which is the cost of a matrix-vector product.

2.3 Gaussian elimination

We come back to the case of an arbitrary linear system. The gaussian elimination method consists in reducing the initial linear system to an equivalent upper triangular system (obviously, with right hand side modified accordingly). In the sequel $e_j, j = 1 \dots n$ shall refer to the canonical basis of $\mathbb{C}^{n\times n}$.

We know that $\mathrm{A}x = b \iff \mathcal{L}^{(1)}\mathrm{A}x = \mathcal{L}^{(1)}b$ if the matrix $\mathcal{L}^{(1)} \in \mathbb{C}^{n\times n}$ is invertible. The matrix of the transformed linear system takes the form $\mathrm{A}^{(1)} := \mathcal{L}^{(1)}\mathrm{A} = (a^{(1)}_{j,k})_{j,k=1\dots n} \in \mathbb{C}^{n\times n}$, and $b^{(1)} := \mathcal{L}^{(1)}b$. Let us choose $\mathcal{L}^{(1)}$ so as to make sure that $a^{(1)}_{j,1} = 0$ for $j = 2, \dots n$ i.e. so that $\mathrm{A}^{(1)}$ takes the (block lower triangular) form

$$\mathrm{A}^{(1)} = \begin{bmatrix} a^{(1)}_{1,1} & * & \cdots & * \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{bmatrix}$$

It suffices to choose

$$\mathcal{L}^{(1)} := \mathrm{id} - \ell_1 \cdot e_1^\top$$
$$\text{where} \quad \ell_1^\top := (0, a_{2,1}/a_{1,1}, \dots, a_{n,1}/a_{1,1}).$$

Note that $\ell_1 \cdot e_1^\top$ is indeed a rank 1 matrix of size $n \times n$ not to be confused with $e_1^\top \cdot \ell_1$ (which is a simple scalar value). Besides we have $e_1^\top \cdot \ell_1 = 0$. Let us verify that $\mathcal{L}^{(1)}$ is invertible. We have $(\mathrm{id} - \ell_1 \cdot e_1^\top) \cdot (\mathrm{id} + \ell_1 \cdot e_1^\top) = \mathrm{id} - (\ell_1 \cdot e_1^\top)^2 = \mathrm{id} - \ell_1 \cdot (e_1^\top \cdot \ell_1) \cdot e_1^\top = \mathrm{id}$. We have just proved that

$$(\mathcal{L}^{(1)})^{-1} = \mathrm{id} + \ell_1 \cdot e_1^\top.$$

Note that $e_1^\top \cdot \mathrm{A}^{(1)} = e_1^\top \cdot \mathrm{A}$. To conclude, note the particular role played by the coefficient $a_{1,1}$ in this procedure. This coefficient is called first *pivot*. For the matrix $\mathcal{L}^{(1)}$ to be properly defined, this pivot needs to be non-zero.

Second iteration We can reiterate the procedure that we have just described. Indeed we have $\mathrm{A}^{(1)}x = b^{(1)} \iff \mathcal{L}^{(2)}\mathrm{A}^{(1)}x = \mathcal{L}^{(2)}b^{(1)}$ provided that $\mathcal{L}^{(2)}$ is invertible. Set $\mathrm{A}^{(2)} = \mathcal{L}^{(2)}\mathrm{A}^{(1)} = (\mathcal{L}^{(2)} \cdot \mathcal{L}^{(1)}) \cdot \mathrm{A} = (a^{(2)}_{j,k})_{j,k=1\dots n}$ and $b^{(2)} = \mathcal{L}^{(2)}b^{(1)}$. We will choose $\mathcal{L}^{(2)}$ so as to make sure that $\mathrm{A}^{(2)}$ takes the form

$$\mathrm{A}^{(2)} = \begin{bmatrix} \mathrm{U}_2 & * \\ 0 & * \end{bmatrix} \quad \text{where} \quad \mathrm{U}_2 = \begin{bmatrix} a^{(2)}_{1,1} & * \\ 0 & a^{(2)}_{2,2} \end{bmatrix}$$

Here the matrix U_2 is of size 2×2. To obtain the form above, it suffices then to define $\mathcal{L}^{(2)}$ as a lower triangular matrix given by

$$\mathcal{L}^{(2)} := \mathrm{id} - \ell_2 \cdot e_2^\top$$
$$\text{avec} \quad \ell_2^\top = (0, 0, a^{(1)}_{3,2}/a^{(1)}_{2,2}, \dots, a^{(1)}_{n,2}/a^{(1)}_{2,2}).$$

Again the matrix $\mathcal{L}^{(2)}$ is properly defined provided that the second pivot verifies $a^{(1)}_{2,2} \neq 0$ and, in this case, $\mathcal{L}^{(2)}$ is invertible with inverse given by $(\mathcal{L}^{(2)})^{-1} = \mathrm{id} + \ell_2 \cdot e_2^\top$. Then we have $e_1^\top \mathrm{A}^{(2)} = e_1^\top \mathrm{A}^{(1)} = e_1^\top \mathrm{A}$ (in particular $a^{(2)}_{1,1} = a^{(1)}_{1,1} = a_{1,1}$), and $e_2^\top \mathrm{A}^{(2)} = e_2^\top \mathrm{A}^{(1)}$.

General iteration We can generalize the procedure above up to any order. Assume that $\mathrm{A}^{(p-1)}$ has been defined for $p = 2 \dots n$, and that it is upper triangular on its $p-1$ first columns i.e. it admits the following form

$$\mathrm{A}^{(p-1)} = \begin{bmatrix} \mathrm{U}_{p-1} & * \\ 0 & * \end{bmatrix}$$

where U_{p-1} is an upper triangular matrix of size $(p-1) \times (p-1)$. We then define $\mathrm{A}^{(p)} = \mathcal{L}^{(p)} \cdot \mathrm{A}^{(p-1)} = (a^{(p)}_{j,k})_{j,k=1\dots n}$ and $b^{(p)} = \mathcal{L}^{(p)} b^{(p-1)}$, where $\mathcal{L}^{(p)} \in \mathbb{C}^{n \times n}$ is a lower triangular matrix given by the formula

$$\begin{aligned} &\mathcal{L}^{(p)} := \mathrm{id} - \ell_p \cdot e_p^\top \\ &\text{avec} \quad \ell_p^\top = (0, \dots, 0, a^{(p-1)}_{p+1,p}/a^{(p-1)}_{p,p}, \dots, a^{(p-1)}_{n,p}/a^{(p-1)}_{p,p}). \end{aligned} \tag{2.5}$$

A matrix of the form (2.5) is called a *Gauss transformation*. In the definition of ℓ_p, the first p coefficients are zero, which writes $e_j^\top \cdot \ell_p = 0 \, \forall j = 1 \dots p$. This implies $e_j^\top \mathcal{L}^{(p)} = e_j^\top, \forall j = 1 \dots p$. This construction of $\mathrm{A}^{(p)}$ hence guarantees that $e_j^\top \mathrm{A}^{(p)} = e_j^\top \mathcal{L}^{(p)} \mathrm{A}^{(p-1)} = e_j^\top \mathrm{A}^{(p-1)}$ for all $j = 1 \dots p$ and, reccursively, we deduce that

$$e_j^\top \mathrm{A}^{(p)} = e_j^\top \mathrm{A}^{(j-1)} \quad \forall j = 1 \dots p$$

setting $\mathrm{A}^{(0)} := \mathrm{A}$. Here again, the matrix $\mathcal{L}^{(p)}$ is properly defined only if the p-th pivot satisfies $a^{(p-1)}_{p,p} \neq 0$, and its inverse is given by $(\mathcal{L}^{(p)})^{-1} = \mathrm{id} + \ell_p \cdot e_p^\top$. To conclude, the construction above garantees that $\mathrm{A}^{(p)}$ admits the form

$$\mathrm{A}^{(p)} = \begin{bmatrix} \mathrm{U}_p & \star \\ 0 & \star \end{bmatrix}$$

where U_p is an upper triangular matrix of size $p \times p$. From what precedes, we see that the first p rows and columns remain unchanged when transforming the system from $\mathrm{A}^{(p-1)}$ to $\mathrm{A}^{(p)}$. We obtain the following formulas expressing $a^{(p)}_{j,k}$ with respect to $a^{(p-1)}_{j,k}$,

$$\begin{cases} a^{(p)}_{j,k} = a^{(p-1)}_{j,k} & \text{for } 1 \leq j \leq p \text{ or } 1 \leq k < p \\ a^{(p)}_{j,k} = 0 & \text{if } p < j \leq n \text{ and } k = p \\ a^{(p)}_{j,k} = a^{(p-1)}_{j,k} - a^{(p-1)}_{j,p} a^{(p-1)}_{p,k} / a^{(p-1)}_{p,p} & \text{if } p < j \leq n \text{ and } p < k \leq n \end{cases} \tag{2.6}$$

Conclusion of the algorithm The gaussian elimination algorithm terminates whenever $p = n - 1$, since the matrix $\mathrm{A}^{(n-1)}$ is then upper triangular. We have obtained that $\mathrm{A}x = b \iff \mathrm{A}^{(n-1)} x = b^{(n-1)}$. The later system being upper triangular (by construction...), it can be solved by a backward substitution algorithm as explained in the previous section 2.2.

Algorithmic complexity One can show that the gaussian elimination algorithm requires $2n^3/3 + n^2/3 - n$ operations (this is left as an exercise).

Remarks on the pivots The gaussian elimination algorithm is well defined provided that $a_{p,p}^{(p-1)} \neq 0$ for all p *i.e.* no pivot is equal to zero. There are classes of matrices for which this condition is systematically fulfilled. Here are three examples:

- symmetric positive definite matrices,
- row-wise diagonally dominant matrices: $|a_{j,j}| > \sum_{k\neq j} |a_{j,k}| \ \forall j = 1 \dots n$,
- column-wise diagonally dominant matrices: $|a_{j,j}| > \sum_{k\neq j} |a_{k,j}| \ \forall j = 1 \dots n$.

2.4 LU decomposition

The gaussian elimination method leads to a factorisation of the matrix under the form $\mathrm{A} = \mathrm{L}\cdot\mathrm{U}$ where L is lower triangular with $\mathrm{L}_{j,j} = 1, j = 1 \dots n$ and U is upper triangular. Indeed, let us start by setting $\mathrm{U} = \mathrm{A}^{(n-1)}$ which is the matrix obtained after the $n-1$-th step of the Gauss method. We thus have $\mathcal{L}^{(n-1)}\mathcal{L}^{(n-2)} \cdots \mathcal{L}^{(1)}\mathrm{A} = \mathrm{U} \iff \mathrm{A} = \mathrm{L}\cdot\mathrm{U}$ with

$$\mathrm{L} := (\mathcal{L}^{(1)})^{-1}(\mathcal{L}^{(2)})^{-1} \cdots (\mathcal{L}^{(n-1)})^{-1}$$

Let us study in more details the matrix L. It appears in factorised form. We will develop this product, taking account of the elementary property $e_j^\top \ell_p = 0$ for $j = 1 \dots p$. As a particular case of this property, we see that $e_1^\top \ell_2 = 0$, which leads to the conclusion that

$$\begin{aligned}(\mathcal{L}^{(1)})^{-1}(\mathcal{L}^{(2)})^{-1} &= (\mathrm{id} + \ell_1 e_1^\top)(\mathrm{id} + \ell_2 e_2^\top) \\ &= \mathrm{id} + \ell_1 e_1^\top + \ell_2 e_2^\top + \ell_1 (e_1^\top \ell_2) e_2^\top \\ &= \mathrm{id} + \ell_1 e_1^\top + \ell_2 e_2^\top\end{aligned}$$

We then proceed by induction on p to show that $(\mathcal{L}^{(1)})^{-1}(\mathcal{L}^{(2)})^{-1} \cdots (\mathcal{L}^{p})^{-1} = \mathrm{id} + \sum_{j=1}^{p} \ell_j e_j^\top$. We have just proved that this property holds for $p = 1$. Assume that it holds for p, and let us prove that it holds for $p+1$ as well. We have

$$\begin{aligned}(\mathcal{L}^{(1)})^{-1}(\mathcal{L}^{(2)})^{-1} \cdots (\mathcal{L}^{(p)})^{-1}(\mathcal{L}^{(p+1)})^{-1} &= (\mathrm{id} + \sum_{j=1}^{p} \ell_j e_j^\top)(\mathrm{id} + \ell_{p+1} e_{p+1}^\top) \\ &= \mathrm{id} + (\sum_{j=1}^{p} \ell_j e_j^\top) + \ell_{p+1} e_{p+1}^\top + (\sum_{j=1}^{p} \ell_j e_j^\top)\ell_{p+1} e_{p+1}^\top \\ &= \mathrm{id} + \sum_{j=1}^{p+1} \ell_j e_j^\top + (\sum_{j=1}^{p} \ell_j e_j^\top)\ell_{p+1} e_{p+1}^\top \\ &= \mathrm{id} + \sum_{j=1}^{p+1} \ell_j e_j^\top + \sum_{j=1}^{p} \ell_j (e_j^\top \ell_{p+1}) e_{p+1}^\top\end{aligned}$$

Since $e_j^\top \ell_{p+1} = 0$ for $j = 1 \dots p$, the last term in the right hand side above is zero, so that the property that we want to prove holds for $p+1$. We finally obtain, for $p = n-1$,

$$\mathrm{L} = \mathrm{id} + \sum_{j=1}^{n-1} \ell_j e_j^\top$$

Let us examine the j-th column of this matrix. We have $\mathrm{L} \cdot e_j = e_j + \ell_j$, with $(e_j + \ell_j)^\top = (0, \dots, 0, 1, a_{j+1,j}^{(j-1)}/a_{j,j}^{(j-1)}, \dots, a_{n,j}^{(j-1)}/a_{j,j}^{(j-1)})$. We see that L is indeed lower triangular, with 1's on the diagonal, and the expression that we have just obtained for its columns suggests that this matrix should be assembled on-the-fly during the gaussian elimination algorithm.

2.5 Factorisation algorithm

Let us now examine the algorithmic details of an LU decomposition. Before going into the description of the algorithm itself, we introduce a few simple notations. If J and K are two subsets of $[\![1, n]\!] := \{1, \dots n\}$, we will denote by $\mathrm{A}_{\mathrm{J,K}} \in \mathbb{C}^{|\mathrm{J}|\times|\mathrm{K}|}$ the submatrix obtained out of A by extracting the columns $k \in \mathrm{K}$ and the rows $j \in \mathrm{J}$ (here we denote $|\mathrm{J}|$ the cardinal of J, and accordingly for $|\mathrm{K}|$). For a $j \in [\![1, n]\!]$ and a subset $\mathrm{K} \subset [\![1, n]\!]$, we denote $\mathrm{A}_{j,\mathrm{K}}$ instead of $\mathrm{A}_{\{j\},\mathrm{K}}$ and, similarly, for $k \in [\![1, n]\!]$ and $\mathrm{J} \subset [\![1, n]\!]$ we will denote $\mathrm{A}_{\mathrm{J},k}$ instead of $\mathrm{A}_{\mathrm{J},\{k\}}$.

As suggested by the algorithm of the gaussian elimination described above, an LU decomposition method involves $n-1$ for a square matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$. Iteration p consists in an update of the matrix $\mathrm{A}^{(p-1)}$ to obtain the matrix $\mathrm{A}^{(p)}$. On the theoretical side, this update takes the form of a left-multiplication by the matrix $\mathcal{L}^{(p)} := \mathrm{id} - \ell_p \cdot e_p^\top$, but this is not the actual way this update takes place in practice: this would be unecessarily costly. Here are a few simple observations concerning this update:

- it does not modify the rows $1, \dots, p$
- it does not modify the columns $1, \dots, p-1$
- in column p, it cancels out the elements $a_{j,p}^{(p-1)}$ for $j = p+1, \dots n$

As a consequence, to change $\mathrm{A}^{(p-1)}$ into $\mathrm{A}^{(p)}$, it suffices to modify the block $\mathrm{A}_{\mathrm{J,J}}^{(p-1)}$ for $\mathrm{J} = [\![p+1, n]\!]$. Besides, the matrix $\mathcal{L}^{(p)}$ is entirely determined by ℓ_p. These elementary remarks together with §2.3, lead to Algorithm 2.1 that follows

Algorithm 2.1

function LU_NAIVE(A)
  $\mathrm{L} = \mathrm{id}$
  $\mathrm{A}^{(0)} = \mathrm{A}$
  for $p = 1 \dots n-1$ **do**
    $\mathrm{J} = [\![p+1, n]\!]$
    $\mathrm{L}_{\mathrm{J},p} = \mathrm{A}_{\mathrm{J},p}^{(p-1)} / \mathrm{A}_{p,p}^{(p-1)}$
    $\mathrm{A}_{\mathrm{J,J}}^{(p)} = \mathrm{A}_{\mathrm{J,J}}^{(p-1)} - \mathrm{L}_{\mathrm{J},p} \mathrm{A}_{p,\mathrm{J}}^{(p-1)}$
  end for
  $\mathrm{U} = \mathrm{A}^{(n-1)}$
  return L, U
end function

In fact Algorithm 2.1 makes use of many unnecessary intermediate variables. One may store L (resp. U) in the lower (resp. upper) triangular part of a single matrix $\mathrm{T} \in \mathbb{C}^{n\times n}$.

During this construction, it is also possible to store the matrices $\mathrm{A}^{(p)}$ inside T. This leads to Algorithm 2.2 that only involves a single additional matrix.

Once the LU decomposition of the matrix A is available and stored in the matrix T, we can solve the linear system $\mathrm{A}x = b$ by means of the successive application of a backward and a forward substitution method based on the upper and lower triangular parts of T (the diagonal coefficients of the lower triangular system must equal 1 hence do not need to be stored). The effective solution of a linear system making use of the a priori knowledge of the LU decomposition then takes the form of Algorithm 2.3 below.

Algorithm 2.2

function LU_DECOMPOSE(A)
  $\mathrm{T} = \mathrm{A}$
  for $p = 1 \dots n-1$ **do**
    $\mathrm{J} = [\![p+1, n]\!]$
    $\mathrm{T}_{\mathrm{J},p} = \mathrm{T}_{\mathrm{J},p} / \mathrm{T}_{p,p}$
    $\mathrm{T}_{\mathrm{J},\mathrm{J}} = \mathrm{T}_{\mathrm{J},\mathrm{J}} - \mathrm{T}_{\mathrm{J},p}\mathrm{T}_{p,\mathrm{J}}$
  end for
  return T
end function

Algorithm 2.3

function LU_SOLVE(A,b)
  $\mathrm{T} =$ LU_DECOMPOSE(A)
  `//L solve`
  for $j = 1 \dots n$ **do**
    $v_j = b_j$
    for $k = 1 \dots j-1$ **do**
      $v_j = v_j - \mathrm{T}_{j,k} v_k$
    end for
  end for
  `//U solve`
  for $p = 1 \dots n$ **do**
    $j = n - p + 1$
    $u_j = v_j$
    for $k = j+1 \dots n$ **do**
      $u_j = u_j - \mathrm{T}_{j,k} u_k$
    end for
    $u_j = u_j / \mathrm{T}_{j,j}$
  end for
  return u
end function

2.6 Partial pivoting

As we saw, the gaussian elimination algorithm stops if one of the pivots is zero i.e. $a_{q,q}^{(q-1)} = 0$. One way to circumvent this issue relies on a so-called *partial pivoting strategy*. It consists in a preliminary step taking place at each iteration q, where the rows q and r are swapped with r chosen so that

$$|a_{r,q}^{(q-1)}| = \max_{j=q\dots n} |a_{j,q}^{(q-1)}| \tag{2.7}$$

This is equivalent to a left multiplication of the matrix A^{q-1} by a permutation matrix P_q defined by: $\mathrm{P}_q \cdot e_q = e_r$, $\mathrm{P}_q \cdot e_r = e_q$ and $\mathrm{P}_q \cdot e_j = e_j$ if $j \neq q, r$. It is important to note that $(\mathrm{P}_q)^2 = \mathrm{id}$ and that $\mathrm{P}_q^\top = \mathrm{P}_q$. In addition observe that the coefficient $a_{r,q}^{(q-1)}$ in (2.7) necessarily satisfies $a_{r,q}^{(q-1)} \neq 0$ otherwise the matrix A would not be invertible.

With the pivoting strategy described above, the result of $n-1$ iterations of the gaussian elimination algorithm writes $\mathcal{L}^{(n-1)}\mathrm{P}_{n-1}\cdots\mathrm{P}_2\mathcal{L}^{(1)}\mathrm{P}_1\cdot\mathrm{A}=\mathrm{U}$. This can be rewritten equivalently as follows:

$$\begin{aligned}
&\mathrm{A}=\mathrm{P}_1(\mathcal{L}^{(1)})^{-1}\mathrm{P}_2\cdots\mathrm{P}_{n-1}(\mathcal{L}^{(n-1)})^{-1}\cdot\mathrm{U}\\
\iff\ &\mathrm{PA}=\mathrm{P}\cdot\mathrm{P}_1(\mathcal{L}^{(1)})^{-1}\mathrm{P}_2\cdots\mathrm{P}_{n-1}(\mathcal{L}^{(n-1)})^{-1}\cdot\mathrm{U}\\
&\text{with}\quad \mathrm{P}=\mathrm{P}_{n-1}\mathrm{P}_{n-2}\cdots\mathrm{P}_1\\
\iff\ &\mathrm{PA}=\mathcal{T}_1\cdot\mathcal{T}_2\cdots\mathcal{T}_{n-1}\cdot\mathrm{U}\\
&\text{where}\quad \mathcal{T}_q=\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot(\mathcal{L}^{(q)})^{-1}\cdot\mathrm{P}_{q+1}\cdots\mathrm{P}_{n-1}
\end{aligned}$$

In the calculation above, we have considered that $\mathcal{T}_{n-1}=(\mathcal{L}^{(n-1)})^{-1}$. Let us check that the matrices $\mathcal{T}_q$ are lower triangular. Recall that $(\mathcal{L}^{(q)})^{-1}=\mathrm{id}+\ell_q\cdot e_q^\top$, where $\ell_q\in\mathbb{C}^{n\times n}$, verifies $e_j^\top\cdot\ell_q=0$ for $j=1\dots q$. As a consequence, we have

$$\begin{aligned}
\mathcal{T}_q &= \mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot(\mathrm{id}+\ell_q\cdot e_q^\top)\cdot\mathrm{P}_{q+1}\cdots\mathrm{P}_{n-1}\\
&= \mathrm{id}+(\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot\ell_q)\cdot(e_q^\top\cdot\mathrm{P}_{q+1}\cdots\mathrm{P}_{n-1})\\
&= \mathrm{id}+(\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot\ell_q)\cdot(\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot e_q)^\top\\
&= \mathrm{id}+\ell'_q\cdot e_q^\top \qquad \text{with}\quad \ell'_q:=\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot\ell_q
\end{aligned}$$

In the last step of the calculation above, we have used the fact that $\mathrm{P}_q e_j=e_j$ whenever $q>j$. Using this same property, we also see that, for all $j=1\dots q$, we have $e_j^\top\cdot\ell'_q=e_j^\top\cdot\mathrm{P}_{n-1}\cdots\mathrm{P}_{q+1}\cdot\ell_q=(\mathrm{P}_{q+1}\cdots\mathrm{P}_{n-1}\cdot e_j)^\top\cdot\ell_q=e_j^\top\cdot\ell_q=0$. This proves that $\mathcal{T}_q$ is lower triangular. Setting this time $\mathrm{L}=\mathcal{T}_1\cdot\mathcal{T}_2\cdots\mathcal{T}_{n-1}$, we have obtained

$$\mathrm{P}\cdot\mathrm{A}=\mathrm{L}\cdot\mathrm{U}$$

where L is lower triangular, and U is upper triangular. Such a decomposition holds as soon as A is invertible.

Algorithm 2.4

function LU_PIVOT(A)
  $\mathrm{T}=\mathrm{A},\ \mathrm{P}=\mathrm{id}$
  for $q=1\dots n-1$ **do**
    `//pivoting`
    $r=\mathrm{argmax}_{j=q,\dots n}|\mathrm{T}_{j,q}|$
    $\mathrm{P}=\tau(q,r)\cdot\mathrm{P},\ \mathrm{T}=\tau(q,r)\cdot\mathrm{T}$
    `//gaussian elimination`
    $\mathrm{J}=[\![q+1,n]\!]$
    $\mathrm{T}_{\mathrm{J},q}=\mathrm{T}_{\mathrm{J},q}/\mathrm{T}_{q,q}$
    $\mathrm{T}_{\mathrm{J},\mathrm{J}}=\mathrm{T}_{\mathrm{J},\mathrm{J}}-\mathrm{T}_{\mathrm{J},q}\mathrm{T}_{q,\mathrm{J}}$
  end for
  return T,P
end function

A modified version of Algorithm 2.2 taking account of the row-wise partial pivoting is given

in Algorithm 2.4 above. In this algorithm $\tau(q,r) \in \mathbb{C}^{n\times n}$ is the transposition matrix such that $\tau(q,r)e_k = e_k$ if $k \neq q,r$, $\tau(q,r)e_q = e_r$ and $\tau(q,r)e_r = e_q$. Obviously, to compute $\tau(q,r)\cdot \mathrm{P}$ and $\tau(q,r)\cdot \mathrm{T}$, no need to perform a full matrix-matrix product (a costly operation with $\mathcal{O}(n^3)$ algorithmic complexity a priori); it suffices to swap rows q and r of matrices P and T (a fast operation with $\mathcal{O}(n)$ algorithmic complexity). At the end of the day, the solution algorithm proceeds as in Algorithm 2.3, applying the permutation P on the right hand side as a preliminary step.

The only computational overhead comes from comparing the pivots in the columns at each step of the LU factorisation. Since there are $1+2+\cdots+n-1 = \frac{(n-1)n}{2} = \mathcal{O}(n^2)$ comparisons, this additional cost is negligeable with respect to the total cost of computing the LU factorisation.

Remark 2.1. *In practice, the partial pivoting strategy is sufficient to accurately solve linear systems, although there are theoretical examples that show that it does not guarantee that the LU factors are not growing exponentially fast (see the exercise on the Wilkinson matrix). In that case, more elaborate or costly pivoting strategies can be considered. The interested reader may refer to [GL13, Chapter 3.4] or [Hig02, Chapter 9] for a more thorough discussion on that topic.*

2.7 Theoretical results regarding the LU factorisation

Apart from the construction and practical details of the LU decomposition that we have presented, theoretical questions naturally arise concerning the existence or uniqueness of such a factorisation.

Theorem 2.2. *Given an invertible matrix* $\mathrm{A} \in \mathbb{C}^{n\times n}$*, we have* $\det(\mathrm{A}_{\mathrm{J,J}}) \neq 0$ *for* $\mathrm{J} = [\![1, j]\!]$ *and for all* $j = 1 \dots n-1$ *if and only if there exists a unique pair* $\mathrm{L}, \mathrm{U} \in \mathbb{C}^{n\times n}$ *satisfying* $\mathrm{A} = \mathrm{LU}$ *with* U *upper triangular and invertible and* L *lower triangular with* $\mathrm{L}_{j,j} = 1 \, \forall j = 1 \dots n$*.*

The pair (L,U) *of matrices is called the* LU factorisation *of* A*.*

Proof. The converse is straightforward to check, as U invertible means that its diagonal entries are non zero, hence $A_{J,J} = L_{J,J}U_{J,J}$ with $L_{J,J} = (L_{ik})_{1\leq i,k\leq j}$ and $U_{J,J} = (U_{ik})_{1\leq i,k\leq j}$ is also invertible.

For the direct implication, we first check the uniqueness. Let (L,U) and $(\widetilde{L},\widetilde{U})$ be two LU factorisations of A. Then $A = LU = \widetilde{L}\widetilde{U}$ so $\widetilde{L}^{-1}L = \widetilde{U}U^{-1}$. The left hand side is a lower triangular matrix, with 1 on the diagonal, whereas the right hand side is an upper triangular matrix. This means that $\widetilde{L}^{-1}L = \widetilde{U}U^{-1} = \mathrm{id}_n$, thus $L = \widetilde{L}$ and $U = \widetilde{U}$.

For the existence, we proceed by induction on the dimension n. The result for $n = 1$ is obvious. Assume that the result holds for $\mathrm{A} \in \mathbb{C}^{m\times m}$ satisfying the assumption in the theorem with $m = 1 \dots n$, and let us prove that this result still holds $\mathrm{A} \in \mathbb{C}^{(n+1)\times(n+1)}$ invertible.

Suppose that A has a LU factorisation (L,U). Now let us set $\mathrm{B} := \mathrm{A}_{\mathrm{J,J}}$ for $\mathrm{J} = [\![1, n]\!]$, so that

$$\mathrm{A} = \begin{bmatrix} \mathrm{B} & a \\ b^* & \alpha \end{bmatrix} \tag{2.8}$$

where $a, b \in \mathbb{C}^n$ and $\alpha \in \mathbb{C}$. By the induction hypothesis, there is existence of the factorisation $\mathrm{B} = \tilde{\mathrm{L}}\tilde{\mathrm{U}}$. As a consequence we have the decomposition $\mathrm{A} = \mathrm{LU}$ if and only if there exist

$u, v \in \mathbb{C}^n$ and $\beta \in \mathbb{C} \setminus \{0\}$ such that

$$\mathrm{L} = \begin{bmatrix} \tilde{\mathrm{L}} & 0 \\ v^* & 1 \end{bmatrix}, \quad \mathrm{U} = \begin{bmatrix} \tilde{\mathrm{U}} & u \\ 0 & \beta \end{bmatrix} \quad \text{and} \quad \begin{array}{l} \tilde{\mathrm{L}}u = a \\ \tilde{\mathrm{U}}^* v = b \\ \beta + v^* u = \alpha \end{array} \tag{2.9}$$

Since $\tilde{\mathrm{L}}$ and $\tilde{\mathrm{U}}$ are invertible, we have the existence of u, v, β solution to (2.9). It remains to check that U is indeed invertible, but by construction, L is invertible, and by assumption A is invertible, so $U = L^{-1}A$ is invertible. □

The criterion on the $\det(\mathrm{A}_{\mathrm{J},\mathrm{J}})$'s mentionned above is not very easy to verify in practice. However there exist certain classes of matrices for which this criterium is systematically satisfied. This is the case of diagonally dominant matrices. Recall that a matrix $\mathrm{A} = (a_{j,k}) \in \mathbb{C}^{n\times n}$ is said row-wise diagonally dominant if $|a_{j,j}| > \sum_{k\neq j} |a_{j,k}| \ \forall j = 1 \dots n$. Similarly it is said column-wise diagonally dominant when $|a_{j,j}| > \sum_{k\neq j} |a_{k,j}| \ \forall j = 1 \dots n$.

Proposition 2.3. *If the matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$ is either row-wise or column-wise diagonally dominant, then it admits a unique LU-decomposition.*

Proof. Let us assume row-wise diagonal dominance, since the case of a column-wise diagonal dominance proceeds very similarly. According to Theorem 2.2, it suffices to show that $\det(\mathrm{A}_{\mathrm{J},\mathrm{J}}) \neq 0$ for $\mathrm{J} = [\![1, k]\!]$ for all $k = 1 \dots n$. It appears clear that if A is row-wise diagonally dominant, this is also the case for $\mathrm{A}_{\mathrm{J},\mathrm{J}}$. Hence there remains to verify that a row-wise diagonally dominant matrix is necessarily invertible.

Let us proceed by contradiction and consider a row-wise diagonally dominant matrix $\mathrm{B} = (b_{j,k}) \in \mathbb{C}^{n\times n}$ such that $\mathrm{B}u = 0$ for a certain $u = (u_j) \in \mathbb{C}^n \setminus \{0\}$. Let $p \in \{1 \dots n\}$ such that $|u_p| = \max_{j=1\dots n} |u_j|$. Then we have $0 = b_{p,p}u_{p,p} + \sum_{j\neq p} b_{p,j}u_j$ and thus

$$|b_{p,p}u_{p,p}| = |\sum_{j\neq p} b_{p,j}u_j| \leq \sum_{j\neq p} |b_{p,j}| \max_{j=1\dots n} |u_j| \leq \sum_{j\neq p} |b_{p,j}||u_{p,p}|$$

which contradicts the diagonal dominance. □

The next result shows that if the matrix A has a band structure with a certain width, then each of the factors of the LU decomposition admits the same band structure.

Proposition 2.4. *Let $\mathrm{A} \in \mathbb{C}^{n\times n}$ a matrix admitting a unique LU factorisation. Let us assume in addition that there exists $p \geq 0$ such that $\mathrm{A}_{j,k} = 0$ if $|j-k| > p$. Then we also have $L_{j,k} = U_{j,k} = 0$ for $|j-k| > p$.*

Proof. Again we proceed by induction on the band width, and assume that the result holds for all matrices uniquely LU-factorisable with a band width of m with $1 \leq m \leq n$. We pick a matrix $\mathrm{A} \in \mathbb{C}^{(n+1)\times(n+1)}$ satisfying the assumptions of the proposition we are seeking to prove. Coming back to the notations of the proof of Theorem 2.2, we must in particular have (2.9) with $\tilde{\mathrm{L}}_{j,k} = \tilde{\mathrm{U}}_{j,k} = 0$ whenever $|j-k| > p$ according to the induction hypothesis.

Let us also take the notation $u = (u_j), v = (v_j), a = (a_j), b = (b_j)$ for the vectors coming into play in (2.9) and (2.8). To prove that L and U admit the same band structure, that is $\mathrm{L}_{j,k} = \mathrm{U}_{j,k} = 0$ for $|j-k| > p$, there remains to verify that $u_j = v_j = 0$ for $|j-n| > p \iff j < n-p$. We already know that $a_j = b_j = 0$ for $j < n-p$ according to the band structure satisfied by A. On the other hand $\tilde{\mathrm{L}}$ and $\tilde{\mathrm{U}}^*$ are lower triangular, hence $(\tilde{\mathrm{L}})^{-1}$ and $(\tilde{\mathrm{U}}^*)^{-1}$ also, and we have $u = (\tilde{\mathrm{L}})^{-1}a$ and $v = (\tilde{\mathrm{U}}^*)^{-1}b$, which indeed implies $u_j = v_j = 0$ for $j < n-p$. □

2.8 Cholesky factorisation

Recall that a matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$ is hermitian whenever $\mathrm{A} = \mathrm{A}^*$, and it is called positive when $x^*\mathrm{A}x \in (0, +\infty)$ for all $x \neq 0$. Finally, when it is positive, the matrix A is called definite if $x^*\mathrm{A}x = 0 \Rightarrow x = 0$. In the case where the matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$ is hermitian positive definite (HPD), its LU-factorisation simplifies, and we can obtain simpler formulas.

Theorem 2.5. *Let $\mathrm{A} = (a_{j,k})_{j,k=1\ldots n} \in \mathbb{C}^{n\times n}$ be hermitian positive definite. Then there exists a lower triangular matrix $\mathrm{H} \in \mathbb{C}^{n\times n}$ whose diagonal terms are real, positive and such that the following so-called Cholesky factorisation holds*

$$\mathrm{A} = \mathrm{H}\mathrm{H}^*.$$

Proof. As A is hermitian positive-definite, for all $1 \leq j \leq n$, $J = \{1, \ldots, j\}$, $A_{J,J}$ is invertible, hence A has a unique LU decomposition (L, U). Let D be the diagonal matrix of the diagonal elements of U. Since A is invertible, U is invertible, so there is an upper triangular matrix $\widetilde{U}$ with entries equal to 1 on the diagonal such that $U = D\widetilde{U}$. Then we have $A = LD\widetilde{U}$. A is hermitian, thus $A^* = \widetilde{U}^*D^*L^*$. Since the LU decomposition is unique, we have that $L = \widetilde{U}^*$, thus $A = LDL^*$. $L^{-1}AL^{-*}$ is also hermitian positive-definite, since for any vector $x \in \mathbb{C}^n$, we have $x^*L^{-1}AL^{-*}x = (L^{-*}x)^*A(L^{-*}x) = 0 \Rightarrow L^{-*}x = 0 \Rightarrow x = 0$. Thus the diagonal elements of D are all real and positive. Take $H = L\,\mathrm{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ where $D = \mathrm{diag}(d_1, \ldots, d_n)$, then $A = HH^*$. □

An explicit algorithm can be proposed for the Cholesky factorisation. By directly expressing the matrix product $\mathrm{A} = \mathrm{H}\mathrm{H}^*$, coefficient by coefficient, we obtain $a_{j,j} = \sum_{k=1}^{j} |h_{j,k}|^2$ and $a_{j,k} = \sum_{p=1}^{k} h_{j,p}\overline{h}_{k,p}$ for $k < j$. From there we deduce the following formulas

$$\begin{aligned} h_{j,j} &= (a_{j,j} - \textstyle\sum_{k=1}^{j-1} |h_{j,k}|^2)^{1/2}, \\ h_{j,k} &= (a_{j,k} - \textstyle\sum_{p=1}^{k-1} h_{j,p}\overline{h}_{k,p})/h_{k,k} \quad \text{pour } k < j. \end{aligned}$$

The construction of the matrix H then proceeds for j growing from 1 to n and for k growing from 1 to j. These formulas yield Algorithm 2.5.

Remark 2.6. *The Cholesky factorisation is* not *the square root of the matrix in general. The square root S of a hermitian positive-definite matrix A is a hermitian positive-definite matrix satisfying $A = S^2$. In the case where A is hermitian positive-definite, there is a link between both matrices. Let H be the Cholesky factor of A, $H = U\Sigma V^*$ is a SVD of H, then $S = U\Sigma U^*$ is a square root of A:*

$$S^2 = (U\Sigma U^*)(U\Sigma U^*) = (U\Sigma V^*)(V^*\Sigma U) = HH^* = A.$$

Algorithm 2.5

```
function CHOLESKY(A)
    H = 0
    for j = 1...n do
        for k = 1...j do
            H_{j,k} = A_{j,k}
            for p = 1...k-1 do
                H_{j,k} = H_{j,k} - H_{j,p} conj(H)_{k,p}
            end for
            if k < j then
                H_{j,k} = H_{j,k} / H_{k,k}
            else
                H_{j,j} = sqrt(H_{j,j})
            end if
        end for
    end for
    return H
end function
```

Chapter 3

Stationary iterative methods

In this chapter, we are introducing *iterative methods* to solve the linear system

$$Ax_* = b, \tag{3.1}$$

where $A \in \mathbb{C}^{n\times n}$ is a matrix assumed to be invertible, $x_* \in \mathbb{C}^n$ is the vector of unknowns and $b \in \mathbb{C}^n$ is the right-hand side vector.

The general idea of iterative methods is to define a sequence of vectors $(x^{(k)})_{k\in\mathbb{N}}$ such that $\lim x^{(k)} = x_*$. In this regard, there are two big families of iterative methods that can be distinguished:

1. the stationary iterative methods where $(x^{(k)})$ is defined by
$$x^{(k+1)} = Gx^{(k)} + v,$$
where $G \in \mathbb{C}^{n\times n}$ is some iteration matrix and $v \in \mathbb{C}^n$;
2. Krylov subspace methods where for all n, $x^{(k)} \in \mathrm{Span}_{0\leq j\leq k-1}(A^j b)$.

The advantage of using an iterative solver instead of a Gaussian elimination process relies on the following observation: the Gaussian elimination algorithm requires $\mathcal{O}(n^3)$ operations for a dense matrix in order to compute x_*. This quickly becomes untractable. The iterative methods on the other hand only requires matrix-vector multiplication whose cost scales as $\mathcal{O}(n^2)$ for dense matrices and $\mathcal{O}(n)$ for sparse matrices. *If* the iterative method converges quickly, an approximate solution can be computed using $\mathcal{O}(kn^2)$ for a dense matrix ($\mathcal{O}(kn)$ for a sparse matrix) with $k \ll n$ where k is the number of steps of the iterative method. With an efficient iterative method, it is possible to gain a factor n in the resolution of the linear system.

Remark 3.1. *We say that a matrix $A \in \mathbb{C}^{n\times n}$ is sparse if for each row of A, there are $s \ll n$ nonzero elements. The matrix-vector product of with a sparse matrix scales as $\mathcal{O}(sn)$ instead of n^2 as for each $1 \leq i \leq n$, we have*

$$(Ax)_i = \sum_{j=1}^{n} A_{ij}x_j = \sum_{j \text{ such that } A_{ij}\neq 0} A_{ij}x_j,$$

where the last term contains at most s nonzero terms. In terms of storage, it is not necessary to store all the elements of the matrix, but only the nonzero ones. We thus also gain a factor n.

A typical example of a sparse matrix is the discrete Laplacian matrix which is tridiagonal, hence it is sparse with $s = 3$. This type of matrices appear in graph theory, when the graph is not well-connected or in differential equations, when the operators are discretised using finite elements or finite differences.

There are several ways to store a sparse matrix, we will mention these two that are standard:

1. *the COO format (COOrdinates): the matrix A is given as three vectors (I, J, V), each of size* nnz *where* nnz *is the number of nonzero entries in A. For each $1 \leq k \leq \text{nnz}$, we have $V_k = A_{I_k,J_k}$, or said differently, the k-th entry of V corresponds to the coordinate (I_k, J_k) in the matrix A. This format is easy to manipulate, but is not efficient for performing matrix-vector multiplications, as the vectors (I, J) are not necessarily ordered. For the matrix*
$$A = \begin{bmatrix} 2 & 6 & 0 & 0 \\ 0 & 0 & 0 & 7 \\ -1 & 13 & 0 & 0 \\ 0 & -2 & -4 & 0 \end{bmatrix},$$
an example of the COO format is given as
$$I = \big[1, 1, 2, 3, 3, 4, 4\big], \quad J = \big[1, 2, 4, 1, 2, 2, 3\big], \quad V = \big[2, 6, 7, -1, 13, -2, -4\big].$$

2. *the CSR format (Compressed Sparse Row): the matrix $A \in \mathbb{C}^{n\times n}$ is given as three vectors* $(\text{row}, \text{col}, \text{val})$ *where* col, val *have size* nnz *and* row *is of size $n + 1$. In this case, we have for each $1 \leq k \leq \text{nnz}$, $\text{val}_k = A_{i,\text{col}_k}$ where i is such that $\text{row}_i \leq k < \text{row}_{i+1}$. Compared to the COO format, the storage cost is reduced since the size* row *is usually much smaller than* nnz*. Moreover, now that the entries are ordered, the matrix-vector multiplication is much more efficient than in the COO format. For the example above, the corresponding CSR format is given by*
$$\text{row} = \big[1, 3, 4, 6, 8\big], \quad \text{col} = \big[1, 2, 4, 1, 2, 2, 3\big], \quad \text{val} = \big[2, 6, 7, -1, 13, -2, -4\big].$$

The central question for iterative methods is the convergence and the speed of convergence of these algorithms. We are first presenting historical iterative methods, that are rarely used in practice but which give a good insight on these methods.

3.1 Principle of stationary iterative methods

The general framework of this type of methods is to define a splitting of the matrix $A = M - N$, where $M, N \in \mathbb{C}^{n\times n}$ and define the stationary iterative method by
$$\begin{cases} x^{(0)} \in \mathbb{C}^n \\ Mx^{(k+1)} = Nx^{(k)} + b, \quad k \geq 1. \end{cases} \tag{3.2}$$
If the sequence $(x^{(k)})$ converges to a vector x_∞, then $Mx_\infty = Nx_\infty + b$ hence $Ax_\infty = b$. Thus the limit solves the linear system (3.1).

To study the convergence of the sequence $(x^{(k)})$, we see that $Mx_* = Nx_* + b$, so $x^{(k)} - x_*$ satisfies
$$x^{(k+1)} - x_* = M^{-1}Nx^{(k)} - M^{-1}b - x_* = M^{-1}N(x^{(k)} - x_*). \tag{3.3}$$
Hence the convergence of the sequence $(x^{(k)})$ is governed by the properties of the matrix $M^{-1}N$, and in particular, as we will see in the next section, the spectral radius of $M^{-1}N$.

3.1.1 Convergence of stationary iterative methods

Proposition 3.2. *For any matrix* $\mathrm{A} \in \mathbb{C}^{n\times n}$ *and any* $\epsilon > 0$*, there exists an induced norm* $\| \ \|_*$ *over* $\mathbb{C}^{n\times n}$ *such that*

$$\|\mathrm{A}\|_* \leq \varrho(\mathrm{A}) + \epsilon. \tag{3.4}$$

We already know that for any induced matrix norm $\|\cdot\|$, we have that $\varrho(A) \leq \|A\|$. The proposition above shows that there is a particular choice of vector norm, that *depends* on the matrix such that the converse is true, up to a tolerance ϵ.

Proof. Consider a matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$. We have to propose an induced norm satisfying (3.4). According to Proposition 1.10, there exists a unitary matrix $Q \in \mathbb{C}^{n\times n}$ such that $\mathrm{T} := Q^*\mathrm{A}Q$ is upper triangular,

$$\mathrm{T} = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & t_{n,n} \end{bmatrix}.$$

Note that the entries $t_{j,j}, j = 1 \dots n$ are the eigenvalues of the matrix A. Given some $\delta > 0$ that we shall choose a posteriori, set $\mathrm{D}_\delta := \mathrm{diag}(1, \delta, \dots, \delta^{n-1})$ and define the matrix $\mathrm{T}_\delta := \mathrm{D}_\delta^{-1}Q^{-1}\mathrm{A}Q\mathrm{D}_\delta = (Q\mathrm{D}_\delta)^{-1}\mathrm{A}(Q\mathrm{D}_\delta) = \mathrm{D}_\delta^{-1}\mathrm{T}\mathrm{D}_\delta$. Examining its values, we see that

$$\mathrm{T}_\delta = \begin{bmatrix} t_{1,1} & \delta t_{1,2} & \cdots & \delta^{n-1}t_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \delta t_{n-1,n} \\ 0 & \cdots & 0 & t_{n,n} \end{bmatrix}.$$

The matrix above decomposes as $\mathrm{T}_\delta = \Lambda + \mathrm{R}_\delta$ where $\Lambda = \mathrm{diag}(t_{1,1}, \dots, t_{n,n})$ and R_δ is the upper triangular part located strictly above the diagonal. According to Proposition 1.3, we have $\lim_{\delta\to 0} \|\mathrm{R}_\delta\|_1 = 0$. Given some ϵ, we can choose δ small enough to guarantee that $\|\mathrm{R}_\delta\|_1 \leq \epsilon$. In addition, one readily checks that $\|\Lambda\|_1 = \max_{j=1\dots n} |t_{j,j}| = \varrho(\mathrm{A})$. With δ chosen as indicated, we obtain $\|\mathrm{T}_\delta\|_1 \leq \varrho(\mathrm{A}) + \epsilon$. Now set $Q_\delta := Q\mathrm{D}_\delta$ and consider the norm $\|x\|_* := \|Q_\delta^{-1}x\|_1$. With the matrix norm $\| \ \|_*$ induced by $\| \ \|_*$, the matrix A then satisfies

$$\|\mathrm{A}\|_* = \sup_{x\in\mathbb{C}^n\setminus\{0\}} \frac{\|\mathrm{A}x\|_*}{\|x\|_*} = \sup_{x\in\mathbb{C}^n\setminus\{0\}} \frac{\|Q_\delta^{-1}\mathrm{A}Q_\delta x\|_1}{\|x\|_1} = \|\mathrm{T}_\delta\|_1 \leq \varrho(\mathrm{A}) + \epsilon.$$

□

Theorem 3.3 (Convergence of stationary iterative methods)**.** *Let* $A \in \mathbb{C}^{n\times n}$ *be invertible,* $b \in \mathbb{C}^n$ *and* $x_* = A^{-1}b$*. The sequence* $(x^{(k)})_{k\geq 0}$ *defined by Equation (3.2) converges to* x_* *for any* $x^{(0)} \in \mathbb{C}^n$ *if and only if* $\rho(M^{-1}N) < 1$*, where* $\rho(M^{-1}N) = \max\{|\lambda|, \lambda \text{ eigenvalue of } M^{-1}N\}$*.*

Proof. If $\rho(M^{-1}N) < 1$ then there is an induced matrix norm $\|\cdot\|_*$ by a vector norm such that $\|M^{-1}N\|_* < 1$. Thus we have

$$\|x^{(k)} - x_*\|_* \leq \|M^{-1}N(x^{(k-1)} - x_*)\|_* \leq \|M^{-1}N\|_*\|x^{(k-1)} - x_*\|_* \leq \|M^{-1}N\|_*^k\|x^{(0)} - x_*\|_*.$$

Thus $\lim x^{(k)} = x_*$.

On the other hand if $\rho(M^{-1}N) \geq 1$ then there is an eigenvector $y \in \mathbb{C}^n$ of $M^{-1}N$ such that $\|(M^{-1}N)^k y\| = \rho(M^{-1}N)^k\|y\|$ does not converge to 0 as k goes to infinity. □

It remains to choose the matrix M in a wise manner, such that at each step the inversion of M has a cost comparable to a matrix-vector product.

3.2 Classical iterative methods

To define the methods in this section, we introduce the following notation $D, E, F \in \mathbb{C}^{n\times n}$ such that $A = D - E - F$ with

$$D = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a_{nn} \end{bmatrix}, \quad -E = \begin{bmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{bmatrix}, \text{ and } -F = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \dots & 0 & 0 \end{bmatrix}.$$

3.2.1 Jacobi method

For the Jacobi method, we set $M = D$ and $N = E + F$.

In that case, the i-th entry of the vector $x^{(k)}$ is given by

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

The cost of one iteration in the Jacobi method consists in applying $E+F$ which costs as $\mathcal{O}(n^2)$ for a dense matrix, and $\mathcal{O}(n)$ for a sparse matrix and inverting a diagonal matrix, which costs n divisions. Thus overall, the cost of a Jacobi iteration, scales as $\mathcal{O}(n^2)$ operations for dense matrices and $\mathcal{O}(n)$ for sparse matrices.

Remark 3.4. *Since the entries of $x^{(k)}$ only depend on $x^{(k-1)}$, its entries can be updated in parallel.*

We can ensure that the Jacobi method converges for the class of row-wise diagonally dominant matrices.

Proposition 3.5. *If A is row-wise diagonally dominant,* i.e. *for each $1 \leq i \leq n$, $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$, then the Jacobi method converges.*

Proof. We simply need to check that the spectral radius $\rho(M^{-1}N) < 1$ and use Theorem 3.3 to conclude on the convergence. For $y \in \mathbb{C}^n$, we have

$$\begin{aligned} |(M^{-1}Ny)_i| &= \Big| \sum_{j\neq i} \frac{a_{ij}}{a_{ii}} y_j \Big| \\ &< \|y\|_\infty, \end{aligned}$$

thus $\|M^{-1}N\|_\infty < 1$, so $\rho(M^{-1}N) \leq \|M^{-1}N\|_\infty < 1$. □

3.2.2 Gauss-Seidel method

For the Gauss-Seidel method, we set $M = D - E$ and $N = F$.

In terms of number of operations, the Gauss-Seidel algorithm requires the inversion of a triangular system which scales as $\mathcal{O}(n^2)$ if the matrix is dense, but as $\mathcal{O}(n)$ if the matrix

is sparse. Thus the total cost of one iteration of the Gauss-Seidel method scales as $\mathcal{O}(n^2)$ operations for a dense matrix and $\mathcal{O}(n)$ operations for a sparse matrix.

In that case, the i-th entry of the vector $x^{(k)}$ is given by

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

Once $x_i^{(k)}$ is computed, $x_i^{(k-1)}$ is not useful anymore. The update can be implemented in place. Contrary to the Jacobi method, the Gauss-Seidel algorithm is hardly parallelisable.

We have the same convergence theorem as previously.

Proposition 3.6. *If A is row-wise diagonally dominant,* i.e. *for each $1 \leq i \leq n$, $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$, then the Gauss-Seidel method converges.*

Proof. Let $y, z \in \mathbb{C}^n$ such that $z = M^{-1}Ny$. Then we have $Mz = Ny$ so

$$a_{ii} z_i = -\sum_{j>i} a_{ij} y_j - \sum_{j<i} a_{ij} z_j.$$

Let i_0 such that $|z_{i_0}| = \|z\|_\infty$. Then

$$|a_{i_0 i_0} z_{i_0}| \leq \sum_{j<i_0} |a_{i_0 j}| \|z\|_\infty + \sum_{j>i_0} |a_{i_0 j}| \|y\|_\infty$$

but since A is diagonally dominant

$$|a_{i_0 i_0}| - \sum_{j<i_0} |a_{i_0 j}| > \sum_{j>i_0} |a_{i_0 j}| > 0,$$

thus

$$\|z\|_\infty \leq \frac{\sum_{j>i_0} |a_{i_0 j}|}{|a_{i_0 i_0}| - \sum_{j<i_0} |a_{i_0 j}|} \|y\|_\infty \leq \max_{1\leq i_0 \leq n} \Big(\frac{\sum_{j>i_0} |a_{i_0 j}|}{|a_{i_0 i_0}| - \sum_{j<i_0} |a_{i_0 j}|} \Big) \|y\|_\infty.$$

Since A is diagonally dominant, $\max_{1\leq i_0\leq n} \Big(\frac{\sum_{j>i_0} |a_{i_0 j}|}{|a_{i_0 i_0}| - \sum_{j<i_0} |a_{i_0 j}|} \Big) < 1$. This shows that $\rho(M^{-1}N) \leq \|M^{-1}N\|_\infty < 1$. □

3.2.3 Successive over relaxation (SOR) method

For the SOR method, we have a positive parameter ω and we set $M_\omega = \frac{1}{\omega} D - E$ and $N_\omega = (\frac{1}{\omega} - 1)D + F$.

The SOR method also involves the inversion of a triangular matrix, as such, the cost of one iteration of the algorithm scales as $\mathcal{O}(n^2)$ for a dense matrix and $\mathcal{O}(n)$ for a sparse matrix.

We also have the same type of convergence result.

Proposition 3.7. *If A is row-wise diagonally dominant,* i.e. *for each $1 \leq i \leq n$, $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$, and if $0 < \omega \leq 1$ then the SOR method converges.*

Proof. Exercise. □

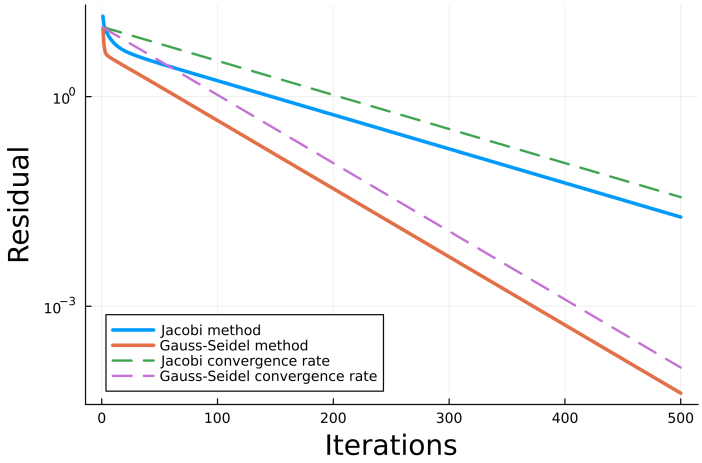
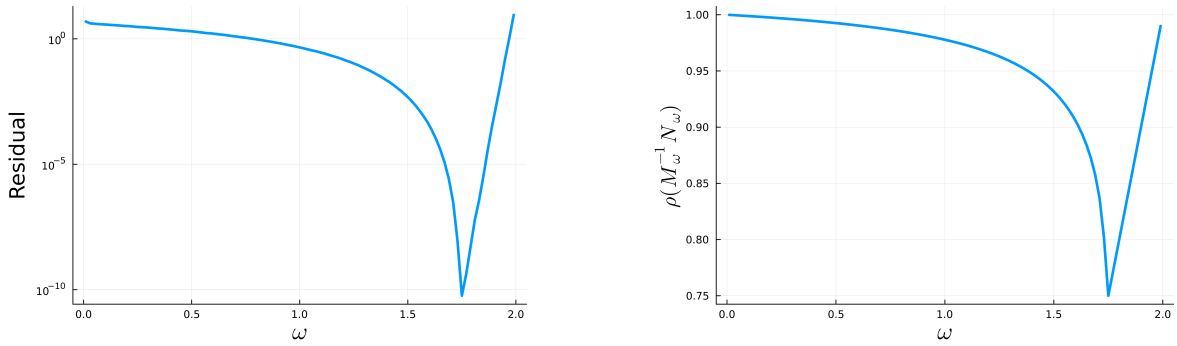


Figure 3.1: Jacobi and Gauss-Seidel methods for the one-dimensional discrete Laplacian. The speed of convergence is asymptotically given by the spectral radius of the iteration matrix of both methods.

For a given matrix A, one can wonder how to optimise ω in order to minimise the number of iterations, hence for that, we can try to find ω such that the spectral radius $\rho(M_\omega^{-1}N_\omega)$ is the smallest. It turns out that it is tricky to find the optimal relaxation parameter, as it depends heavily on the matrix. For certain classes of matrices, the explicit value of the optimal relaxation parameter is known (for example for symmetric, positive-definite tridiagonal matrices), but usually it depends on the eigenvalues of some matrix, that is not straightforward to compute.



(a) Residuals after 100 iterations

(b) Spectral radius

Figure 3.2: SOR method for the one-dimensional discrete Laplacian. The speed of convergence of the SOR method strongly depends on the choice of the parameter ω.

3.3 Richardson iteration

For Richardson iteration, the method corresponds to taking $M = \frac{1}{\alpha}\,\mathrm{id}$ and $N = \frac{1}{\alpha}\,\mathrm{id} - A$, with $\alpha \in \mathbb{R}$ a parameter:

$$x^{(k+1)} = (\mathrm{id} - \alpha A)x^{(k)} + \alpha b. \tag{3.5}$$

Proposition 3.8. *Assume that $A \in \mathbb{C}^{n\times n}$ is invertible and diagonalisable with eigenvalues $\lambda_1, \dots, \lambda_n$. Then the Richardson iteration converges if and only if $0 < \alpha < 2\frac{\min \mathrm{Re}(\lambda_j)}{|\lambda_j|^2}$ or $2\frac{\max \mathrm{Re}(\lambda_j)}{|\lambda_j|^2} < \alpha < 0$.*

Proof. Again we need to study the spectral radius of $M^{-1}N = (\mathrm{id} - \alpha A)$. The eigenvalues of $\mathrm{id} - \alpha A$ are simply $1 - \alpha\lambda_j, j = 1 \dots n$. Hence $\rho(M^{-1}N) < 1 \iff \forall\, 1 \le j \le n, |1 - \alpha\lambda_j|^2 < 1$. But $|1 - \alpha\lambda_j|^2 = 1 - 2\alpha\,\mathrm{Re}(\lambda_j) + \alpha^2|\lambda_j|^2 < 1$ thus the condition is

$$\forall\, 1 \le j \le n, \alpha^2 < 2\alpha\,\mathrm{Re}(\lambda_j).$$

This can be satisfied only if α has the same sign as all the λ_j and we find the result. □

Suppose now that the matrix A is diagonalisable and has only positive eigenvalues $0 < \lambda_1 < \cdots < \lambda_n$. We can wonder for which value α, the spectral radius of the iteration matrix $\mathrm{id} - \alpha A$ is the smallest:

$$\rho(\mathrm{id} - \alpha A) = \max(|1 - \alpha\lambda_1|, \dots, |1 - \alpha\lambda_n|) = \max(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|).$$

Proposition 3.9. *The spectral radius of the iteration matrix $\mathrm{id} - \alpha A$ is minimal for*

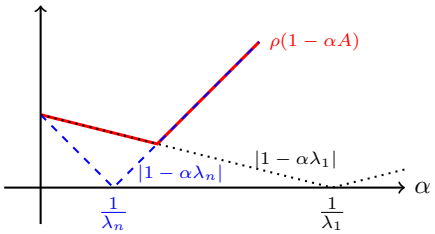
$$\alpha = \frac{2}{\lambda_1 + \lambda_n},$$

and for this value we have

$$\rho(\mathrm{id} - \alpha A) = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}.$$

Proof. We have that

$$\rho(\mathrm{id} - \alpha A) = \max_{1\le k\le n} |1 - \alpha\lambda_k| = \max(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|).$$



Graphically, we see that the minimal value of $\rho(\mathrm{id} - \alpha A)$ is attained when

$$-1 + \alpha\lambda_n = 1 - \alpha\lambda_1,$$

which gives $\alpha = \frac{2}{\lambda_1+\lambda_n}$ and $\rho(\mathrm{id} - \alpha A) = \frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1}$. □

Remark 3.10. *For a Hermitian positive definite matrix, this can be rewritten as*

$$\rho(\mathrm{id} - \alpha A) = \frac{\mathrm{cond}_2(A) - 1}{\mathrm{cond}_2(A) + 1}.$$

If the condition number $\mathrm{cond}_2(A)$ is large, the spectral radius of the iteration matrix is close to 1.

3.3.1 Interpretation as a gradient descent method

Suppose that $A \in \mathbb{R}^{n\times n}$ is a symmetric positive-definite matrix and consider the functional F

$$F(x) = \frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle, \tag{3.6}$$

where $\langle\cdot,\cdot\rangle$ denotes the scalar product of $\mathbb{R}^n$. Since A is positive-definite, the functional F is convex. Moreover, $\lim_{\|x\|\to\infty} F(x) = \infty$, hence F has a unique minimum satisfying $\nabla F(x_*) = Ax_* - b = 0$. This means that to solve the linear problem $Ax = b$, we can use a minimisation algorithm to the functional F. A simple fixed-step gradient algorithm is thus

$$x^{(k+1)} = x^{(k)} - \alpha\nabla F(x^{(k)}) = (\mathrm{id} - \alpha A)x^{(k)} + \alpha b,$$

which is simply the Richardson iteration of the previous subsection.

3.3.2 Steepest descent

The parameter α can also be chosen adaptively, a natural choice being to minimise at each iteration the function $f : \alpha \mapsto F(x^{(k)} + \alpha p^{(k)})$ where $p^{(k)} = b - Ax^{(k)}$.

By composition, the function f is convex, hence the minimum is attained where the derivative vanishes. First we have

$$F(x^{(k)} + \alpha p^{(k)}) = \frac{1}{2}\langle x^{(k)}, Ax^{(k)}\rangle + \frac{\alpha^2}{2}\langle p^{(k)}, Ap^{(k)}\rangle + \alpha\langle p^{(k)}, Ax^{(k)}\rangle - \langle x^{(k)}, b\rangle - \alpha\langle p^{(k)}, b\rangle,$$

thus

$$f'(\alpha) = \alpha\langle p^{(k)}, Ap^{(k)}\rangle + \langle p^{(k)}, Ax^{(k)}\rangle - \langle p^{(k)}, b\rangle.$$

Thus the parameter α_k such that $f'(\alpha_k) = 0$ is given by

$$\alpha_k = \frac{\langle p^{(k)}, b - Ax^{(k)}\rangle}{\langle p^{(k)}, Ap^{(k)}\rangle} = \frac{\langle p^{(k)}, p^{(k)}\rangle}{\langle p^{(k)}, Ap^{(k)}\rangle}. \tag{3.7}$$

Algorithm 3.1 Steepest descent gradient

```
function SteepestDescent(A, b, ε_tol)
    x = 0
    p = b
    while ‖p‖ > ε_tol do
        α = ‖p‖² / ⟨p, Ap⟩
        x = x + αp
        p = p − αAp
    end while
    return x
end function
```

Since A is symmetric, positive-definite, the bilinear form $(x, y) \mapsto \langle x, Ay\rangle$ defines a scalar product. Let us denote the associated norm by $\|\cdot\|_A$. Note that

$$\begin{aligned}\frac{1}{2}\langle x - x_*, A(x - x_*)\rangle &= \frac{1}{2}\langle x, Ax\rangle - \langle x, Ax_*\rangle + \frac{1}{2}\langle x_*, Ax_*\rangle \\ &= \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle + \frac{1}{2}\langle x_*, Ax_*\rangle \\ &= F(x) + \frac{1}{2}\langle x_*, Ax_*\rangle.\end{aligned}$$

Thus minimising F is the same thing as minimising $\|x - x_*\|_A$.

With this observation, we can prove the following theorem on the convergence of the steepest descent algorithm.

Theorem 3.11. *Assume that A is a symmetric, positive-definite matrix. Denote by $(x^{(k)})$ the sequence by Algorithm 3.1. Then we have for all $k \geq 0$*

$$\|x^{(k)} - x_*\|_A \leq \Big(\frac{\mathrm{cond}_2(A) - 1}{\mathrm{cond}_2(A) + 1}\Big)^k \|x^{(0)} - x_*\|. \tag{3.8}$$

Proof. By definition of $x^{(k)}$, recalling that $\alpha_{\mathrm{opt}} = \frac{2}{\lambda_{\min}+\lambda_{\max}}$ and we have

$$\begin{aligned}\|x^{(k)} - x_*\|_A &= \min_{\alpha\in\mathbb{R}} \|x^{(k-1)} - x_* + \alpha p^{(k-1)}\|_A \\ &\leq \|x^{(k-1)} - x_* + \alpha_{\mathrm{opt}} p^{(k-1)}\|_A \\ &\leq \|x^{(k-1)} - x_* + \alpha_{\mathrm{opt}}(b - Ax^{(k-1)})\|_A \\ &\leq \|x^{(k-1)} - x_* + \alpha_{\mathrm{opt}}(Ax_* - Ax^{(k-1)})\|_A \\ &\leq \|(\mathrm{id} - \alpha_{\mathrm{opt}} A)(x^{(k-1)} - x_*)\|_A, \\ &\leq \|A^{1/2}(\mathrm{id} - \alpha_{\mathrm{opt}} A)(x^{(k-1)} - x_*)\|_2 \\ &\leq \|\,\mathrm{id} - \alpha_{\mathrm{opt}} A\|_2 \|x^{(k-1)} - x_*\|_A, \\ &\leq \rho(\mathrm{id} - \alpha_{\mathrm{opt}} A)\|x^{(k-1)} - x_*\|_A,\end{aligned}$$

where we have used that $\rho(\mathrm{id} - \alpha_{\mathrm{opt}} A) = \|\,\mathrm{id} - \alpha_{\mathrm{opt}} A\|_2$ because A is hermitian. The result follows from $\rho(\mathrm{id} - \alpha_{\mathrm{opt}} A) = \frac{\mathrm{cond}_2(A)-1}{\mathrm{cond}_2(A)+1}$. □

Again an ill-conditioned matrix impedes the speed of convergence of the steepest descent algorithm.

3.4 Stationary iterative methods in the XXI$^{\mathrm{st}}$ century

Most of the methods introduced here have been proposed in the XIX$^{\mathrm{th}}$ or in the early XX$^{\mathrm{th}}$ century. Since then, these methods have been superseded by more modern approachs based on the *Krylov methods*, like the conjugate gradient algorithm or GMRES. These have better convergence properties and are numerically more efficient to reach accurate approximation to the solution.

These methods should however not be dismissed as they provide theoretical insights on the convergence theory of iterative methods, and the importance of conditioning in that regard. Finally, stationary iterative methods can be used in combination with Krylov methods.

This is the idea underlying modern numerical linear algebra methods, such as the multigrid method [CG22, Chapter 4], where classical iterative methods are used as a cheap preconditioner.

Chapter 4

Krylov subspace methods

The iterative methods in the previous chapter are only using the knowledge of the previous iterate to build the next one. Instead, it seems preferable to include more directions to improve the approximation of the solution to the linear system $Ax_* = b$. This idea is formalised in the framework of the projection processes and in this setting, we will see that the Krylov subspace methods emerge as a natural candidate for these projection processes. The celebrated conjugate gradient algorithm and GMRES are two instances of projection processes based on Krylov subspaces.

4.1 Projection process

4.1.1 Definition and well-posedness of the projection process

Definition 4.1. *Let $\mathscr{C}_k$ and $\mathscr{S}_k$ be k-dimensional linear subspaces of $\mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$ and $x^{(0)} \in \mathbb{C}^n$. We say that $x^{(k)} \in \mathbb{C}^n$ is the result of a projection process if there exists $z^{(k)} \in \mathscr{S}_k$ such that*

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)} \\ r^{(k)} = b - Ax^{(k)} \perp \mathscr{C}_k. \end{cases} \tag{4.1}$$

We say that the projection process is well-defined if $x^{(k)}$ exists and is uniquely defined.

We immediately notice that $r^{(k)} = r^{(0)} - Az^{(k)}$, hence the condition can also be phrased as $Az^{(k)} \perp r^{(0)} + \mathscr{C}_k$. The goal is to establish natural conditions on $\mathscr{C}_k$ and $\mathscr{S}_k$ under which the projection process is well-defined. We will call $\mathscr{S}_k$ the *search space* and $\mathscr{C}_k$ the *constraint space*.

Proposition 4.2. *Let $(c_1, \dots, c_k)$ (resp. $(s_1, \dots, s_k)$) be a basis of $\mathscr{C}_k$ (resp. $\mathscr{S}_k$) and let $C_k = [c_1; \dots; c_k]$ and $S_k = [s_1; \dots; s_k]$. The projection process is well-defined if and only if $C_k^* A S_k$ is invertible.*

Proof. By definition of the projection process, we can write $x^{(k)} = x^{(0)} + S_k t_k$ for a vector $t_k \in \mathbb{C}^k$. By the orthogonal constraint, we have $r^{(k)} \perp \mathscr{C}_k$, which means that $C_k^* r^{(k)} = 0$. But $r^{(k)} = r^{(0)} - AS_k t_k$, thus we find that t_k solves $C_k^* A S_k t_k = C_k^* r^{(0)}$. t_k is uniquely defined for all $r^{(0)}$ if and only if $C_k^* A S_k$ is invertible. □

Under the assumption that the projection process is well-defined, we can wonder whether there are conditions such that the norm of the residual $r^{(k)}$ is bounded by the norm of the initial one $\|r^{(0)}\|$. We have

$$\begin{aligned} r^{(k)} &= r^{(0)} - AS_k t_k \\ &= r^{(0)} - AS_k(C_k^* AS_k)^{-1} C_k^* r^{(0)} \\ &= \big(\mathrm{id} - AS_k(C_k^* AS_k)^{-1} C_k^*\big) r^{(0)}. \end{aligned}$$

By a simple calculation, we see that the operator $P_k = AS_k(C_k^* AS_k)^{-1} C_k^*$ is a projection. If we require $\|r^{(k)}\| \leq \|r^{(0)}\|$ for any $r^{(0)}$, then we need P_k to be an orthogonal projector[1]. In that case, a possible choice is to take $C_k = AS_k$.

If the matrix A is Hermitian positive-definite, we can also investigate whether there is another choice by looking at the operator norm of the error $x^{(k)} - x_*$:

$$\begin{aligned} \|x^{(k)} - x_*\|_A &= \|A^{1/2}(x^{(k)} - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k t_k - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k(C_k^* AS_k)^{-1} C_k^* r^{(0)} - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k(C_k^* AS_k)^{-1} C_k^*(Ax_* - Ax^{(0)}) - x_*)\| \\ &= \big\|\big(\mathrm{id} - A^{1/2} S_k(C_k^* AS_k)^{-1} C_k^* A^{1/2}\big)(A^{1/2}x^{(0)} - A^{1/2}x_*)\big\|. \end{aligned}$$

The matrix $Q_k = A^{1/2} S_k(C_k^* AS_k)^{-1} C_k^* A^{1/2}$ is also a projection, which is not orthogonal in general. Again a natural choice to ensure that Q_k is an orthogonal projector is to set $C_k = S_k$.

In both cases, we will show that such choices for the constraint space lead to a well-defined projection process.

Proposition 4.3. *Suppose that A is invertible and let $\mathscr{S}_k$ be a k-dimensional subspace of $\mathbb{C}^n$. Then*

1. *if $\mathscr{C}_k = A\mathscr{S}_k$, then the projection process is well-defined;*

2. *if A is Hermitian positive-definite and $\mathscr{C}_k = \mathscr{S}_k$, then the projection process is well-defined.*

Proof. 1. By Proposition 4.2, it is enough to check that $C_k^* AS_k$ is invertible where $C_k = [c_1; \ldots; c_k]$ and $S_k = [s_1; \ldots; s_k]$, for some basis $(c_1, \ldots, c_k)$ (resp. $(s_1, \ldots, s_k)$) of $\mathscr{C}_k$ (resp. $\mathscr{S}_k$). In that case, as A is invertible, $(As_1, \ldots, As_k)$ is a basis of $\mathscr{C}_k$. Thus we have $C_k^* AS_k = S_k^* A^* AS_k$, which is Hermitian. It is also positive-definite as for $y \in \mathbb{C}^k$, we have $\langle y, S_k^* A^* AS_k y\rangle = \|AS_k y\|^2$. Hence $\langle y, S_k^* A^* AS_k y\rangle = 0$ if and only if $AS_k y = 0$, which is equivalent to $y = 0$ as S_k is full-rank and A invertible.

2. Again it suffices to check that $C_k^* AS_k$ is invertible. Let $y \in \mathbb{C}^k$ such that $S_k^* AS_k y = 0$, then $y^* S_k^* AS_k y = 0$, thus $\|S_k y\|_A = 0$. Since S_k is full-rank, $y = 0$.

□

[1] for P a projector, we have the following characterisation: $\|P\| \leq 1 \Leftrightarrow P$ is an orthogonal projector

4.1.2 Krylov subspace methods

Looking at the residual or the norm of the error gives a condition on the constraint space. It remains to see how to wisely pick the search space $\mathscr{S}_k$.

Proposition 4.4. *Suppose that the projection process (4.1) is well-defined. Assume that $r^{(0)} \in \mathscr{S}_k$ and $A\mathscr{S}_k = \mathscr{S}_k$. Then we have $r^{(k)} = 0$.*

This means that the projection process gives the exact solution to the linear system $Ax_* = b$, if $\mathscr{S}_k$ is a stable subspace under A.

Proof. By definition, we have $r^{(k)} = r^{(0)} - Az^{(k)}$, where $z^{(k)} \in \mathscr{S}_k$. Since $r^{(0)} \in \mathscr{S}_k$ and $A\mathscr{S}_k = \mathscr{S}_k$, $r^{(k)} \in A\mathscr{S}_k$. Thus there is $y \in \mathbb{C}^k$ such that $r^{(k)} = AS_k y$. By definition, $r^{(k)} \perp \mathscr{C}_k$, hence $C_k^* r^{(k)} = 0$, so $C_k^* AS_k y = 0$. Because the projection process is well-defined, this means that $C_k^* AS_k$ is invertible hence $y = 0$ and $r^{(k)} = 0$. □

From the previous result, it seems reasonable to start with $\mathscr{S}_1 = \mathrm{Span}(r^{(0)})$ and work with nested sequences spaces $\mathscr{S}_1 \subset \mathscr{S}_2 \subset \dots$. Since we want to find a stable subspace under A, it is natural to introduce the Krylov subspaces.

Definition 4.5. *Let $v \in \mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$ and $k \in \mathbb{N}$. We call $\mathcal{K}_k(A, v) = \mathrm{Span}(v, Av, A^2v, \dots, A^{k-1}v)$ the Krylov subspace.*

Proposition 4.6. *With the notation of Definition 4.5, the following assertions are true*

1. *$\mathcal{K}_k(A, v) \subset \mathcal{K}_{k+1}(A, v)$;*
2. *there is an integer $d \in \mathbb{N}$ such that*

$$\begin{cases} \mathcal{K}_{d+1}(A, v) = \mathcal{K}_d(A, v) \\ \mathcal{K}_{j-1}(A, v) \neq \mathcal{K}_j(A, v), \quad \forall 1 \leq j \leq d \end{cases}$$

   The integer d is called the grade of v with respect to A;

3. *for $j \leq d$, where d is the grade of v with respect to A, $\dim \mathcal{K}_j(A, v) = j$;*
4. *for $j \leq d$, where d is the grade of v with respect to A, $\mathcal{K}_j(A, v) = \{P(A)v,\ P \in \mathbb{C}^{j-1}[X]\}$;*
5. *if A is invertible and d is the grade of v with respect to A, then $A\mathcal{K}_d(A, v) = \mathcal{K}_d(A, v)$.*

Thanks to the last property, the Krylov subspace $\mathcal{K}_k(A, r^{(0)})$ is a good candidate for the search space of the projection process.

Proof. The first four properties are clear. We have $A\mathcal{K}_d(A, v) \subset \mathcal{K}_{d+1}(A, v) = \mathcal{K}_d(A, v)$, so it suffices to show that $\dim A\mathcal{K}_d(A, v) = \dim \mathcal{K}_d(A, v)$ to have the equality. Let $(v_1, \dots, v_d)$ be a basis of $\mathcal{K}_d(A, v)$. As A is invertible, $(Av_1, \dots, Av_d)$ is a free family, thus $\dim A\mathcal{K}_d(A, v) \geq d$, hence $\dim\ A\mathcal{K}_d(A, v) = d = \dim\ \mathcal{K}_d(A, v)$. □

From this study, we have established that Krylov subspaces are good candidates for a search space for the projection process, as they guarantee a termination of the algorithm after a finite number of steps. For the constraint space, based on the norm of the residual or the error, we have identified two possible choices

1. $\mathscr{C}_k = A\mathscr{S}_k$;

2. $\mathscr{C}_k = \mathscr{S}_k$, if A is Hermitian positive-definite.

In both cases, we have already proved that the projection process is well-defined. Taking $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$, the first choice yields the GMRES algorithm and the second choice the conjugate gradient algorithm. We collect all these results and state the mathematical characterisation of both algorithms in the next theorem.

Theorem 4.7. *Let $A \in \mathbb{C}^{n\times n}$ be an invertible matrix, $b \in \mathbb{C}^n$ and $x_* \in \mathbb{C}^n$ be the solution to $Ax_* = b$. Let $x^{(0)} \in \mathbb{C}^n$ and $r^{(0)} = b - Ax^{(0)}$. Assume that $r^{(0)}$ has a grade $d \geq 1$ with respect to A. Let $x^{(k)}$ be defined by the projection process (4.1):*

$$\begin{cases} x^{(k)} & = x^{(0)} + z^{(k)} \text{ where } z^{(k)} \in \mathscr{S}_k \\ r^{(k)} & = b - Ax^{(k)} \perp \mathscr{C}_k. \end{cases}$$

1. *(Characterisation of the conjugate gradient algorithm) If A is Hermitian and positive-definite, $\mathscr{S}_k = \mathscr{C}_k = \mathcal{K}_k(A, r^{(0)})$ for all $1 \leq k \leq d$, then the projection process is well-defined for every $1 \leq k \leq d$ and $x^{(d)} = x_*$. Moreover we have the following characterisation for all iterates $x^{(k)}$, $1 \leq k \leq d$*

$$x^{(k)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}), \quad \text{and} \quad \|x^{(k)} - x_*\|_A = \min_{x \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})} \|x - x_*\|_A. \tag{4.2}$$

2. *(Characterisation of GMRES) If $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$ and $\mathscr{C}_k = A\mathcal{K}_k(A, r^{(0)})$, then the projection process is well-defined for every $1 \leq k \leq d$ and $x^{(d)} = x_*$. Moreover we have the following characterisation for all residuals $r^{(k)}$, $1 \leq k \leq d$*

$$r^{(k)} \perp A\mathcal{K}_k(A, r^{(0)}), \quad \text{and} \quad \|r^{(k)}\| = \min_{x \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})} \|b - Ax\|. \tag{4.3}$$

Proof. The well-posedness is given by Proposition 4.2 and the termination of the projection process is obtained by combining Proposition 4.4 and Proposition 4.6.

We now turn to the mathematical characterisations (4.2) and (4.3). These characterisations rely on rephrasing the orthogonalisation with respect to the constraint space $\mathscr{C}_k$ as an orthogonal projection with respect to a new scalar product.

1. by definition of the projection process, we have $r^{(k)} = b - Ax^{(k)} = A(x_* - x^{(k)}) \perp \mathscr{C}_k = \mathcal{K}_k(A, r^{(0)})$. Hence we have, using that A defines a scalar product

   $$x^{(k)} - x_* \perp A\mathcal{K}_k(A, r^{(0)}) \Leftrightarrow x^{(k)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}) \Leftrightarrow z^{(k)} + x^{(0)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}).$$

   Since $z^{(k)}$ belongs to the subspace $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$, we have that

   $$\|z^{(k)} + x^{(0)} - x_*\|_A = \min_{z \in \mathcal{K}_k(A, r^{(0)})} \|z + x^{(0)} - x_*\|_A,$$

   which is exactly Equation (4.2).

2. by definition of the projection process, we have $r^{(k)} = b - Ax^{(k)} \perp \mathscr{C}_k = A\mathcal{K}_k(A, r^{(0)})$. This is equivalent to $A^*A(x^{(k)} - x_*) \perp \mathcal{K}_k(A, r^{(0)})$. Since A^*A defines a scalar product, we have by the same reasoning as before

$$\|x^{(k)} - x_*\|_{A^*A} = \min_{z \in \mathcal{K}_k(A, r^{(0)})} \|z + x^{(0)} - x_*\|_{A^*A}.$$

Using that for any $y \in \mathbb{C}^n$, $\|y\|^2_{A^*A} = \langle y, A^*Ay\rangle = \|Ay\|^2$, we have (4.3).

□

The mathematical characterisations (4.2) and (4.3) will be central in establishing the practical algorithms and the convergence rates of both algorithms.

4.2 The conjugate gradient algorithm

The CG algorithm is an iterative method to solve $Ax_* = b$ when A is Hermitian positive-definite. This algorithm has several properties that make the algorithm efficient numerically:

- it has a short-term recurrence that makes it cheap to implement;
- it has a well-understood convergence behaviour based on the spectrum of the matrix A.

Such features are not shared with the GMRES algorithm as it will be exposed in Section 4.3.

In order to see that the CG algorithm has a short-term recurrence, it is natural to look at the Gram-Schmidt process to build an orthogonal basis to $\mathcal{K}_k(A, r^{(0)})$. This is the goal of the Arnoldi algorithm.

4.2.1 The Arnoldi algorithm

For this algorithm, we are not going to assume that A is Hermitian positive-definite. We simply require A to be invertible. The Arnoldi algorithm is a slightly modified Gram-Schmidt process for $\mathcal{K}_k(A, v) = \mathrm{Span}(v, Av, \dots, A^{k-1}v)$.

The Arnoldi algorithm breaks down if in the course of the algorithm $h_{j+1,j} = 0$. As we are going to show, it does not happen if $j \leq d$ where d is the grade of v with respect to A.

Proposition 4.8. *Let $v \in \mathbb{C}^n$ be of grade d with respect to A. Then the following assertions are true*

1. *the Arnoldi algorithm 4.1 is well-posed for $k \leq d$ (*i.e. *$h_{j+1,j} \neq 0$ for $j \leq d-1$), moreover for all $j \leq d-1$, $(v_1, \dots, v_j)$ is an orthonormal basis of $\mathcal{K}_j(A, v)$;*
2. *let $V_k = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix} \in \mathbb{C}^{n\times k}$ and let $H_{kk} = \begin{bmatrix} h_{11} & \dots & \dots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & & \\ 0 & & h_{k,k-1} & h_{kk} \end{bmatrix} \in \mathbb{C}^{k\times k}$ then*

$$AV_k = V_k H_{kk} + h_{k+1,k} v_{k+1} e_k^T, \tag{4.4}$$

and

$$V_k^* A V_k = H_{kk}; \tag{4.5}$$

[2] the convention used is $\langle x, y\rangle = \sum_{i=1}^n x_i^* y_i$

Algorithm 4.1 Arnoldi algorithm

function ARNOLDI(A, v, k)
  $v_1 = \frac{v}{\|v\|}$
  for $j = 1, \dots, k$ **do**
    for $i = 1, \dots, j$ **do**
      $h_{ij} = \langle v_i, Av_j \rangle$[2]
    end for
    $\widehat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
    $h_{j+1,j} = \|\widehat{v}_{j+1}\|$
    if $h_{j+1,j} \neq 0$ **then**
      $v_{j+1} = \frac{\widehat{v}_{j+1}}{h_{j+1,j}}$
    end if
  end for
  return $(v_1, \dots, v_k)$
end function

3. *if A is Hermitian, then H_{kk} is tridiagonal with real entries.*

Remark 4.9. *Matrices of the form* $\begin{bmatrix} h_{11} & \dots & \dots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & & \\ 0 & & h_{k,k-1} & h_{kk} \end{bmatrix} \in \mathbb{C}^{k \times k}$ *are called* upper Hessenberg.

Proof. 1. we proceed by contradiction. Let $j < d$ be the integer such that $(v_1, \dots, v_j)$ is an orthonormal basis of $\mathcal{K}_j(A, v)$ but $Av_j \in \mathrm{Span}(v_1, \dots, v_j)$. By construction of $(v_i)_{1 \leq i \leq j}$, $(v_1, \dots, v_{j-1})$ is a basis of $\mathcal{K}_{j-1}(A, v)$, hence v_j has a nonzero component along $A^{j-1}v$, but not $(v_1, \dots, v_{j-1})$. Thus Av_j has a nonzero component along $A^j v \notin \mathrm{Span}(v_1, \dots, v_j) = \mathcal{K}_j(A, v)$, which is a contradiction.

2. by definition of the algorithm, at each step $j \leq d$ we have

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i = V_{j+1} \begin{bmatrix} h_{1j} \\ \vdots \\ h_{j+1,j} \end{bmatrix}.$$

Thus

$$\begin{aligned} AV_k &= V_{k+1} \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1k} \\ h_{21} & h_{22} & \dots & \\ 0 & h_{32} & \dots & \vdots \\ \vdots & & \ddots & \\ 0 & & & h_{k+1,k} \end{bmatrix} \\ &= V_k H_{kk} + h_{k+1,k} v_{k+1} e_k^T. \end{aligned}$$

The second identity (4.5) follows from the orthogonality of $(v_j)_{1 \leq j \leq k+1}$.

3. we have $V_k^* A V_k = H_{kk}$. The matrix H_{kk} is upper Hessenberg and $V_k^* A V_k$ is Hermitian if A is Hermitian, hence H_{kk} is tridiagonal. It remains to show that the entries of H_{kk} are real. We have $h_{j+1,j} = \|\widehat{v}_{j+1}\|$ and $h_{jj} = \langle v_j, A v_j \rangle$, thus the entries are real.

□

The Arnoldi algorithm has a remarkable simplification when A is Hermitian. It is not necessary to reorthogonalise the vectors Av_j against $(v_i)_{1 \leq i \leq j-2}$, by the property above. The resulting algorithm is called the Hermitian Lanczos algorithm.

Algorithm 4.2 Hermitian Lanczos algorithm

```
function HermitianLanczos(A, v, k)
    v_1 = v / ||v||
    for j = 1, ..., k do
        h_jj = ⟨v_j, A v_j⟩
        v̂_{j+1} = A v_j − h_jj v_j − h_{j−1,j} v_{j−1}
        h_{j+1,j} = ||v̂_{j+1}||
        if h_{j+1,j} ≠ 0 then
            v_{j+1} = v̂_{j+1} / h_{j+1,j}
        end if
    end for
    return (v_1, ..., v_k)
end function
```

The three-term recurrence in the Hermitian Lanczos algorithm is the reason why the CG algorithm has also a short term recurrence.

4.2.2 The practical CG algorithm

From the Hermitian Lanczos algorithm, we will at first derive the three-term recurrence of the CG algorithm.

Let $(v_1, \ldots, v_d)$ be the family of orthonormal vectors obtained by the Hermitian Lanczos algorithm applied to $\mathcal{K}_d(A, r^{(0)})$. We are going to exploit the tridiagonal structure of the matrix $T_k = V_k^* A V_k, k = 1, \ldots, d$.

Lemma 4.10. *There exist* $(\mu_1, \ldots, \mu_{d-1}) \in \mathbb{R}^{d-1}$ *and* $(\lambda_1, \ldots, \lambda_d) \in \mathbb{R}^d$ *such that for all* $1 \leq k \leq d$, *we have* $T_k = L_k \Lambda_k L_k^T$ *where* $L_k = \begin{bmatrix} 1 & & & \\ \mu_1 & 1 & & \\ & \ddots & \ddots & \\ & & \mu_{k-1} & 1 \end{bmatrix}$ *and* $\Lambda_k = \operatorname{diag}(\lambda_1, \ldots, \lambda_k)$.

Proof. The matrix T_k is tridiagonal and positive-definite, so it has a unique LU factorisation $T_k = L_k U_k$. Since T_k is Hermitian and invertible, we can factorise the diagonal elements of U_k. Using the uniqueness of the LU factorisation, we have a unique factorisation of T_k of the form

$$T_k = L_k \Lambda_k L_k^T,$$

where $L_k = \begin{bmatrix} 1 & & & \\ \mu_1^{(k)} & 1 & & \\ & \ddots & \ddots & \\ & & \mu_{k-1}^{(k)} & 1 \end{bmatrix}$ and $\Lambda_k = \operatorname{diag}(\lambda_1^{(k)}, \dots, \lambda_k^{(k)})$. It remains to show that $(\lambda_j^{(k)})$ and $(\mu_j^{(k)})$ are independent of k. This is done by noticing that

$$\begin{aligned} T_{k+1} &= \left[\begin{array}{c|c} T_k & \alpha_k \\ \hline \alpha_k & \beta_{k+1} \end{array}\right] = \left[\begin{array}{c|c} L_k & \\ \hline \mu_k & 1 \end{array}\right] \left[\begin{array}{c|c} \Lambda_k & \\ \hline & \lambda_{k+1} \end{array}\right] \left[\begin{array}{c|c} L_k^T & \mu_k \\ \hline & 1 \end{array}\right] \\ &= \begin{bmatrix} L_k \Lambda_k L_k^T & \mu_k L_k \Lambda_k e_k \\ \mu_k e_k^T \Lambda_k L_k & \lambda_{k+1} + \lambda_k \mu_k^2 \end{bmatrix}. \end{aligned}$$

By identification, the claim is proved. □

Proposition 4.11. *Let $r^{(0)}$ be of grade d with respect to A Hermitian positive-definite. Let $(v_1, \dots, v_d)$ be the vectors obtained by the Hermitian Lanczos algorithm. With the notation of Lemma 4.10, there are coefficients $(c_k)_{1 \le k \le d}$ defined iteratively such that the CG iterates $(x^{(k)})_{1 \le k \le d}$ and $(r^{(k)})_{1 \le k \le d}$ are defined by*

$$\begin{cases} \hat{p}_k = v_{k+1} - \mu_k \hat{p}_{k-1} \\ x^{(k)} = x^{(k-1)} + c_k \hat{p}_{k-1} \\ r^{(k)} = r^{(k-1)} - c_k A \hat{p}_{k-1} \end{cases} \tag{4.6}$$

where $\hat{p}_{-1} = 0$ and $c_0^{(0)} = 0$.

Proof. By definition of the CG algorithm, we have

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)}, \quad z^{(k)} \in \mathcal{K}_k(A, r^{(0)}) \\ r^{(k)} = b - A x^{(k)} \perp \mathcal{K}_k(A, r^{(0)}). \end{cases} \tag{4.7}$$

We know that for each $k \le d$, $(v_1, \dots, v_k)$ is a basis of $\mathcal{K}_k(A, r^{(0)})$ so $x^{(k)} = x^{(0)} + V_k t_k$, $t_k \in \mathbb{C}^k$ and $V_k = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix}$. Hence $r^{(k)} = r^{(0)} - A V_k t_k$. By orthogonality, we have $V_k^* r^{(k)} = 0$, thus $V_k^* r^{(0)} - V_k^* A V_k t_k = 0$ and $t_k = (V_k^* A V_k)^{-1} V_k^* r^{(0)}$. Plugging this in $x^{(k)}$ and using Lemma 4.10 yield

$$\begin{aligned} x^{(k)} &= x^{(0)} + V_k (V_k^* A V_k)^{-1} V_k^* r^{(0)} \\ &= x^{(0)} + V_k L_k^{-T} \Lambda_k^{-1} L_k^{-1} V_k^* r^{(0)}. \end{aligned}$$

Let $\widehat{P}_k = V_k L_k^{-T} = \begin{bmatrix} \hat{p}_0, \dots, \hat{p}_{k-1} \end{bmatrix}$. $\widehat{P}_k$ solves $\widehat{P}_k L_k^T = V_k$ so the columns of $\widehat{P}_k$ satisfies

$$\begin{bmatrix} \hat{p}_0, \dots, \hat{p}_{k-1} \end{bmatrix} \begin{bmatrix} 1 & \mu_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \mu_{k-1} \\ & & & 1 \end{bmatrix} = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix}$$

$$\begin{bmatrix} \hat{p}_0, \hat{p}_1 + \mu_1 \hat{p}_0, \dots, \hat{p}_{k-1} + \mu_{k-1} \hat{p}_{k-2} \end{bmatrix} = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix},$$

which is the first item in Equation (4.6).

We have

$$\begin{aligned} x^{(k)} &= x^{(0)} + V_k L_k^{-T} \Lambda_k^{-1} L_k^{-1} V_k^* r^{(0)} \\ &= x^{(0)} + \widehat{P}_k \Lambda_k^{-1} \widehat{P}_k^* r^{(0)} \\ &= x^{(0)} + \begin{bmatrix} \widehat{P}_{k-1}, & \hat{p}_{k-1} \end{bmatrix} \left[\begin{array}{c|c} \Lambda_{k-1}^{-1} & \\ \hline & \lambda_k^{-1} \end{array}\right] \begin{bmatrix} \widehat{P}_{k-1}^* \\ \hat{p}_{k-1}^* \end{bmatrix} r^{(0)} \\ &= x^{(0)} + \widehat{P}_{k-1} \Lambda_{k-1}^{-1} \widehat{P}_{k-1}^* r^{(0)} + \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k} \hat{p}_{k-1} \\ &= x^{(k-1)} + \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k} \hat{p}_{k-1}, \end{aligned}$$

which is the second item in (4.6) with $c_k = \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k}$.

For the last item, we use the definition of $r^{(k)}$ and the expression of $x^{(k)}$

$$r^{(k)} = b - Ax^{(k)} = r^{(k-1)} - c_k A \hat{p}_{k-1}.$$

□

Since the Arnoldi vectors $(v_j)_{1 \le j \le d}$ can be generated using a three-term recurrence, the CG algorithm derived from Equation (4.6) can be rephrased in a three-term recurrence. There is however a way to get rid of the Arnoldi vectors and rewrite the CG algorithm into a two-term recurrence, which is the standard way to implement CG. This is based on the following observations on the vectors $\hat{p}_j$ and $r^{(j)}$.

Remark 4.12. *1. the vectors $(\hat{p}_j)_{0 \le j \le d-1}$ define an A-orthogonal basis (*i.e. $\langle \hat{p}_i, A\hat{p}_j \rangle = 0$ *if $i \neq j$): we have*

$$\widehat{P}_d^* A \widehat{P}_d = L_d^{-1} V_d^* A V_d L_d^{-T} = \Lambda_d.$$

2. we have $r^{(k)} \perp r^{(j)}$ for any $j < k$: by definition of $r^{(j)}$, we have $r^{(j)} = r^{(0)} - Az^{(j)}$ with $z^{(j)} \in \mathcal{K}_j(A, r^{(0)})$, thus $r^{(j)} \in \mathcal{K}_{j+1}(A, r^{(0)})$ but $r^{(k)} \perp \mathcal{K}_k(A, r^{(0)}) \supset \mathcal{K}_{j+1}(A, r^{(0)})$.

3. $\hat{p}_k \in \mathrm{Span}(r^{(k)}, \hat{p}_{k-1})$ for all $0 \le k \le d-1$. Since $\hat{p}_k = v_{k+1} - \mu_{k-1}\hat{p}_{k-1}$, it is sufficient to prove that $r^{(k)}$ and v_{k+1} are colinear: for all $\ell \le k$, we have $(r^{(j)})_{0 \le j \le \ell-1}$ and $(v_j)_{1 \le j \le \ell}$ are orthogonal vectors, that span the same space $\mathcal{K}_k(A, r^{(0)})$, thus necessarily, $r^{(k)}$ and v_{k+1} are colinear for $0 \le k \le d-1$.

The idea is to generate the sequences $(x^{(k)}), (p_k), (r^{(k)})$ of the CG algorithm (we will show that the vectors p_k and $\hat{p}_k$ are colinear) starting with $p_0 = r^{(0)}$ and using that

- $p_k \perp_A p_{k-1}$ with p_k of the form $p_k = r^{(k)} + \omega_k p_{k-1}$;
- $r^{(k)} \perp r^{(k-1)}$ with $r^{(k)} = r^{(k-1)} - \alpha_{k-1} A p_{k-1}$.

For any k, the vectors p_k and $\hat{p}_k$ are colinear, as their initial conditions are colinear and they satisfy the same orthogonality constraints between successive iterates. Using this fact, the sequence of residuals $(r^{(k)})$ defined above and in Proposition 4.11 are equal. The sequence $(x^{(k)})$ defined by $x^{(k)} = x^{(k-1)} - \alpha_{k-1} p_{k-1}$ thus turns out to be also the same as in Proposition 4.11. Quick calculations show that the coefficients in the CG sequence are given by

- $\alpha_{k-1} = \frac{\|r^{(k-1)}\|^2}{\langle r^{(k-1)}, Ap_{k-1}\rangle} = \frac{\|r^{(k-1)}\|^2}{\langle p_{k-1}, Ap_{k-1}\rangle}$ using that $r^{(k-1)} = p_{k-1} - \omega_{k-1}p_{k-2}$ and the A-orthogonality of the p_{k-2} and p_{k-1};
- $\omega_k = -\frac{\langle Ap_{k-1}, r^{(k)}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} = -\frac{\langle r^{(k-1)} - r^{(k)}, r^{(k)}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}\frac{1}{\alpha_{k-1}} = \frac{\|r^{(k)}\|^2}{\|r^{(k-1)}\|^2}$, using the orthogonality of $r^{(k-1)}$ and $r^{(k)}$.

Algorithm 4.3 Conjugate-gradient algorithm

function CG($A, b, x^{(0)}, \varepsilon_{\text{tol}}$)
  $p_0 = r^{(0)} = b - Ax^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\|r^{(k-1)}\|^2}{\langle p_{k-1} Ap_{k-1}\rangle}$
    $x^{(k)} = x^{(k-1)} + \alpha_{k-1}p_{k-1}$
    $r^{(k)} = r^{(k-1)} - \alpha_{k-1}Ap_{k-1}$
    $\omega_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$
    $p_k = r^{(k)} + \omega_k p_{k-1}$
  end while
  return $x^{(k)}$
end function

The cost of implementing the CG algorithm is a single matrix vector multiplication at each step, and the storage of the vectors $x^{(k)}, r^{(k)}$ and p_k.

The properties of the CG sequences $(x^{(k)})$, (p_k) and $(r^{(k)})$ which have been discussed above is summarised in the theorem below.

Theorem 4.13. *Let $A \in \mathbb{C}^{n\times n}$ a Hermitian, positive-definite matrix, $x^{(0)} \in \mathbb{C}^n$ such that $r^{(0)} = b - Ax^{(0)}$ is of grade d with respect to A.*

Then the sequence $(x^{(k)})$ defined by Algorithm 4.3 is the conjugate gradient algorithm characterised by $\|x^{(k)} - x_*\|_A = \min_{z\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|z - x_*\|_A$ where x_* is the solution to $Ax_* = b$.

The algorithm stops after d iterations with the exact solution: $x^{(d)} = x_*$. The family of residuals $(r^{(j)})_{0\le j\le k-1}$ defines an orthogonal basis of $\mathcal{K}_k(A, r^{(0)})$ for each $1 \le k \le d$ and $(p_j)_{0\le j\le k-1}$ is an A-orthogonal basis of $\mathcal{K}_k(A, r^{(0)})$ for each $1 \le k \le d$.

4.2.3 Convergence of the CG algorithm

Using the mathematical characterisation of the CG algorithm, we can estimate the speed of convergence of the CG algorithm.

Recall that $x^{(k)}$ is defined by $\|x^{(k)} - x_*\|_A = \min_{z\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|z - x_*\|_A$. Let $z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})$, by definition of the Krylov space $\mathcal{K}_k(A, r^{(0)})$, we can write

$$z = x^{(0)} + \sum_{i=0}^{k-1} \zeta_i A^i r^{(0)}, \quad (\zeta_i)_{0\le i\le k-1} \in \mathbb{C}^k,$$

thus

$$z - x_* = x^{(0)} - x_* + \sum_{i=0}^{k-1} \zeta_i A^{i+1}(x^{(0)} - x_*) = \phi(A)(x^{(0)} - x_*),$$

where ϕ is a polynomial such that $\phi(0) = 1$ and $\deg \phi \leq k$.

The minimisation problem becomes

$$\begin{aligned}\|x^{(k)} - x_*\|_A &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)(x^{(0)} - x_*)\|_A \\ &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|A^{1/2}\phi(A)(x^{(0)} - x_*)\| \\ &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)A^{1/2}(x^{(0)} - x_*)\| \\ &\leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)\|\|x^{(0)} - x_*\|_A.\end{aligned}$$

Since A is Hermitian and positive-definite, there is a unitary matrix U and a diagonal matrix Λ with positive entries such that $A = U\Lambda U^*$. The matrix norm of $\phi(A)$ is then $\|\phi(A)\| = \max_{1\leq i\leq n} |\phi(\lambda_i)|$. This gives a convergence result for the CG algorithm.

Theorem 4.14. *Let $x^{(k)}$ be the k-th iterate of the CG algorithm with A. Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of A. Then we have*

$$\|x^{(k)} - x_*\|_A \leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\leq i\leq n} |\phi(\lambda_i)| \ \|x^{(0)} - x_*\|_A. \tag{4.8}$$

Remark 4.15. *If $k = n$, by picking ϕ as the Lagrange interpolation polynomial such that $\phi(0) = 1$ and $\phi(\lambda_i) = 0$ for all $1 \leq i \leq n$, we prove that CG stops after n iterations.*

It appears that it is convenient to relax $\max_{1\leq i\leq n} |\phi(\lambda_i)|$ to $\max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)|$ in order to explicit a convergence rate of the CG algorithm.

Corollary 4.16. *Let $x^{(k)}$ be the k-th iterate of the CG algorithm with A and $\mathrm{cond}_2(A)$ the condition number of A with respect to the 2-norm. Then we have*

$$\|x^{(k)} - x_*\|_A \leq 2\Big(\frac{\sqrt{\mathrm{cond}_2(A)} - 1}{\sqrt{\mathrm{cond}_2(A)} + 1}\Big)^k \ \|x^{(0)} - x_*\|_A. \tag{4.9}$$

Proof. We have $\|x^{(k)} - x_*\|_A \leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)| \ \|x^{(0)} - x_*\|_A$ and we use the fact that the min-max problem has an explicit solution given by the rescaled Chebyshev polynomial [3]

$$\chi_k(\lambda) = \frac{T_k\Big(\frac{\lambda_1+\lambda_n-2\lambda}{\lambda_n-\lambda_1}\Big)}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)} = \frac{\cos\Big(k\arccos\Big(\frac{\lambda_1+\lambda_n-2\lambda}{\lambda_n-\lambda_1}\Big)\Big)}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)}.$$

Then

$$\min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)| = \frac{1}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)}.$$

[3] the Chebyshev polynomial of the first kind are defined by $T_k(\cos(\theta)) = \cos(k\theta)$, for $\theta \in [0, \pi]$.

Let $\kappa = \frac{\lambda_n}{\lambda_1} = \mathrm{cond}_2(A)$, then

$$\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1} = \frac{\kappa + 1}{\kappa - 1} = \frac{1}{2}\Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} + \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\Big).$$

We invoke another property[4] of the Chebyshev polynomials T_k

$$T_k\Big(\frac{x + \frac{1}{x}}{2}\Big) = \frac{1}{2}(x^k + x^{-k}), \quad \forall x \in \mathbb{R}.$$

So we deduce

$$T_k\Big(\frac{\kappa + 1}{\kappa - 1}\Big) = \frac{1}{2}\Big(\Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\Big)^k + \Big(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\Big)^k\Big) \geq \frac{1}{2}\Big(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\Big)^k.$$

Thus we obtain

$$\min_{\substack{\phi \in \mathbb{C}^k[X] \\ \phi(0)=1}} \max_{\lambda_1 \leq \lambda \leq \lambda_n} |\phi(\lambda)| \leq 2\Big(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\Big)^k.$$

□

Remark 4.17. *For the steepest gradient algorithm, we had*

$$\|x_{\mathrm{SG}}^{(k)} - x_*\|_A \leq \Big(\frac{\mathrm{cond}_2(A) - 1}{\mathrm{cond}_2(A) + 1}\Big)^k \|x^{(0)} - x_*\|_A.$$

Asymptotically, the convergence rate obtained for the CG algorithm is much better than the one for the steepest gradient, but still sensitive to an ill-conditioned matrix A.

Remark 4.18. *We have proved that for $0 < a < b$, we have*

$$T_m\Big(\frac{a + b}{b - a}\Big) \geq \frac{1}{2}\Big(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\Big),$$

with $\kappa = \frac{b}{a}$.

In the case where A has clustered eigenvalues $0 < \lambda_1 \leq \cdots \leq \lambda_{n-\ell} \ll \lambda_{n-\ell+1} \leq \ldots$, we can improve the previous estimate by considering another relaxation of the min-max problem. For $k \geq \ell$, we can choose $\phi \in \mathbb{C}^k[X], \phi(0) = 1$ as $\phi(\lambda) = q(\lambda)\widetilde{\phi}(\lambda)$, where $\widetilde{\phi}$ is a polynomial of degree at most $k - \ell$ with $\widetilde{\phi}(0) = 1$, and $q(\lambda) = \prod_{i=n-\ell+1}^{n}(1 - \frac{\lambda}{\lambda_i})$ *i.e.* the polynomial of degree ℓ such that $q(0) = 1$ and $q(\lambda_i) = 0$ for $n - \ell + 1 \leq i \leq n$. Now using that $|q(\lambda)| \leq 1$ on $[0, \lambda_{n-\ell+1}]$, we have

$$\begin{aligned}
\min_{\substack{\phi \in \mathbb{C}^k[X] \\ \phi(0)=1}} \max_{1 \leq i \leq n} |\phi(\lambda_i)| &\leq \max_{1 \leq i \leq n} |q(\lambda_i)| \min_{\substack{\widetilde{\phi} \in \mathbb{C}^{k-\ell}[X] \\ \widetilde{\phi}(0)=1}} \max_{1 \leq i \leq n-\ell} |\widetilde{\phi}(\lambda_i)| \\
&\leq \min_{\substack{\widetilde{\phi} \in \mathbb{C}^{k-\ell}[X] \\ \widetilde{\phi}(0)=1}} \max_{\lambda_1 \leq \lambda \leq \lambda_{n-\ell}} |\widetilde{\phi}(\lambda)| \\
&\leq 2\left(\frac{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} - 1}{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} + 1}\right)^{k-\ell}.
\end{aligned}$$

[4] by definition, the equation is true for $x = e^{i\theta}$, thus it extends to any complex number.

The corresponding convergence rate is then

$$\|x^{(k)} - x_*\|_A \leq 2 \left(\frac{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} - 1}{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} + 1} \right)^{k-\ell} \|x^{(0)} - x_*\|_A. \tag{4.10}$$

As $\frac{\lambda_n}{\lambda_1} \gg \frac{\lambda_{n-\ell}}{\lambda_1}$, the previous estimate is much better than (4.9). This explains the good convergence properties of the CG algorithm in practice (see Figure 4.1).

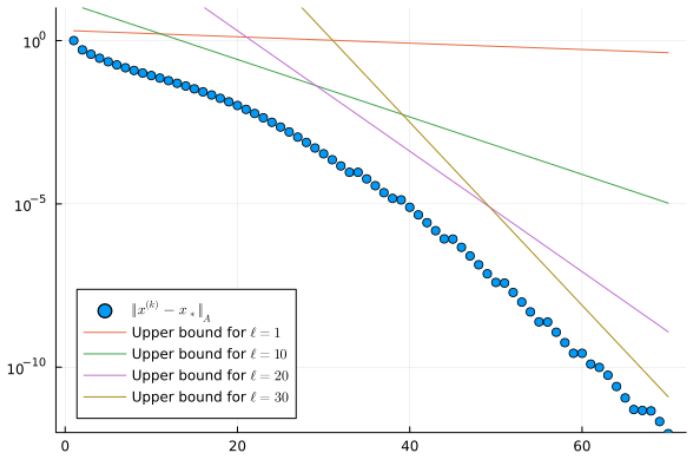


Figure 4.1: CG convergence rate compared to various upper bounds (4.10)

4.2.4 Preconditioned conjugate gradient algorithm

It is often advised to use a preconditioner to solve $Ax_* = b$ to reduce the number of iterations of the solver. A good preconditioner $M \in \mathbb{C}^{n\times n}$ is an invertible matrix such that $\mathrm{cond}_2(M^{-1}A) \ll \mathrm{cond}_2(A)$. Then solving $M^{-1}Ax_* = M^{-1}b$ is significantly easier than the original system. In our case, even if M is Hermitian, positive-definite, $M^{-1}A$ is in general not Hermitian. It is necessary to adapt the CG algorithm in order to incorporate the preconditioner. If we assume that M is Hermitian, positive-definite, we can write the Cholesky decomposition of $M = EE^*$, where $E \in \mathbb{C}^{n\times n}$ is a lower triangular matrix with positive entries. Instead of solving $M^{-1}Ax_* = M^{-1}b$, we can look at the symmetrised system

$$E^{-1}AE^{-*}\tilde{x}_* = E^{-1}b \tag{4.11}$$

Note that we have $x_* = E^{-*}\tilde{x}_*$. For the preconditioned linear system (4.11), the CG algorithm is the following (see Algorithm 4.4).

It is possible to simplify Algorithm 4.4 and get rid of the Cholesky matrices E and E^*. To do so, we are going to work with the variables $x^{(k)} = E^{-*}\tilde{x}^{(k)}$ and $r^{(k)} = E\tilde{r}^{(k)}$ and introduce a new variable $d_k = E^{-*}\tilde{p}_k$. Note that since $\tilde{r}^{(k)} = E^{-1}b - E^{-1}AE^{-*}\tilde{x}^{(k)}$, we have that $r^{(k)} = b - AE^{-*}\tilde{x}^{(k)} = b - Ax^{(k)}$.

We now reexpress the quantities appearing in the transformed CG algorithm 4.4 in the variables $x^{(k)}$, $r^{(k)}$ and d_k:

- $\|\tilde{r}^{(k)}\|^2 = \langle \tilde{r}^{(k)}, \tilde{r}^{(k)} \rangle = \langle E^{-1}r^{(k)}, E^{-1}r^{(k)} \rangle = \langle r^{(k)}, E^{-*}E^{-1}r^{(k)} \rangle = \langle r^{(k)}, M^{-1}r^{(k)} \rangle$

Algorithm 4.4 Transformed conjugate-gradient algorithm

function TCG($A, b, \tilde{x}^{(0)}, \varepsilon_{\text{tol}}, E$)
  $\tilde{p}_0 = \tilde{r}^{(0)} = E^{-1}b - E^{-1}AE^{-*}\tilde{x}^{(0)}$, $k = 0$
  while $\|\tilde{r}^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\|\tilde{r}^{(k-1)}\|^2}{\langle \tilde{p}_{k-1}, E^{-1}AE^{-*}\tilde{p}_{k-1}\rangle}$
    $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \alpha_{k-1}\tilde{p}_{k-1}$
    $\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_{k-1}E^{-1}AE^{-*}\tilde{p}_{k-1}$
    $\omega_k = \frac{\|\tilde{r}_k\|^2}{\|\tilde{r}_{k-1}\|^2}$
    $\tilde{p}_k = \tilde{r}^{(k)} + \omega_k\tilde{p}_{k-1}$
  end while
  return $E^{-*}\tilde{x}^{(k)}$
end function

- $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \alpha_{k-1}\tilde{p}_{k-1} \Leftrightarrow x^{(k)} = x^{(k-1)} + \alpha_{k-1}E^{-*}\tilde{p}_{k-1} = x^{(k-1)} + \alpha_{k-1}d_{k-1}$
- $\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_{k-1}E^{-1}AE^{-*}\tilde{p}_{k-1} \Leftrightarrow r^{(k)} = r^{(k-1)} - \alpha_{k-1}AE^{-*}\tilde{p}_{k-1} = r^{(k-1)} - \alpha_{k-1}Ad_{k-1}$
- $\tilde{p}_k = \tilde{r}^{(k)} + \omega_k\tilde{p}_{k-1} \Leftrightarrow d_k = E^{-*}\tilde{r}^{(k)} + \omega_k d_{k-1} = M^{-1}r^{(k)} + \omega_k d_{k-1}$.

It is thus possible to rewrite Algorithm 4.4 without E or E^*.

Algorithm 4.5 Preconditioned conjugate-gradient algorithm

function PCG($A, b, x^{(0)}, \varepsilon_{\text{tol}}, M$)
  $r^{(0)} = b - Ax^{(0)}$, $d_0 = M^{-1}r^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\langle r^{(k-1)}, M^{-1}r^{(k-1)}\rangle}{\langle d_{k-1}Ad_{k-1}\rangle}$
    $x^{(k)} = x^{(k-1)} + \alpha_{k-1}d_{k-1}$
    $r^{(k)} = r^{(k-1)} - \alpha_{k-1}Ad_{k-1}$
    $\omega_k = \frac{\langle r^{(k)}, M^{-1}r^{(k)}\rangle}{\langle r^{(k-1)}, M^{-1}r^{(k-1)}\rangle}$
    $d_k = M^{-1}r^{(k)} + \omega_k d_{k-1}$
  end while
  return $x^{(k)}$
end function

Compared to the CG algorithm, we need an additional linear solve of the system $My = r^{(k)}$ at each step of the preconditioned CG algorithm. Usually M has a simple structure (*i.e.* diagonal or block-diagonal) such that the linear solve is cheap compared to the total cost of the preconditioned CG algorithm.

Remark 4.19. *We can check that the iterates that are produced by the preconditioned CG algorithm satisfy:*

- $(r^{(k)})$ *are* M^{-1}*-orthogonal,* i.e. $\forall\, i \neq j, \langle r^{(i)}, M^{-1}r^{(j)}\rangle = 0$*;*
- (d_k) *are* A*-orthogonal,* i.e. $\forall\, i \neq j, \langle d_i, Ad_j\rangle = 0$.

4.2.5 Conjugate gradient algorithm in the XXI$^{\text{st}}$ century

The CG algorithm has become the reference method to solve linear problems with Hermitian positive-definite matrices, as it combines all the advantages of a numerical method. The numerical convergence is fast, and is fully understood from a theoretical point of view. The algorithm is numerically stable, and requires minimal memory. Finally it is straightforward to use any preconditioner with CG.

4.3 GMRES

The generalised minimal residual (GMRES) algorithm is a popular iterative method to solve the linear system $Ax_* = b$ when A is invertible and non-Hermitian.

Contrary to the CG algorithm studied previously, GMRES does not have a short-term recurrence. This stems from the fact that we do not have the simplification of the Arnoldi algorithm for general matrices.

4.3.1 The mathematical characterisation and the minimisation problem

Recall that GMRES is mathematically characterised by

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)}, \quad z^{(k)} \in \mathcal{K}_k(A, r^{(0)}) \\ r^{(k)} = b - Ax^{(k)} \perp A\mathcal{K}_k(A, r^{(0)}), \end{cases}$$

or equivalently

$$\|r^{(k)}\| = \min_{x \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})} \|b - Ax\|. \tag{4.12}$$

Let $(v_1, \dots, v_k)$ be the Arnoldi vectors forming an orthonormal basis of $\mathcal{K}_k(A, r^{(0)})$ and satisfying

$$\begin{cases} AV_k = V_{k+1}\underline{H}_{kk} \\ v_1 = \dfrac{r^{(0)}}{\|r^{(0)}\|}, \end{cases}$$

with

$$V_k = [v_1, \dots, v_k], \quad \text{and} \quad \underline{H}_{kk} = \begin{bmatrix} h_{11} & \dots & & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}.$$

A vector $z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})$ can be expressed as $z = x^{(0)} + V_k t_k$, for some $t_k \in \mathbb{C}^k$. The minimisation problem (4.12) becomes

$$\begin{aligned} \|r^{(k)}\| &= \min_{t_k \in \mathbb{C}^k} \|r^{(0)} - AV_k t_k\| \\ &= \min_{t_k \in \mathbb{C}^k} \|\|r^{(0)}\| V_{k+1} e_1 - V_{k+1}\underline{H}_{kk} t_k\| \\ &= \min_{t_k \in \mathbb{C}^k} \|V_{k+1}(\|r^{(0)}\| e_1 - \underline{H}_{kk} t_k)\| \\ &= \min_{t_k \in \mathbb{C}^k} \|\|r^{(0)}\| e_1 - \underline{H}_{kk} t_k\|, \end{aligned} \tag{4.13}$$

where we have used that V_{k+1} has orthonormal columns. The last equation is a mean square minimisation problem. The standard way to solve such a problem is to write the so-called *QR factorisation* of $\underline{H}_{kk}$.

4.3.2 The GMRES algorithm

We are now solving the minimisation (4.13) by using a QR factorisation of $\underline{H}_{kk} = Q_k R_k$, $Q_k \in \mathbb{C}^{(k+1)\times(k+1)}$ and $R_k \in \mathbb{C}^{(k+1)\times k}$. In this case, using Proposition 1.14, Equation (4.13) becomes

$$\min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - \underline{H}_{kk}t_k\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - Q_k R_k t_k\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|Q_k^* e_1 - R_k t_k\|.$$

R_k is upper triangular, so we have $R_k = \begin{bmatrix}\widetilde{R}_k \\ 0\end{bmatrix}$ and denoting $\|r^{(0)}\|Q_k^* e_1 = \begin{bmatrix} g_k \\ \gamma_{k+1}\end{bmatrix}$ with $g_k \in \mathbb{C}^k, \gamma_{k+1} \in \mathbb{C}$, we see that $t_k \in \mathbb{C}^k$ solves $\widetilde{R}_k t_k = g_k$. Moreover we have

$$\|r^{(k)}\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - \underline{H}_{kk}t_k\| = |\gamma_{k+1}|. \tag{4.14}$$

It remains to implement efficiently a QR factorisation of $\underline{H}_{kk}$. To this end, we are going to use the fact that $\underline{H}_{kk}$ is an upper-Hessenberg matrix and that we can do a simple update of the QR factorisation of $\underline{H}_{k-1,k-1}$. Indeed we have

$$\underline{H}_{kk} = \begin{bmatrix}\underline{H}_{k-1,k-1} & h^{(k)} \\ 0 & h_{k+1,k}\end{bmatrix} = \begin{bmatrix}Q_{k-1}R_{k-1} & h^{(k)} \\ 0 & h_{k+1,k}\end{bmatrix},$$

where $h_{k+1,k} \in \mathbb{R}$ (see Proposition 4.8) and $h^{(k)} \in \mathbb{C}^k$. Consider Q_k defined by

$$Q_k = \begin{bmatrix}Q_{k-1} & 0 \\ 0 & 1\end{bmatrix} \Omega_k, \tag{4.15}$$

where $\Omega_k \in \mathbb{C}^{(k+1)\times(k+1)}$ is some unitary matrix fixed later. Then we have

$$\Omega_k^* Q_k^* \underline{H}_{kk} = \Omega_k^* \begin{bmatrix}R_{k-1} & Q_{k-1}^* h^{(k)} \\ 0 & h_{k+1,k}\end{bmatrix}.$$

We simply need Ω_k^* to cancel the $(k+1,k)$ entry of the above matrix. Let $\tilde{h}^{(k)} = Q_{k-1}^* h^{(k)} \in \mathbb{C}^k$, and let

$$\Omega_k^* = \begin{bmatrix}\mathrm{id}_{k-1} & & \\ & c_k^* & s_k \\ & -s_k & c_k,\end{bmatrix} \tag{4.16}$$

with

$$c_k^* = \frac{(\tilde{h}_k^{(k)})^*}{\sqrt{|h_k^{(k)}|^2 + h_{k+1,k}^2}}, \quad \text{and} \quad s_k = \frac{h_{k+1,k}}{\sqrt{|h_k^{(k)}|^2 + h_{k+1,k}^2}}. \tag{4.17}$$

By a matrix multiplication, we can check that $R_k = \Omega_k^* Q_k^* \underline{H}_{kk}$ is upper triangular, and R_k is given by

$$R_k = \begin{bmatrix}R_{k-1} & \tilde{h}_{1:k-1}^{(k)} \\ & 1 \\ 0 & 0\end{bmatrix} \in \mathbb{C}^{(k+1)\times k}. \tag{4.18}$$

Algorithm 4.6 GMRES

function GMRES($A, b, x^{(0)}, \varepsilon_{\text{tol}}$)
  $r^{(0)} = b - Ax^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    Compute v_k of the Arnoldi algorithm 4.1 for A with $v = r^{(0)}$
    Update Q_k according to Eq. (4.15), (4.16) and (4.17)
    Compute $\begin{bmatrix} g_k \\ \gamma_{k+1} \end{bmatrix} = \|r^{(0)}\| Q_k^* e_1$
    Set $\|r^{(k)}\| = |\gamma_{k+1}|$
  end while
  Compute $t_k = \widetilde{R}_k^{-1} g_k$, where $R_k = \begin{bmatrix} \widetilde{R}_k \\ 0 \end{bmatrix}$ and R_k given by Eq. (4.18)
  return $x^{(0)} + V_k t_k$
end function

We are now in position to write the GMRES algorithm 4.6.

Note that in GMRES, only the last approximation $x^{(k)}$ to the solution x_* to the linear equation is computed. Indeed, at each iteration, we just need to estimate the residual which is given by Equation (4.14). Concerning the cost of GMRES, at each step, one step of Arnoldi algorithm has to be performed, which costs one matrix-vector multiplication, and k scalar products. The matrix Q_k needs to be updated, but this cost is negligeable. However, in terms of storage cost, all the Arnoldi vectors $(v_1, \dots, v_k)$ have to be kept for each iteration. This is a serious limitation to the algorithm and in practice, a full GMRES by keeping all the Arnoldi vectors is not advisable, especially if the convergence is slow.

4.3.3 Restarted GMRES

The idea is to limit the number of Arnoldi vectors to K and restart a GMRES run from the latest GMRES iteration.

The storage cost of the restarted GMRES scales as the number of Arnoldi vectors stored at each step of the algorithm. Note that contrary to GMRES, we have no guarantee that the algorithm converges after a finite number of iterations, although the residuals are still nonincreasing, due to the mathematical characterisation of GMRES (4.3).

Remark 4.20. *Let $A \in \mathbb{C}^{n\times n}$ be given by*

$$A = \begin{bmatrix} 0 & \dots & 0 & \alpha_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & \alpha_{n-2} \\ & & 1 & \alpha_{n-1} \end{bmatrix}, \tag{4.19}$$

where $(\alpha_0, \dots, \alpha_{n-1}) \in \mathbb{C}^n$. The characteristic polynomial of A is given by $P(\lambda) = \det(\lambda \,\mathrm{id} - A) = \lambda^n - \sum_{k=0}^{n-1} \alpha_k \lambda^k$, thus the coefficients can be chosen to have any eigenvalue distribution. Then for any $b \in \mathbb{C}^n$, the restarted GMRES with $x^{(0)} = A^{-1}(b - e_1)$ does not converge, except if $K = n$. In fact, the residual is constant $r^{(k)} = r^{(0)}$ for $k \le n - 1$. This shows that there is

Algorithm 4.7 Restarted GMRES(K)

function GMRES($A, b, x^{(0)}, \varepsilon_{\text{tol}}, K$)
  $r^{(0)} = b - Ax^{(0)}$
  while $\|r^{(0)}\| > \varepsilon_{\text{tol}}$ **do**
    Compute $(v_1, \dots, v_K)$ the Arnoldi vectors of Algorithm 4.1 for A with $v = r^{(0)}$
    Update Q_K according to Eq. (4.15),(4.16) and (4.17)
    Compute $\begin{bmatrix} g_K \\ \gamma_{K+1} \end{bmatrix} = \|r^{(0)}\| Q_K^* e_1$
    Set $\|r^{(0)}\| = |\gamma_{K+1}|$
    Compute $t_K = \widetilde{R}_K^{-1} g_K$, where $R_K = \begin{bmatrix} \widetilde{R}_K \\ 0 \end{bmatrix}$ and R_K given by Eq. (4.18)
    $x^{(0)} = x^{(0)} + V_K t_K$
  end while
  return $x^{(0)}$
end function

no hope to give an accurate characterisation of the convergence of GMRES solely based on the spectrum of the matrix.

In practice, it is customary to take $K = 20$ and in general, a larger K improves the convergence of the restarted GMRES algorithm.

Remark 4.21. *The latter comment is a general advice but counterexamples exist to this rule of thumb. Let* $A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ *and* $b = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}$*, then for* $x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$*, restarted GMRES converges after three steps for* $K = 1$ *but does not converge for* $K = 2$*.*

4.3.4 Convergence of GMRES

Convergence results on GMRES are less powerful than for the CG algorithm. An attempt consists in following the same steps as in the convergence estimate of the CG algorithm:

$$\begin{aligned} \|r^{(k)}\| &= \min_{z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})} \|b - Az\| \\ &= \min_{\tilde{z} \in \mathcal{K}_k(A, r^{(0)})} \|r^{(0)} - A\tilde{z}\| \\ &= \min_{\substack{\phi \in \mathbb{C}^k[X] \\ \phi(0)=1}} \|\phi(A) r^{(0)}\|. \end{aligned}$$

Since A is no longer Hermitian, we have to resort to another decomposition of the matrix A, namely the Jordan decomposition which is recalled in the next proposition.

Proposition 4.22. *Let* $A \in \mathbb{C}^{n \times n}$ *and* $(\lambda_1, \dots, \lambda_r)$ *be the distinct eigenvalues of* A*. For* $1 \le \ell \le r$*, let* $J_{\lambda_\ell} = \begin{bmatrix} \lambda_\ell & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_\ell \end{bmatrix} \in \mathbb{C}^{n_\ell \times n_\ell}$ *be the Jordan blocks, where* $\sum_{\ell=1}^{r} n_\ell = n$*.*

Then there exists $Y \in \mathbb{C}^{n\times n}$ invertible such that

$$A = Y \begin{bmatrix} J_{\lambda_1} & & \\ & \ddots & \\ & & J_{\lambda_r} \end{bmatrix} Y^{-1}. \tag{4.20}$$

This is the Jordan decomposition of A *and if the columns of Y are of norm 1, it is unique up to the permutations of the Jordan blocks and rotations in the Jordan blocks.*

Using the Jordan decomposition of A, we have

$$\begin{aligned}\|r^{(k)}\| &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \left\| Y \begin{bmatrix} \phi(J_{\lambda_1}) & & \\ & \ddots & \\ & & \phi(J_{\lambda_r}) \end{bmatrix} Y^{-1} r^{(0)} \right\| \\ &\leq \|Y\|\|Y^{-1}\|\|r^{(0)}\| \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\leq\ell\leq r} \|\phi(J_{\lambda_\ell})\|.\end{aligned}$$

Proposition 4.23. *Let $A = Y \begin{bmatrix} J_{\lambda_1} & & \\ & \ddots & \\ & & J_{\lambda_r} \end{bmatrix} Y^{-1}$ be a Jordan decomposition of A, and $r^{(k)}$ be the k-th residual of the GMRES algorithm 4.6. Then*

$$\|r^{(k)}\| \leq \|Y\|\|Y^{-1}\|\|r^{(0)}\| \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\leq\ell\leq r} \|\phi(J_{\lambda_\ell})\|.$$

If Y is ill-conditioned, the bound given is meaningless. Consider the matrix $A = \text{tridiag}(-\alpha, \alpha, -\frac{1}{\alpha}) \in \mathbb{R}^{n\times n}$, with $\alpha \in \mathbb{R}, \alpha > 1$. The eigenvalues of A are $\alpha - 2\cos\left(\frac{j\pi}{n+1}\right)$ for $1 \leq j \leq n$ and the associated eigenvectors are $y_j = \frac{Dz_j}{\|Dz_j\|}$ for $1 \leq j \leq n$ where (z_j) is some orthonormal basis of $\mathbb{R}^n$ and $D = \text{diag}(\alpha, \ldots, \alpha^n)$. The conditioning of Y then scales as α^n, but $\|r^{(k)}\| \leq \|r^{(0)}\|$.

4.3.5 Beyond GMRES?

The bottleneck of GMRES is the Arnoldi algorithm and in particular, the absence of a short recurrence in the Arnoldi algorithm for a general matrix A.

A natural question to ask is whether the Arnoldi algorithm is a good starting point to derive a short-term iterative linear solver. In other words, for a given matrix A, is there an $(s+1)$-term recurrence of the form

$$\begin{cases} x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)} \\ p^{(k)} = Ap^{(k-1)} - \displaystyle\sum_{j=0}^{s-2} \beta_{k-1,j} p^{(k-1-j)}, \end{cases} \tag{4.21}$$

with $x^{(0)} \in \mathbb{C}^n$ and $p^{(0)} = r^{(0)} = b - Ax^{(0)}$ which stops after $m \leq n$ iterations at the exact solution x_* to $Ax_* = b$, for a well-chosen set of coefficients $(\alpha_k)_{0\leq k\leq m-1}$ and $(\beta_{k-1,j})_{\substack{0\leq k\leq m-1\\ 0\leq j\leq s-2}}$?

By Eq. (4.21), the error $e_k = x^{(k)} - x_*$ satisfies $e_k = e_{k-1} + \alpha_{k-1} p^{(k-1)}$. Hence by iteration,

$$e_k \in e_0 + \mathrm{Span}(p^{(0)}, \dots, p^{(k-1)}).$$

The error e_k is thus minimised when $e_k \perp \mathrm{Span}(p^{(0)}, \dots, p^{(k-1)})$, thus $\langle e_k, p^{(j)} \rangle = 0$ for all $0 \leq j \leq k-1$. This gives in particular for $j = k-1$

$$\alpha_{k-1} = \frac{\langle p^{(k-1)}, e_{k-1} \rangle}{\langle p^{(k-1)}, p^{(k-1)} \rangle},$$

and for $0 \leq j \leq k-2$, we have

$$0 = \langle p^{(j)}, e_k \rangle = \langle p^{(j)}, e_{k-1} \rangle + \alpha_{k-1} \langle p^{(j)}, p^{(k-1)} \rangle = \alpha_{k-1} \langle p^{(j)}, p^{(k-1)} \rangle.$$

This means that if $\alpha_{k-1} \neq 0$, we can enforce the orthogonality of the vectors (p^{k-1-j}) for $0 \leq j \leq s-2$ by setting the coefficients $(\beta_{k-1,j})_{\substack{0 \leq k \leq m-1 \\ 0 \leq j \leq s-2}}$ such thatt

$$\beta_{k-1,j} = \frac{\langle p^{(k-1-j)}, A p^{(k-1)} \rangle}{\langle p^{(k-1-j)}, p^{(k-1-j)} \rangle}.$$

This motivates to restrict the search of a short-term iterative linear solver to the following class of sequences.

Definition 4.24. *The sequences $(x^{(k)}), (p^{(k)})$ defined by Equation (4.21) such that for any $b \in \mathbb{C}^n$, there is $m \leq n$ such that $x^{(m)}$ solves $Ax^{(m)} = b$ and $\langle p^{(k-1-j)}, p^{(k)} \rangle = 0$ for $0 \leq k \leq m, 0 \leq j \leq s-2$ are called an $(s+1)$-term CG method.*

Notice in the definition of an $(s+1)$-term CG sequence, the vectors $(p^{(k)})_{0 \leq k \leq m}$ are not necessarily all two-by-two orthogonal.

Remark 4.25. *The CG sequence defined in Theorem 4.13 is a (3+1)-CG sequence for Hermitian positive definite matrices. Indeed we have that $x^{(k)} = x^{(k-1)} + \alpha_{k-1} p_{k-1}$ and*

$$\begin{aligned} p_k &= r^{(k)} + \omega_k p_{k-1} = b - Ax^{(k)} + \omega_k p_{k-1} = r^{(k-1)} - \alpha_{k-1} A p_{k-1} + \omega_k p_{k-1} \\ &= p_{k-1} - \omega_{k-1} p_{k-2} + \alpha_{k-1} A p_{k-1} + \omega_k p_{k-1}. \end{aligned}$$

Moreover, $(x^{(k)})$ converges in at most n iterations to x_* and $p_k \perp Ap_j$ for all $j \neq k$. Hence up to a rescaling of the vectors (p_k), the CG sequence is an $(s+1)$-term CG sequence, for $s = 3$, and with the scalar product defined by A.

We have a characterisation of the class of matrices which have an $(s+1)$-term CG method.

Theorem 4.26 (Faber, Manteuffel (1984))**.** *An $(s+1)$-term CG method exists for the matrix A if and only if either*

1. *the minimal polynomial of A has degree less than s;*
2. *A^* is a polynomial of degree less than $s-2$ in A.*

Notice that for $s = 3$ and assuming additionally that A is Hermitian positive-definite, the CG sequence defined in Algorithm 4.3 is an example of an $(s+1)$-term CG sequence.

Proof. We are only going to prove the converse, as the proof of the implication is quite involved.

First, assume that the minimal polynomial of A has degree less than s. Let $(x^{(k)}), (p^{(k)})$ be the sequences defined by Eq. (4.21). By induction, we see that for all $1 \leq k \leq s$, $(p^{(0)}, \dots, p^{(k-1)})$ is a basis of $\mathrm{Span}(r^{(0)}, Ar^{(0)}, \dots, A^{k-1}r^{(0)})$. By induction, we also have that $x^{(k)} = x^{(0)} + \sum_{j=0}^{k-1} \alpha_j p^{(j)}$, thus we can choose $(\alpha_j)_{0\leq j\leq s-1}$ and $(\beta_{k,j})_{\substack{0\leq k\leq s-1\\ 0\leq j\leq s-2}}$ such that

$$\begin{cases} \|b - Ax^{(k)}\| = \min\limits_{z^{(k)} \in \mathrm{Span}(p^{(0)},\dots,p^{(k-1)})} \|r^{(0)} - Az^{(k)}\| \\ p^{(k)} \perp \mathrm{Span}(p^{(k-s+1)}, \dots, p^{(k-1)}). \end{cases}$$

In particular for $k = s$, we have that

$$\begin{aligned} \|b - Ax^{(s)}\| &= \min_{z^{(s)} \in \mathrm{Span}(p^{(0)},\dots,p^{(s-1)})} \|r^{(0)} - Az^{(s)}\| \\ &= \min_{\phi \in \mathbb{C}^{s-1}[X]} \|r^{(0)} - \phi(A)r^{(0)}\| \\ &= \min_{\substack{\phi \in \mathbb{C}^{s}[X] \\ \phi(0)=1}} \|\phi(A)r^{(0)}\|. \end{aligned}$$

Choosing ϕ as the minimal polynomial such that $\phi(0) = 1$ shows that $Ax^{(s)} = b$.

Now assume that A^* is a polynomial of degree less than $s-2$ of A. We will first prove by induction that if $(p^{(j)})_{0\leq j\leq k}$ are orthogonal vectors, then $(p^{(j)})_{0\leq j\leq k+1}$ are also orthogonal.

For the initialisation, we know that for $k \leq s-1$, choosing $(\beta_{k-1,j})_{0\leq j\leq s-2}$ such that

$$\beta_{k-1,j} = \frac{\langle p^{(j)}, Ap^{(k-1)}\rangle}{\langle p^{(k-1-j)}, p^{(k-1-j)}\rangle}$$

ensures that $\langle p^{(i)}, p^{(j)}\rangle = 0$ for $0 \leq i \neq j \leq s-1$.

For the induction step, let us assume that $(p^{(j)})_{0\leq j\leq k}$ are orthogonal vectors. We already know that by setting $(\beta_{k,j})_{0\leq j\leq s-2}$ such that

$$\beta_{k,j} = \frac{\langle p^{(k-j)}, Ap^{(k)}\rangle}{\langle p^{(k-j)}, p^{(k-j)}\rangle},$$

we have that $p^{(k+1)} \perp \mathrm{Span}(p^{(k-s+2)}, \dots, p^{(k)})$. For $i \leq k-s+1$, we have from Eq. (4.21)

$$\langle p^{(i)}, p^{(k+1)}\rangle = \langle p^{(i)}, Ap^{(k)}\rangle - \sum_{j=0}^{s-2} \beta_{k,j}\langle p^{(i)}, p^{(k-j)}\rangle.$$

By the induction assumption, $\langle p^{(i)}, p^{(k-j)}\rangle = 0$ for $0 \leq j \leq s-2$. Now $\langle p^{(i)}, Ap^{(k)}\rangle = \langle A^*p^{(i)}, p^{(k)}\rangle$. By assumption, $A^* = q_{s-2}(A)$ for some polynomial $q_{s-2} \in \mathbb{C}^{s-2}[X]$. Thus $A^*p^{(i)} = q_{s-2}(A)p^{(i)} \in \mathrm{Span}(p^{(i)}, \dots, p^{(i+s-2)})$. But $i+s-2 \leq k-1$, thus using the induction assumption $\langle q_{s-2}(A)p^{(i)}, p^{(k)}\rangle = 0$. This shows that $\langle p^{(i)}, p^{(k+1)}\rangle = 0$.

Notice that $\mathrm{Span}(p^{(0)}, \dots, p^{(k)}) \subset \mathrm{Span}(r^{(0)}, \dots, A^k r^{(0)})$ but by orthogonality of the vectors $(p^{(0)}, \dots, p^{(k)})$, we deduce that $\mathrm{Span}(p^{(0)}, \dots, p^{(k)}) = \mathrm{Span}(r^{(0)}, \dots, A^k r^{(0)})$. Since $x^{(k)} = x^{(0)} + \sum_{i=0}^{k-1} \alpha_i p^{(i)}$, by selecting $\alpha_i = -\langle p^{(i)}, x^{(0)} - x_*\rangle$, we have that $(x^{(k)})$ converges to x_* in at most n steps. This finishes the proof. □

The Faber-Manteuffel theorem essentially states that for a general matrix A, there is no equivalent to the conjugate-gradient algorithm solely based on the Krylov space $\mathcal{K}_k(A, r^{(0)})$. This means that we need to look for another construction to define a short-term recurrence for the iterative method.

Going back to the essence of the conjugate-gradient and GMRES methods, it is reasonable to look for alternatives to the Arnoldi/Lanczos algorithms that would give a short-term recurrence for any matrix.

One way to achieve such a short recurrence is to relax the orthogonality constraint on the Arnoldi vectors $(v_1, \dots, v_k)$ while remaining a basis of $\mathcal{K}_k(A, v)$. The new constraint is to build a family of vectors $(w_1, \dots, w_k)$ which is a basis of $\mathcal{K}_k(A^*, w)$ and also a dual family to $(v_1, \dots, v_k)$, *i.e.* $\forall 1 \leq i, j \leq k, \langle w_i, v_j \rangle = \delta_{ij}$. This gives the non-Hermitian Lanczos algorithm 4.8.

Algorithm 4.8 Non-Hermitian Lanczos algorithm

function NonHermitianLanczos(A, v, w, k)
  $v_0 = w_0 = 0$, $\beta_1 = \delta_1 = 0$
  $v_1 = \frac{v}{\|v\|}$, $w_1 = \frac{w}{\langle v_1, w \rangle}$
  for $j = 1, \dots, k$ **do**
    $\gamma_j = \langle w_j, A v_j \rangle$
    $\widehat{v}_{j+1} = A v_j - \gamma_j v_j - \beta_j v_{j-1}$
    $\widehat{w}_{j+1} = A^* w_j - \gamma_j^* w_j - \delta_j w_{j-1}$
    $\delta_{j+1} = \|\widehat{v}_{j+1}\|$
    if $\delta_{j+1} = 0$ **then**
      Stop
    end if
    $\beta_{j+1} = \langle v_{j+1}, \hat{w}_{j+1} \rangle$
    if $\beta_{j+1} = 0$ **then**
      Stop
    end if
    $w_{j+1} = \frac{\widehat{w}_{j+1}}{\beta_{j+1}}$
  end for
  return $(v_1, \dots, v_k), (w_1, \dots, w_k)$
end function

For the non-Hermitian Lanczos vectors, we have the following properties.

Proposition 4.27. *If there is no breakdown in Algorithm 4.8,* i.e. *for all* $1 \leq j \leq k-1$ $\widehat{v}_{j+1} \neq 0, \widehat{w}_{j+1} \neq 0, \beta_{j+1} \neq 0$, *then we have*

$$\begin{cases} A V_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T \\ A^* W_k = W_k T_k + \beta_{k+1} w_{k+1} e_k^T \\ W_k^* A V_k = T_k \\ W_k^* V_k = \mathrm{id}_k, \end{cases} \tag{4.22}$$

where $V_k = \begin{bmatrix} v_1 & \dots & v_k \end{bmatrix}$, $W_k = \begin{bmatrix} w_1 & \dots & w_k \end{bmatrix}$ *and* $T_k = \begin{bmatrix} \gamma_1 & \beta_2 & & \\ \delta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \delta_k & \gamma_k \end{bmatrix}$.

Proof. The first two matrix equations are matrix expressions resulting from the non-Hermitian Lanczos algorithm. The equation $W_k^* A V_k = T_k$ can also be read from the algorithm, and using the biorthonormality of the vectors (v_j) and (w_j). Let us now check by induction that the sequences $(v_i)_{1\leq i\leq j}$ and $(w_i)_{1\leq i\leq j}$ satisfy $\langle w_k, v_\ell\rangle = \delta_{k\ell}$. For $j = 1$, the statement is true. Suppose that the biorthogonality holds true for all the vectors $(v_i)_{1\leq i\leq j}$ and $(w_i)_{1\leq i\leq j}$ and there is no breakdown in the algorithm. We have

$$\begin{aligned}\langle w_{j-1}, \hat{v}_{j+1}\rangle &= \langle w_{j-1}, Av_j - \beta_j v_{j-1}\rangle \\ &= \langle A^* w_{j-1}, v_j\rangle - \beta_j = \langle \beta_j w_j + \gamma_{j-1} w_{j-1} + \delta_{j-1} w_{j-2}, v_j\rangle - \beta_j = 0,\end{aligned}$$

thus $\langle w_{j-1}, v_{j+1}\rangle$. With similar calculations, we also deduce that $\langle w_{j+1}, v_{j-1}\rangle = 0$. It remains to show that $v_i \perp w_{j+1}$ and $w_i \perp v_{j+1}$ for $i \leq j-2$. Let $i \leq j-2$. We have

$$\begin{aligned}\langle w_i, \hat{v}_{j+1}\rangle &= \langle w_i, Av_j - \gamma_j v_j - \beta_j v_{j-1}\rangle = \langle A^* w_i, v_j\rangle \\ &= \langle \beta_{i+1} w_{i+1} + \gamma_i w_i + \delta_i w_{i-1}, v_j\rangle = 0,\end{aligned}$$

by the induction hypothesis. Proceeding similarly for proving $v_i \perp w_{j+1}$ for $i \leq j-2$, we have that $(v_i)_{1\leq i\leq j+1}$ and $(w_i)_{1\leq i\leq j+1}$ are biorthonormal. □

It is possible to derive an iterative method as a projection process, using the non-Hermitian Lanczos to define a basis for the Krylov subspace. This yields the biconjugate gradient method (BiCG) which by construction enjoys a short recurrence but does not preserve the relations between the search space and the constraint space for the residuals. The non-Hermitian Lanczos algorithm has two types of breakdowns:

- when $\widehat{v}_{j+1} = 0$ or $\widehat{w}_{j+1} = 0$: this is related to a stagnation of the corresponding subspace, which means that we have reached convergence in the projection process;

- when $\widehat{v}_{j+1} \neq 0, \widehat{w}_{j+1} \neq 0$ but $\langle \widehat{v}_{j+1}, \widehat{w}_{j+1}\rangle = 0$: this is called a serious breakdown as we do not have a stable Krylov subspace under A, and so we do not have reached convergence in the iterative algorithm. This can happen for well-conditioned matrices which indicate that it can happen generically.

Remark 4.28. *Let* $A = \begin{bmatrix} 5 & 1 & -1 \\ -5 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$, $v_1 = \begin{bmatrix} 0.6 \\ -1.4 \\ 0.3 \end{bmatrix}$ *and* $w_1 = \begin{bmatrix} 0.6 \\ 0.3 \\ -0.1 \end{bmatrix}$. *The eigenvalues of* A *are* $1, 2$ *and* 3 *and* $v_2 = \frac{1}{3}\begin{bmatrix} 1.5 \\ -2.5 \\ 1.5 \end{bmatrix}$, $w_2 = \frac{1}{3}\begin{bmatrix} 1.8 \\ 0.6 \\ -0.8 \end{bmatrix}$. *The vectors* v_2 *and* w_2 *are orthogonal although the matrix* A *is well-conditioned.*

4.3.6 GMRES in the XXIst century

GMRES has become the method of choice for solving nonhermitian linear problems. However, compared to CG, GMRES suffer from some drawbacks:

- theoretically, there is no fully satisfying explanation of the convergence behaviour of GMRES as the upper bounds are often much more pessimistic than what is actually observed;
- in practice, one would often use restarted GMRES (or one of its variants), but again, the behaviour of the algorithm with respect to the restart parameter is rather unclear.

Excellent references on that topic are the monograph [LS12] or the book by Y. Saad [Saa03].

Chapter 5

Eigenvalue problems

In this chapter, we are interested in the symmetric eigenvalue problem

$$\text{Solve for } (\mu, x) \in \mathbb{R} \times \big(\mathbb{R}^n \setminus \{0\}\big), Ax = \mu x, \tag{5.1}$$

where A is assumed symmetric.

Contrary to the linear solve case, where there is an explicit algorithm to solve the linear problem $Ax_* = b$, for $n \geq 5$, by the Abel-Ruffini theorem, there is no algorithm that computes the eigenvalues of A that stops after a finite number of operations. Otherwise, it would be possible to find the roots of a polynomial of degree n within a finite number of operations. This means that the methods to compute the eigenvalues of a matrix have to be iterative. We start with algorithms which approximate the eigenvalues of a matrix using single vector iterations.

5.1 Single vector iteration methods

5.1.1 Power iteration

Suppose that the eigenvalues $(\mu_i)_{1 \leq i \leq n}$ of the matrix A are ordered such that

$$|\mu_n| > |\mu_{n-1}| \geq \cdots \geq |\mu_1|. \tag{5.2}$$

For a starting vector $x \in \mathbb{R}^n$, consider the sequences $(y^{(k)})$ and $(\lambda^{(k)})$ defined by $y^{(0)} = \frac{x}{\|x\|}$ and $\lambda^{(0)} = \langle y^{(0)}, Ay^{(0)} \rangle$ and for $m \geq 1$

$$\begin{cases} \hat{y}^{(m)} = Ay^{(m-1)} \\ y^{(m)} = \dfrac{\hat{y}^{(m)}}{\|\hat{y}^{(m)}\|} \\ \lambda^{(m)} = \langle y^{(m)}, Ay^{(m)} \rangle. \end{cases} \tag{5.3}$$

By iteration, we see that $y^{(m)} = \frac{A^m x}{\|A^m x\|}$.

Expanding x in the basis of the eigenvectors $(q_i)_{1 \leq i \leq n}$ of A, we have

$$x = \sum_{i=1}^{n} \alpha_i q_i, \quad \alpha_i \in \mathbb{R}, \tag{5.4}$$

thus assuming that $\alpha_n \neq 0$

$$A^m x = \alpha_n \mu_n^m \Big(q_n + \sum_{i=1}^{n-1} \frac{\alpha_i}{\alpha_n} \Big(\frac{\mu_i}{\mu_n} \Big)^m q_i \Big). \tag{5.5}$$

Since $\left|\frac{\mu_i}{\mu_n}\right| < 1$ for $1 \leq i \leq n-1$, $A^m x$ tends to be aligned with the vector q_n *i.e.* the eigenvector associated to the eigenvalue with the largest magnitude.

Algorithm 5.1 Power method

```
function PowerMethod(A, x, ε_tol)
    y^(0) = x / ||x||
    λ^(0) = ⟨y^(0), A y^(0)⟩
    m = 0
    while ||A y^(m) − λ^(m) y^(m)|| > ε_tol do
        m = m + 1
        ŷ^(m) = A y^(m−1)
        y^(m) = ŷ^(m) / ||ŷ^(m)||
        λ^(m) = ⟨y^(m), A y^(m)⟩
    end while
    return (λ^(m), y^(m))
end function
```

Theorem 5.1 (Convergence of the power method)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Let $(\mu_i)_{1\leq i\leq n}$ be its eigenvalues and $(q_i)_{1\leq i\leq n}$ be the corresponding orthonormal eigenvectors. Suppose that the eigenvalues satisfy Eq. (5.2) and $\mu_n > 0$. Let $x \in \mathbb{R}^n$ such that $\langle x, q_n\rangle > 0$. Then there exists a constant $C > 0$ such that for all $m \geq 1$*

$$\|y^{(m)} - q_n\| \leq C \Big| \frac{\mu_{n-1}}{\mu_n} \Big|^m, \tag{5.6}$$

and

$$|\lambda^{(m)} - \mu_n| \leq C \Big| \frac{\mu_{n-1}}{\mu_n} \Big|^{2m}. \tag{5.7}$$

Proof. From Eq. (5.5), we have

$$\begin{aligned}
\|A^m x\|^2 &= \Big\| \alpha_n \mu_n^m \Big(q_n + \sum_{i=1}^{n-1} \frac{\alpha_i}{\alpha_n} \Big(\frac{\mu_i}{\mu_n} \Big)^m q_i \Big) \Big\|^2 \\
&= |\alpha_n \mu_n^m|^2 \Big(1 + \sum_{i=1}^{n-1} \Big| \frac{\alpha_i}{\alpha_n} \Big(\frac{\mu_i}{\mu_n} \Big)^m \Big|^2 \Big) \\
&= |\alpha_n \mu_n^m|^2 \Big(1 + \mathcal{O}\big(\frac{\mu_{n-1}}{\mu_n} \big)^{2m} \Big).
\end{aligned}$$

Thus we have

$$\begin{aligned}\|q_n - y^{(m)}\| &= \Big\|q_n - \frac{A^m x}{\|A^m x\|}\Big\| \\ &\leq \frac{\|(\|A^m x\| - \alpha_n \mu_n^m) q_n\|}{\|A^m x\|} + \frac{\Big\|\sum_{i=1}^{n-1} \alpha_i \mu_i^m q_i\Big\|}{\|A^m x\|} \\ &\leq C\Big|\frac{\mu_{n-1}}{\mu_n}\Big|^m,\end{aligned}$$

for some positive constant C independent of m.

For $\lambda^{(m)}$, we first notice that $\|q_n - y^{(m)}\|^2 = 2 - 2\langle q_n, y^{(m)}\rangle$. Thus we have

$$\begin{aligned}\langle q_n - y^{(m)}, A(q_n - y^{(m)})\rangle &= \langle q_n, Aq_n\rangle - 2\langle y^{(m)}, Aq_n\rangle + \langle y^{(m)}, Ay^{(m)}\rangle \\ &= \mu_n - 2\mu_n\langle y^{(m)}, q_n\rangle + \lambda^{(m)} \\ &= \lambda^{(m)} - \mu_n - 2\mu_n(1 - \langle y^{(m)}, q_n\rangle) \\ &= \lambda^{(m)} - \mu_n - \mu_n\|y^{(m)} - q_n\|^2.\end{aligned}$$

We deduce the bound

$$|\lambda^{(m)} - \mu_n| \leq |\mu_n|\|y^{(m)} - q_n\|^2 + |\langle q_n - y^{(m)}, A(q_n - y^{(m)})\rangle| \leq 2|\mu_n|\|y^{(m)} - q_n\|^2 \leq C\Big|\frac{\mu_{n-1}}{\mu_n}\Big|^{2m}. \tag{5.8}$$

□

Notice that the eigenvalue converges at a rate which is twice faster than the eigenvector. This is a result which also applies in general.

Remark 5.2. *The power iteration also converges when A is diagonalisable (not necessarily in an orthonormal basis) under the assumption Eq. (5.2) on the eigenvalues.*

The convergence of the power iteration is sensitive to an eigenvalue that is close (in magnitude) to the dominant one μ_n. In particular if $\mu_{n-1} = \mu_n(1 - \varepsilon)$ for $\varepsilon > 0$, then

$$\frac{\mu_{n-1}}{\mu_n} = 1 - \varepsilon. \tag{5.9}$$

The convergence rate of the power iteration method in that case can be written

$$\|y^{(m)} - q_n\| \leq C(1 - \varepsilon)^m \quad \text{and} \quad |\lambda^{(m)} - \mu_n| \leq C(1 - \varepsilon)^{2m}.$$

5.1.2 Inverse power iteration

Assume now that the eigenvalues of A are ordered such that

$$0 < |\mu_1| < |\mu_2| \leq \cdots \leq |\mu_n|. \tag{5.10}$$

Applying the power method to the matrix A^{-1} gives the following sequence

$$\begin{cases} \hat{y}^{(m)} = A^{-1} y^{(m-1)} \\ y^{(m)} = \dfrac{\hat{y}^{(m)}}{\|\hat{y}^{(m)}\|} \\ \lambda^{(m)} = \langle y^{(m)}, A^{-1} y^{(m)}\rangle, \end{cases} \tag{5.11}$$

where $y^{(0)} = \frac{x}{\|x\|}$ and $\lambda^{(0)} = \langle y^{(0)}, A^{-1}y^{(0)}\rangle$.

By Theorem 5.1, $(\lambda^{(m)}, y^{(m)})$ converge to the eigenpair corresponding to the dominant eigenvalue of A^{-1}, thus $(\frac{1}{\mu_1}, q_1)$.

The formulation in Eq. (5.11) is not interesting in practice as it would require the knowledge of the inverse A. However, by introducing the appropriate intermediate variable, it is possible to replace A^{-1} by linear solve at each step.

Algorithm 5.2 Inverse power method

function InversePowerMethod($A, x, \varepsilon_{\text{tol}}$)
  $y^{(0)} = \frac{x}{\|x\|}$
  $x^{(0)}$ solves $Ax^{(0)} = y^{(0)}$
  $\lambda^{(0)} = \langle y^{(0)}, x^{(0)}\rangle$
  $m = 0$
  while $\|y^{(m)} - \lambda^{(m)} A y^{(m)}\| > \varepsilon_{\text{tol}}$ **do**
    $m = m + 1$
    $y^{(m)} = \frac{x^{(m-1)}}{\|x^{(m-1)}\|}$
    $x^{(m)}$ solves $Ax^{(m)} = y^{(m)}$
    $\lambda^{(m)} = \langle y^{(m)}, x^{(m)}\rangle$
  end while
  return $(\frac{1}{\lambda^{(m)}}, y^{(m)})$
end function

The sequence defined by Algorithm 5.2 is mathematically equivalent to applying the power iteration to A^{-1} but requires only a linear solve per iteration of the while loop. We have by a simple consequence of Theorem 5.1 the following convergence for the inverse power iteration method.

Theorem 5.3. *Let $A \in \mathbb{R}^{n\times n}$ be an Hermitian matrix. Let $(\mu_i)_{1\le i\le n}$ be its eigenvalues and $(q_i)_{1\le i\le n}$ be the corresponding orthonormal eigenvectors. Suppose that the eigenvalues satisfy Eq. (5.10). Let $x \in \mathbb{R}^n$ such that $\langle x, q_1\rangle > 0$. Then there exists a constant $C > 0$ such that for all $m \ge 1$*

$$\|y^{(m)} - q_1\| \le C\left|\frac{\mu_1}{\mu_2}\right|^m, \tag{5.12}$$

and

$$|\lambda^{(m)} - \mu_1| \le C\left|\frac{\mu_1}{\mu_2}\right|^{2m}. \tag{5.13}$$

Again the convergence of the inverse power method is slow if the ratio $\left|\frac{\mu_1}{\mu_2}\right|$ is close to 1.

In this case, if we have a good guess $\tilde{\mu}_1 \neq \mu_1$ to the exact eigenvalue, we can apply the inverse power method to the matrix $A - \tilde{\mu}_1 \,\mathrm{id}_n$ which satisfies

$$|\mu_1 - \tilde{\mu}_1| \ll |\mu_2 - \tilde{\mu}_1| \le \cdots \le |\mu_n - \tilde{\mu}_1|.$$

Hence the inverse power iteration converges much faster for $A - \tilde{\mu}_1$ than for A.

Remark 5.4. *Shifting by $\sigma \in \mathbb{R}$ the matrix A can also be used to target a specific eigenvalue μ_i of A.*

Indeed let σ such that $|\mu_i - \sigma| < |\mu_j - \sigma|$ for all $j \neq i$. Then we have that the inverse power iteration applied to the matrix $A - \sigma\,\mathrm{id}$ converges to the eigenvector q_i of A.

5.2 Krylov methods

We will assume here that A is Hermitian with eigenvalues

$$\mu_1 \leq \cdots \leq \mu_n. \tag{5.14}$$

Inspired by the iterative solver case, we expect a Krylov method for linear solvers to converge faster to the exact eigenvalue than a single vector iteration. In that case, the approximate eigenvector is sought in a Krylov space $\mathcal{K}_{m+1}(A, y)$. If we are interested in the largest eigenvalue μ_n, it is natural to start from the variational characterisation of the eigenvalue (see Proposition 1.12)

$$\mu_n = \max_{\substack{y\in\mathbb{R}^n \\ y\neq 0}} \frac{\langle y, Ay\rangle}{\|y\|^2},$$

and restrict the maximisation problem to the Krylov space $\mathcal{K}_{m+1}(A, y)$

$$\lambda^{(m)} = \max_{\substack{y\in\mathcal{K}_{m+1}(A,y) \\ y\neq 0}} \frac{\langle y, Ay\rangle}{\|y\|^2}.$$

By the restriction, we necessarily have $\mu_n - \lambda^{(m)} \geq 0$.

The Arnoldi/Lanczos algorithm (see Proposition 4.8), as A is symmetric, provides an orthonormal basis $(v_1, \ldots, v_{m+1})$ of $\mathcal{K}_{m+1}(A, y)$ such that $V_{m+1}^T A V_{m+1} = T_{m+1}$ where $V_{m+1} = \begin{bmatrix} v_1, \ldots, v_{m+1} \end{bmatrix} \in \mathbb{R}^{n\times(m+1)}$ and $T_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ is tridiagonal. Using this basis, we can write

$$\lambda^{(m)} = \max_{\substack{y\in\mathcal{K}_{m+1}(A,y) \\ y\neq 0}} \frac{\langle y, Ay\rangle}{\|y\|^2} = \max_{\substack{t\in\mathbb{R}^{m+1} \\ t\neq 0}} \frac{\langle V_{m+1}t, AV_{m+1}t\rangle}{\|V_{m+1}t\|^2} = \max_{\substack{t\in\mathbb{R}^{m+1} \\ t\neq 0}} \frac{\langle t, V_{m+1}^T AV_{m+1}t\rangle}{\|t\|^2} = \max_{\substack{t\in\mathbb{R}^{m+1} \\ t\neq 0}} \frac{\langle t, T_{m+1}t\rangle}{\|t\|^2}.$$

We deduce that $\lambda^{(m)}$ is simply the largest eigenvalue of T_{m+1}. Since T_{m+1} is tridiagonal and smaller than A, it is reasonable to expect efficient eigenvalue solvers for such matrices. Let $t^{(m+1)} \in \mathbb{R}^{m+1}$ a normalised eigenvector associated to $T_{m+1}t^{(m+1)} = \lambda^{(m)}t^{(m+1)}$, then the approximate eigenpair using the Krylov subspace is $(\lambda^{(m)}, V_{m+1}t^{(m+1)})$. The corresponding procedure called the *iterative Arnoldi algorithm* is summarised in the Algorithm 5.3.

Remark 5.5. *In the iterative Arnoldi algorithm 5.3, we can by-pass the estimation of the residual $\|Ay^{(m)} - \lambda^{(m)}y^{(m)}\|$ using that $y^{(m)} = V_m t^{(m)}$ and that*

$$Ay^{(m)} - \lambda^{(m)}y^{(m)} = AV_m t^{(m)} - V_m T_m t^{(m)} = -t_{m+1,m} v_{m+1} e_m^T t^{(m)}.$$

Thus $\|Ay^{(m)} - \lambda^{(m)}y^{(m)}\| = |t_{m+1,m}||t_m^{(m)}|$. To avoid the computation of the largest eigenvalue of T_m, it is also possible to just use the coefficient $t_{m+1,m}$ of the Arnoldi/Lanczos algorithm in order to estimate the residual.

Theorem 5.6. *Let A be a Hermitian matrix with eigenvalues $\mu_1 \leq \cdots \leq \mu_{n-1} < \mu_n$. Let q_n be an eigenvector associated to μ_n. Let $(\lambda^{(m)}, y^{(m)})$ be the result of the iterative Arnoldi algorithm for* `which=largest` *(Algorithm 5.3) starting with the vector y. Assume that $\langle q_n, y\rangle \neq 0$.*

Algorithm 5.3 Iterative Arnoldi algorithm

function ITERATIVEARNOLDI($A, y, \varepsilon_{\text{tol}}$; `which` = {`largest`, `smallest`})
  $y^{(0)} = \frac{y}{\|y\|}$
  $\lambda^{(0)} = \langle v^{(0)}, Av^{(0)} \rangle$
  $m = 0$
  while $\|Ay^{(m)} - \lambda^{(m)} y^{(m)}\| > \varepsilon_{\text{tol}}$ **do**
    $m = m + 1$
    Compute $V_m \in \mathbb{R}^{n \times m}, T_m \in \mathbb{R}^{m \times m}$ of the Hermitian Lanczos algorithm 4.2
    Compute the `which` eigenvalue $\lambda^{(m)}$ and the eigenvector $t^{(m)}$ of $H_{mm} = V_m^* A V_m$
    $y^{(m)} = V_m t^{(m)}$
  end while
  return $\lambda^{(m)}, y^{(m)}$
end function

Then there exists a constant $C > 0$ independent of m such that the error on the approximated eigenvalue $\lambda^{(m)}$ given after m steps of the iterative Arnoldi algorithm is given by

$$0 \leq \mu_n - \lambda^{(m)} \leq C \frac{\mu_n - \mu_1}{T_m\left(\frac{2\mu_n - \mu_{n-1} - \mu_1}{\mu_{n-1} - \mu_1}\right)^2} \leq 4C(\mu_n - \mu_1)\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2m}, \tag{5.15}$$

where T_m is the m-th Chebyshev polynomial of the first kind and $\kappa = \frac{\mu_n - \mu_1}{\mu_n - \mu_{n-1}}$.

Proof. The approximate eigenvalue $\lambda^{(m)}$ is characterised by

$$\lambda^{(m)} = \max_{x \in \mathcal{K}_{m+1}(A,y), x \neq 0} \frac{\langle x, Ax \rangle}{\langle x, x \rangle},$$

hence

$$\mu_n - \lambda^{(m)} = \min_{x \in \mathcal{K}_{m+1}(A,y), x \neq 0} \frac{\langle x, (\mu_n - A)x \rangle}{\langle x, x \rangle}. \tag{5.16}$$

Using that $\mathcal{K}_{m+1}(A, y) = \mathcal{K}_{m+1}(\mu_n - A, y) = \{P(A)y, P \in \mathbb{C}^m[X]\} = \{P(\mu_n - A)y, P \in \mathbb{C}^m[X]\}$, we have that

$$\mu_n - \lambda^{(m)} = \min_{0 \neq P \in \mathbb{C}^m[X]} \frac{\langle P(\mu_n - A)y, (\mu_n - A)P(\mu_n - A)y \rangle}{\langle P(\mu_n - A)y, P(\mu_n - A)y \rangle}. \tag{5.17}$$

Expanding y in the orthonormal eigenvector basis, we have $y = \sum_{i=1}^n \alpha_i q_i$, thus

$$\mu_n - \lambda^{(m)} = \min_{0 \neq P \in \mathbb{C}^m[X]} \frac{\sum_{i=1}^{n-1} |\alpha_i (\mu_n - \mu_i) P(\mu_n - \mu_i)|^2}{\sum_{i=1}^{n} |\alpha_i P(\mu_n - \mu_i)|^2}. \tag{5.18}$$

We thus obtain the upper bound

$$\begin{aligned}\mu_n - \lambda^{(m)} &\leq (\mu_n - \mu_1) \min_{0\neq P\in\mathbb{C}^m[X]} \frac{\sum_{i=1}^{n-1} |\alpha_i P(\mu_n-\mu_i)|^2}{\sum_{i=1}^{n} |\alpha_i P(\mu_n-\mu_i)|^2} \\ &\leq (\mu_n - \mu_1) \min_{0\neq P\in\mathbb{C}^m[X]} \frac{\sum_{i=1}^{n-1} |\alpha_i P(\mu_n-\mu_i)|^2}{|\alpha_n P(0)|^2} \\ &\leq (\mu_n - \mu_1) \frac{\sum_{i=1}^{n-1} |\alpha_i|^2}{|\alpha_n|^2} \min_{0\neq P\in\mathbb{C}^m[X]} \max_{1\leq i\leq n-1} \frac{|P(\mu_n-\mu_i)|^2}{|P(0)|^2}.\end{aligned}$$

We can relax the min-max problem to

$$\min_{0\neq P\in\mathbb{C}^m[X]} \max_{1\leq i\leq n-1} \frac{|P(\mu_n-\mu_i)|^2}{|P(0)|^2} \leq \min_{\substack{P\in\mathbb{C}^m[X]\\ P(0)=1}} \max_{\mu_n-\mu_{n-1}\leq\mu\leq\mu_n-\mu_1} |P(\mu)|^2.$$

We recognise the same min-max problem that is solved by a shifted and rescaled Chebyshev polynomial T_k (see Remark 4.18):

$$\min_{\substack{P\in\mathbb{C}^{m-1}[X]\\ P(0)=1}} \max_{\mu_n-\mu_{n-1}\leq\mu\leq\mu_n-\mu_1} |P(\mu)|^2 = \frac{1}{T_{m-1}\Big(\frac{2\mu_n-\mu_{n-1}-\mu_1}{\mu_{n-1}-\mu_1}\Big)^2} \leq 4\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^{2m},$$

with $\kappa = \frac{\mu_n-\mu_1}{\mu_n-\mu_{n-1}}$. □

Remark 5.7. *The convergence of the Arnoldi process is also sensitive to μ_{n-1} close to μ_n. However, as in the linear solver case, the situation is better for the Krylov solver. Suppose that $\mu_1 \ll \mu_n$ and $\mu_{n-1} = \mu_n(1-\varepsilon)$ for some $1 \gg \varepsilon > 0$. Then we have*

$$\kappa = \frac{\mu_n-\mu_1}{\mu_n-\mu_{n-1}} = \frac{1-\frac{\mu_1}{\mu_n}}{1-\frac{\mu_{n-1}}{\mu_n}} \sim \frac{1}{\varepsilon}.$$

Thus the convergence rate becomes

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \sim \frac{1-\sqrt{\varepsilon}}{1+\sqrt{\varepsilon}} \sim 1-2\sqrt{\varepsilon}.$$

Compared to the power iteration case (5.9), we have gained a square root in the convergence rate.

A similar estimate can be obtained for the lowest eigenvalue of the matrix A, provided that we now assume a gap between μ_1 and μ_2, and that in the iterative Arnoldi algorithm, the lowest eigenvalue is computed instead of the largest one.

Corollary 5.8. *Let A be a Hermitian matrix with eigenvalues $\mu_1 < \mu_2 \leq \cdots \leq \mu_n$. Let q_1 be an eigenvector associated to μ_1. Let $(\lambda^{(m)}, y^{(m)})$ be the result of the iterative Arnoldi algorithm* `which=smallest` *(Algorithm 5.3) starting with the vector y. Suppose that $\langle q_1, y\rangle \neq 0$. Then*

there exists a constant $C > 0$ independent of m such that the error on the approximated eigenvalue $\lambda^{(m)}$ given after m steps of the iterative Arnoldi algorithm is given by

$$0 \leq \lambda^{(m)} - \mu_1 \leq C \frac{\mu_n - \mu_1}{T_m\left(\frac{\mu_n+\mu_2-2\mu_1}{\mu_n-\mu_2}\right)^2} \leq 4C(\mu_n - \mu_1)\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2m}, \tag{5.19}$$

where T_m is the m-th Chebyshev polynomial of the first kind and $\kappa = \frac{\mu_n-\mu_1}{\mu_2-\mu_1}$.

Proof. Apply Theorem 5.6 to the matrix $-A$. □

Bibliography

[CG22] Gabriele Ciaramella and Martin J. Gander. *Iterative Methods and Preconditioners for Systems of Linear Equations*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, January 2022.

[GL13] Gene Howard Golub and Charles F. Van Loan. *Matrix Computations*. JHU Press, February 2013.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2002.

[HJ13] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, second edition, 2013.

[LS12] Jörg Liesen and Zdenek Strakos. *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, October 2012.

[Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2003.

[Saa11] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2011.