Numerical Solution of Linear Eigenvalue Problems

Jessica Bosch and Chen Greif

Abstract. We review numerical methods for computing eigenvalues of matrices. We start by considering the computation of the dominant eigenpair of a general dense matrix using the power method, and then generalize to orthogonal iterations and the QR iteration with shifts. We also consider divide-and-conquer algorithms for tridiagonal matrices. The second part of this survey involves the computation of eigenvalues of large and sparse matrices. The Lanczos and Arnoldi methods are developed and described within the context of Krylov subspace eigensolvers. We also present the idea of the Jacobi–Davidson method.

1. Introduction

Eigenvalue problems form one of the central problems in Numerical Linear Algebra. They arise in many areas of sciences and engineering. In this survey, we study linear eigenvalue problems. The standard algebraic eigenvalue problem has the form

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}.$$

We consider real matrices $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. There is a wide range of publications dealing with numerical methods for solving eigenvalue problems, e.g., [**39, 66, 22, 47, 62, 55, 32, 54**]. Typically, eigensolvers are classified into methods for symmetric (or Hermitian) and nonsymmetric (or non-Hermitian) matrices, or methods for small, dense matrices and large, sparse matrices.

This survey reviews popular methods for computing eigenvalues of a given matrix. It follows a minicourse presented by the second author at the 2015 Summer School on "Geometric and Computational Spectral Theory" at the Université de Montréal, and can viewed as a set of comprehensive lecture notes.

When it comes to numerical computation of eigenvalues, it is reasonable to classify eigensolvers by the size and the nonzero pattern of the matrix. As opposed to the solution of linear systems, where it is possible to obtain a solution within a finite number of steps, most eigenvalue computations (except trivial cases such as a diagonal or a triangular matrix) require an iterative process. For matrices that are not particularly large and do not have a specific nonzero structure, eigensolvers

2010 *Mathematics Subject Classification.* Primary 65F15.

are often based on matrix decompositions. One may be interested in a small number of the eigenvalues and/or eigenvectors, or all of them, and there are methods that are available for accomplishing the stated goal. On the other hand, when the matrix is large and sparse, it is rare to seek the entire spectrum; in most cases we are interested in just a few eigenvalues and eigenvectors, and typical methods are based on matrix-vector products rather than matrix decompositions. Interestingly, despite the fact that all processes of eigenvalue computations are iterative, methods that are based on matrix decompositions are often referred to as *direct*, whereas methods that are based on matrix-vector products are termed *iterative*. This slight abuse of terminology is nonetheless widely understood and typically does not cause any confusion.

It is a bit ambitious to talk in general terms about a recipe for solution of eigenvalue problems, but it is legitimate to identify a few main components. A typical eigensolver starts with applying similarity transformations and transforming the matrix into one that has an appealing nonzero structure: for example tridiagonal if the original matrix was symmetric. Once this is accomplished, an iterative process is pursued, whereby repeated orthogonal similarity transformations are applied to get us closer and closer to a diagonal or triangular form. For large and sparse matrices, an additional component, generally speaking, in state of the art methods, is the transformation of the problem to a small and dense one on a projected subspace.

This survey devotes a significant amount of space to elaborating on the above principles. It is organized as follows. In Section 2, we briefly review basic concepts of Numerical Linear Algebra that are related to eigenvalue problems. We start with presenting methods for computing a few up to all eigenvalues for small to moderate-sized matrices in Section 3. This is followed by a review of eigenvalue solvers for large and sparse matrices in Section 4. Conclusions complete the paper.

2. Background in Numerical Linear Algebra

2.1. Theoretical basics.

We begin our survey with a review of basic concepts in Numerical Linear Algebra. We introduce some notation used throughout the survey.

Let $\boldsymbol{A} \in \mathbb{C}^{m\times n}$. The *kernel* or *nullspace* of $\boldsymbol{A}$ is given as

$$\ker(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{C}^n \colon \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}\,.$$

Another important subspace, which is often related to the kernel, is the *range* of $\boldsymbol{A}$, which given as

$$\operatorname{ran}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} \colon \boldsymbol{x} \in \mathbb{C}^n\}\,.$$

The *rank* of $\boldsymbol{A}$ is the maximal number of linearly independent columns (or rows), i.e.,

$$\operatorname{rank}(\boldsymbol{A}) = \dim\left(\operatorname{ran}(\boldsymbol{A})\right).$$

It holds $n = \operatorname{rank}(\boldsymbol{A}) + \dim\left(\ker(\boldsymbol{A})\right)$. $\boldsymbol{A}$ is called *rank-deficient* if $\operatorname{rank}(\boldsymbol{A}) < \min\{m, n\}$.

In what follows, we consider real matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ if not stated otherwise.

Definition 2.1 (Invertibility). $\boldsymbol{A}$ is called *invertible* or *nonsingular* if there exists a matrix $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ such that

$$\boldsymbol{AB} = \boldsymbol{BA} = \boldsymbol{I}.$$

Here, $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the *identity matrix.* The *inverse* of $\boldsymbol{A}$ is uniquely determined, and we denote it by $\boldsymbol{A}^{-1}$.

Related to the inverse and a matrix norm $\|\cdot\|$ is the *condition number*, which is defined for a general square matrix $\boldsymbol{A}$ as

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\,\|\boldsymbol{A}^{-1}\|.$$

In general, if $\kappa(\boldsymbol{A})$ is large[1], then $\boldsymbol{A}$ is said to be an *ill-conditioned* matrix. Useful matrix norms include the well-known *p-norms* or the *Frobenius norm* $\|\cdot\|_F$, which is given as

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{i,j}|^2},$$

where $a_{i,j}$ is the (i,j) entry of $\boldsymbol{A}$.

Let us come to the heart of this paper. The *algebraic eigenvalue problem* has the following form:

Definition 2.2 (Algebraic Eigenvalue Problem). $\lambda \in \mathbb{C}$ is called an *eigenvalue* of $\boldsymbol{A}$ if there exists a vector $\boldsymbol{0} \neq \boldsymbol{x} \in \mathbb{C}^n$ such that

$$\boldsymbol{Ax} = \lambda \boldsymbol{x}.$$

The vector $\boldsymbol{x}$ is called a (right) *eigenvector* of $\boldsymbol{A}$ associated with λ. We call the pair $(\lambda, \boldsymbol{x})$ an *eigenpair* of $\boldsymbol{A}$. The set of all eigenvalues of $\boldsymbol{A}$ is called the *spectrum* of $\boldsymbol{A}$ and is denoted by $\lambda(\boldsymbol{A})$.

Note from the above definition that real matrices can have complex eigenpairs. Geometrically, the action of a matrix $\boldsymbol{A}$ expands or shrinks any vector lying in the direction of an eigenvector of $\boldsymbol{A}$ by a scalar factor. This scalar factor is given by the corresponding eigenvalue of $\boldsymbol{A}$.

Remark 2.3. Similarly, a left eigenvector of $\boldsymbol{A}$ associated with the eigenvalue λ is defined as a vector $\boldsymbol{0} \neq \boldsymbol{y} \in \mathbb{C}^n$ that satisfies $\boldsymbol{y}^*\boldsymbol{A} = \lambda \boldsymbol{y}^*$. Here, $\boldsymbol{y}^* = \bar{\boldsymbol{y}}^T$ is the *conjugate transpose* of $\boldsymbol{y}$.

Throughout the survey, we use the term *eigenvector* for a right eigenvector.

The eigenvalues of $\boldsymbol{A}$ can be used to determine the invertibility of $\boldsymbol{A}$.

Definition 2.4 (Determinant). Let $\lambda(\boldsymbol{A}) = \{\lambda_1, \ldots, \lambda_n\}$. The *determinant* of $\boldsymbol{A}$ is given as

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i.$$

$\boldsymbol{A}$ is nonsingular if and only if $\det(\boldsymbol{A}) \neq 0$.

Another way to define eigenvalues is the following:

[1]Of course, this depends on the definition of “large”; see, e.g., [**22**, Chap. 3.5].

Definition 2.5 (Characteristic Polynomial). The polynomial

$$p_{\boldsymbol{A}}(x) = \det(\boldsymbol{A} - x\boldsymbol{I})$$

is called the *characteristic polynomial* of $\boldsymbol{A}$. It is a polynomial of degree n. The roots of $p_{\boldsymbol{A}}(x)$ are the eigenvalues of $\boldsymbol{A}$.

A useful concept for eigenvalues solvers is the *Rayleigh quotient*:

Definition 2.6 (Rayleigh quotient). Let $\boldsymbol{0} \neq \boldsymbol{z} \in \mathbb{C}^n$. The *Rayleigh quotient* of $\boldsymbol{A}$ and $\boldsymbol{z}$ is defined by

$$R_{\boldsymbol{A}}(\boldsymbol{z}) = \frac{\boldsymbol{z}^*\boldsymbol{A}\boldsymbol{z}}{\boldsymbol{z}^*\boldsymbol{z}}.$$

Note that if $\boldsymbol{z}$ is an eigenvector of $\boldsymbol{A}$, then the Rayleigh quotient is the corresponding eigenvalue.

Another way to express the eigenvalue problem in Definition 2.2 is that $(\lambda, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{A}$ if and only if $\boldsymbol{0} \neq \boldsymbol{x} \in \ker(\boldsymbol{A} - \lambda\boldsymbol{I})$. Based on this kernel, we can define the *eigenspace* of $\boldsymbol{A}$:

Definition 2.7 (Eigenspace).

$$\mathcal{E}_\lambda(\boldsymbol{A}) = \ker(\boldsymbol{A} - \lambda\boldsymbol{I})$$

is the *eigenspace* of $\boldsymbol{A}$ corresponding to λ.

The following concept of *invariant subspaces* bears similarities to the eigenvalue problem.

Definition 2.8 (Invariant Subspace). A subspace $\mathcal{S} \subset \mathbb{C}^n$ is said to be *invariant* under a matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ ($\boldsymbol{A}$-invariant) if $\boldsymbol{A}\mathcal{S} \subset \mathcal{S}$.

Definition 2.9. Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, $\mathcal{S} \subset \mathbb{C}^n$, and $\boldsymbol{S} \in \mathbb{C}^{n\times k}$ with $k = \operatorname{rank}(\boldsymbol{S}) = \dim(\mathcal{S}) \leq n$ and $\mathcal{S} = \operatorname{ran}(\boldsymbol{S})$. Then, $\mathcal{S}$ is $\boldsymbol{A}$-invariant if and only if there exists a matrix $\boldsymbol{B} \in \mathbb{C}^{k\times k}$ such that

$$\boldsymbol{A}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{B}.$$

Remark 2.10. Using Definition 2.9, it is easy to show the following relations:

- If $(\lambda, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{B}$, then $(\lambda, \boldsymbol{S}\boldsymbol{x})$ is an eigenpair of $\boldsymbol{A}$. Hence, $\lambda(\boldsymbol{B}) \subset \lambda(\boldsymbol{A})$.
- If $k = n$, then $\boldsymbol{S}$ is invertible and hence

  $$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{B}\boldsymbol{S}^{-1}.$$

  This means $\boldsymbol{A}$ and $\boldsymbol{B}$ are *similar* and $\lambda(\boldsymbol{B}) = \lambda(\boldsymbol{A})$. This concept is introduced next.

Most of the presented algorithms will transform a matrix $\boldsymbol{A}$ into simpler forms, such as diagonal or triangular matrices, in order to simplify the original eigenvalue problem. Transformations that preserve the eigenvalues of matrices are called *similarity transformations.*

Definition 2.11 (Similarity Transformation). Two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{n\times n}$ are said to be *similar* if there exists a nonsingular matrix $\boldsymbol{C} \in \mathbb{C}^{n\times n}$ such that

$$\boldsymbol{A} = \boldsymbol{C}\boldsymbol{B}\boldsymbol{C}^{-1}.$$

The mapping $\boldsymbol{B} \to \boldsymbol{A}$ is called a *similarity transformation.* The similarity of $\boldsymbol{A}$ and $\boldsymbol{B}$ implies that they have the same eigenvalues. If $(\lambda, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{B}$, then $(\lambda, \boldsymbol{C}\boldsymbol{x})$ is an eigenpair of $\boldsymbol{A}$.

The simplest form to which a matrix can be transformed is a diagonal matrix. But as we will see, this is not always possible.

Definition 2.12 (Diagonalizability). If $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is similar to a diagonal matrix, then $\boldsymbol{A}$ is said to be *diagonalizable.*

A similarity transformation in which $\boldsymbol{C}$ is orthogonal (or unitary), i.e., $\boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}$ (or $\boldsymbol{C}^*\boldsymbol{C} = \boldsymbol{I}$), is called *orthogonal (or unitary) similarity transformation.* Unitary/orthogonal similarity transformations play a key role in numerical computations since $\|\boldsymbol{C}\|_2 = 1$. Considering the calculation of similarity transformations, it can be shown (cf. [**22**, Chap. 7.1.5]) that the roundoff error $\boldsymbol{E}$ satisfies

$$\|\boldsymbol{E}\| \approx \epsilon_{\text{machine}}\, \kappa_2\,(\boldsymbol{C})\|\boldsymbol{A}\|_2.$$

Here, $\epsilon_{\text{machine}}$ is the *machine precision*[2] and $\kappa_2(\boldsymbol{C})$ the condition number of $\boldsymbol{C}$ with respect to the 2-norm. In particular, $\kappa_2(\boldsymbol{C})$ is the error gain. Therefore, if the similarity transformation is unitary, we get

$$\|\boldsymbol{E}\| \approx \epsilon_{\text{machine}}\|\boldsymbol{A}\|_2$$

and hence no amplification of error.

Theorem 2.13 (Unitary Diagonalizability; see [**22**, Cor. 7.1.4].). *$\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is unitarily diagonalizable if and only if it is normal ($\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^*$).*

Now, let us show the connection between a similarity transformation of a matrix $\boldsymbol{A}$ and its eigenpairs: It follows from Definition 2.5 and the Fundamental Theorem of Algebra that $\boldsymbol{A}$ has n (not necessarily distinct) eigenvalues. If we denote the n eigenpairs by $(\lambda_1, \boldsymbol{x}_1), \dots, (\lambda_n, \boldsymbol{x}_n)$, i.e. $\boldsymbol{A}\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$ for $i = 1, \dots, n$, we can write

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda}, \tag{2.1}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_i)_{i=1,\dots,n} \in \mathbb{C}^{n\times n}$ is a diagonal matrix containing the eigenvalues, and $\boldsymbol{X} = [\boldsymbol{x}_1|\dots|\boldsymbol{x}_n] \in \mathbb{C}^{n\times n}$ is a matrix whose columns are formed by the eigenvectors. This looks almost as a similarity transformation. In fact, the "only" additional ingredient we need is the invertibility of $\boldsymbol{X}$. Under the assumption that $\boldsymbol{X}$ is nonsingular, we obtain $\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{\Lambda}$, and hence, $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ are similar. But when can we expect of $\boldsymbol{X}$ to be nonsingular? To discuss this, we introduce some terminology:

Definition 2.14 (Multiplicity). Let λ be an eigenvalue of $\boldsymbol{A}$.

- λ has *algebraic multiplicity* m^a, if it is a root of multiplicity m^a of the characteristic polynomial $p_{\boldsymbol{A}}$.
- If $m^a = 1$, then λ is called *simple.* Otherwise, λ is said to be *multiple.*
- The *geometric multiplicity* m^g of λ is defined as the dimension of the associated eigenspace, i.e., $m^g = \dim(\mathcal{E}_\lambda(\boldsymbol{A}))$. It is the maximum number of independent eigenvectors associated with λ.
- It holds $m^g \leq m^a$.

[2]The machine precision is $\epsilon_{\text{machine}} = 2^{-53} \approx 1.11 \cdot 10^{-16}$ in the double precision IEEE floating point format and $\epsilon_{\text{machine}} = 2^{-24} \approx 5.96 \cdot 10^{-6}$ in the single precision IEEE floating point format. For more details, we refer to, e.g., [**37, 66, 27**].

- If $m^g < m^a$, then λ and $\boldsymbol{A}$ are called *defective* or *non-diagonalizable.*

Note that if all eigenvalues of $\boldsymbol{A}$ are simple, then they are distinct. Now, we can state a result about the nonsingularity of the eigenvector matrix $\boldsymbol{X}$ in (2.1):

Theorem 2.15 (Diagonal Form; see [**22**, Cor. 7.1.8].). *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$. $\boldsymbol{A}$ is nondefective if and only if there exists a nonsingular matrix $\boldsymbol{X} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \operatorname{diag}(\lambda_i)_{i=1,\dots,n}.$$

The similarity transformation given in Theorem 2.15 transforms $\boldsymbol{A}$ into a diagonal matrix whose entries reveal the eigenvalues of $\boldsymbol{A}$.

We have seen that a similarity transformation to a diagonal matrix is not always possible. Before we come to the next similarity transformation, we introduce the concept of *deflation* – the process of breaking down an eigenvalue problem into smaller eigenvalue problems.

Theorem 2.16 (See [**22**, Lemma 7.1.3].). *Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, $\boldsymbol{S} \in \mathbb{C}^{n\times k}$ with* $\operatorname{rank}(\boldsymbol{S}) = k < n$ *and $\boldsymbol{B} \in \mathbb{C}^{k\times k}$ such that*

$$\boldsymbol{A}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{B},$$

i.e., $\operatorname{ran}(\boldsymbol{S})$ *is an $\boldsymbol{A}$-invariant subspace. Then, there exists a unitary $\boldsymbol{Q} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{0} & \boldsymbol{T}_{22} \end{bmatrix}$$

and

$$\begin{aligned} \lambda(\boldsymbol{T}) &= \lambda(\boldsymbol{T}_{11}) \cup \lambda(\boldsymbol{T}_{22}), \\ \lambda(\boldsymbol{T}_{11}) &= \lambda(\boldsymbol{A}) \cap \lambda(\boldsymbol{B}) \end{aligned}$$

with $\boldsymbol{T}_{11} \in \mathbb{C}^{k\times k}$.

From Theorem 2.16, we obtain a similarity transformation that transforms a matrix $\boldsymbol{A}$ into an upper triangular matrix whose diagonal entries reveal the eigenvalues of $\boldsymbol{A}$. Such a decomposition always exists.

Theorem 2.17 (Schur Decomposition; see [**22**, Theor. 7.1.3].). *Given $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ with eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$. Then, there exists a unitary matrix $\boldsymbol{Q} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T} = \boldsymbol{D} + \boldsymbol{N},$$

where $\boldsymbol{D} = \operatorname{diag}(\lambda_i)_{i=1,\dots,n}$*, and $\boldsymbol{N} \in \mathbb{C}^{n\times n}$ is strictly upper triangular. Moreover, $\boldsymbol{Q}$ can be chosen such that the eigenvalues λ_i appear in any order in $\boldsymbol{D}$.*

The transformation in Theorem 2.17 deals with a complex matrix $\boldsymbol{Q}$ even when $\boldsymbol{A}$ is real. A slight variation of the Schur decomposition shows that complex arithmetic can be avoided in this case. This is based on the fact that complex eigenvalues always occur in complex conjugate pairs, i.e., if $(\lambda, \boldsymbol{x})$ is an eigenpair of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, then $(\bar{\lambda}, \bar{\boldsymbol{x}})$ is an eigenpair of $\boldsymbol{A}$.

THEOREM 2.18 (Real Schur Decomposition; see [**22**, Theor. 7.4.1].). *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$. Then, there exists an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ such that*

$$\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_{1,1} & \cdots & \boldsymbol{T}_{1,m} \\ & \ddots & \vdots \\ & & \boldsymbol{T}_{m,m} \end{bmatrix},$$

where $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ is quasi-upper triangular. The diagonal blocks $\boldsymbol{T}_{i,i}$ are either 1×1 or 2×2 matrices. A 1×1 block corresponds to a real eigenvalue $\lambda_j \in \mathbb{R}$. A 2×2 block corresponds to a pair of complex conjugate eigenvalues. For a complex conjugate eigenvalue pair $\lambda_k = \mu + \imath\nu$, $\lambda_l = \mu - \imath\nu$, $\boldsymbol{T}_{i,i}$ has the form

$$\boldsymbol{T}_{i,i} = \begin{bmatrix} \mu & \nu \\ -\nu & \mu \end{bmatrix}.$$

Moreover, $\boldsymbol{Q}$ can be chosen such that the diagonal block $\boldsymbol{T}_{i,i}$ appear in any order in $\boldsymbol{T}$.

The next similarity transformation we present transforms a matrix $\boldsymbol{A}$ into *upper Hessenberg* form. Such a decomposition always exists and will play an important role in eigenvalue solvers for nonsymmetric matrices.

THEOREM 2.19 (Hessenberg Decomposition). *Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$. Then, there exists a unitary matrix $\boldsymbol{Q} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & h_{n-1,n} \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix}.$$

$\boldsymbol{H}$ is called an upper Hessenberg matrix. Further, $\boldsymbol{H}$ is said to be unreduced if $h_{j+1,j} \neq 0$ for all $j = 1, \dots, n-1$.

From a theoretical point of view, one of the most important similarity transformations is the *Jordan decomposition*, or *Jordan Canonical Form.*

THEOREM 2.20 (Jordan Decomposition; see [**22**, Theor. 7.1.9].). *Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ with exactly p distinct eigenvalues $\lambda_1, \dots, \lambda_p \in \mathbb{C}$ for $p \le n$. Then, there exists a nonsingular matrix $\boldsymbol{X} \in \mathbb{C}^{n\times n}$ such that*

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \begin{bmatrix} \boldsymbol{J}_1(\lambda_1) & & \\ & \ddots & \\ & & \boldsymbol{J}_p(\lambda_p) \end{bmatrix}.$$

Each block $\boldsymbol{J}_i(\lambda_i)$ has the block diagonal structure

$$\boldsymbol{J}_i(\lambda_i) = \begin{bmatrix} \boldsymbol{J}_{i,1}(\lambda_i) & & \\ & \ddots & \\ & & \boldsymbol{J}_{i,m_i^g}(\lambda_i) \end{bmatrix} \in \mathbb{C}^{m_i^a \times m_i^a}$$

with

$$\boldsymbol{J}_{i,k}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_i & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{m_{i,k} \times m_{i,k}},$$

where m_i^a and m_i^g are the algebraic and geometric multiplicity of the eigenvalue λ_i. Each of the subblocks $\boldsymbol{J}_{i,k}(\lambda_i)$ is referred to as a Jordan block.

Unfortunately, from a computational point of view, the computation of the Jordan Canonical Form is numerically unstable.

An important and practical factorization is the *QR decomposition*:

Definition 2.21 (QR Decomposition; see [**22**, Theor. 5.2.1].). Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Then, there exists an orthogonal $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ and an upper triangular $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ such that

$$\boldsymbol{A} = \boldsymbol{QR}.$$

This concludes the theoretical part of the background study. Next, we are getting started with computational aspects.

2.2. First computational aspects. This section quickly reviews aspects of perturbation theory and illustrates possible difficulties in computing eigenvalues accurately. This is followed by a brief overview of different classes of methods for solving eigenvalue problems. Details about all mentioned methods are given in the upcoming sections.

First of all, it should be clear that in general we must iterate to find eigenvalues of a matrix: According to Definition 2.5, the eigenvalues of a matrix $\boldsymbol{A}$ are the roots of the characteristic polynomial $p_{\boldsymbol{A}}(x)$. In 1824, Abel proved that for polynomials of degree $n \geq 5$, there is no formula for its roots in terms of its coefficients that uses only the operations of addition, subtraction, multiplication, division, and taking kth roots. Hence, even if we could work in exact arithmetic, no computer would produce the exact roots of an arbitrarily polynomial in a finite number of steps. (This is different than direct methods for solving systems of linear equations such as Gaussian elimination.) Hence, computing the eigenvalues of any $n \times n$ matrix $\boldsymbol{A}$ requires an iterative process if $n \geq 5$.

As already indicated in the previous section, many methods are based on repeatedly performing *similarity transformations* to bring $\boldsymbol{A}$ into a simpler equivalent form. This typically means generating as many zero entries in the matrix as possible. The goal is eventually to perform a Schur decomposition. If the matrix is normal, then the Schur decomposition simplifies to a diagonal matrix (not only an upper triangular matrix), and this has implications in terms of stability of numerical computations. As part of the process, we often aim to reduce the matrix into tridiagonal form (symmetric case) or upper Hessenberg form (nonsymmetric case). *Deflation*, *projection*, and other tools can be incorporated and are extremely valuable.

Now, let us focus on the reduction of a matrix $\boldsymbol{A}$ to upper Hessenberg form. One way to accomplish this is the use of *Householder reflectors* (also called *Householder* *transformations*). They can be used to zero out selected components of a vector. Hence, by performing a sequence of Householder reflections on the columns of $\boldsymbol{A}$, we can transform $\boldsymbol{A}$ into a simpler form. Householder reflectors are matrices of the form

$$\boldsymbol{P} = \boldsymbol{I} - \frac{2}{\boldsymbol{v}^*\boldsymbol{v}}\boldsymbol{v}\boldsymbol{v}^*,$$

where $\boldsymbol{v} \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}$. Householder matrices are Hermitian ($\boldsymbol{P} = \boldsymbol{P}^*$), unitary, and numerically stable. Geometrically, $\boldsymbol{P}$ applied to a vector $\boldsymbol{x}$ reflects it about the hyperplane $\operatorname{span}\{\boldsymbol{v}\}^\perp$.

Assume we want to bring $\boldsymbol{A}$ into upper Hessenberg form. Then, the first step is to introduce zeros into all except the first two entries of the first column of $\boldsymbol{A}$. Let us denote by $\boldsymbol{x} = [a_{2,1}, \dots, a_{n,1}]^T$ the part of the first column of $\boldsymbol{A}$ under consideration. We are looking for a vector $\boldsymbol{v} \in \mathbb{C}^{n-1} \setminus \{\boldsymbol{0}\}$ such that $\boldsymbol{P}\boldsymbol{x}$ results in a multiple of the first unit vector $\boldsymbol{e}_1$. This can be achieved with the ansatz $\boldsymbol{v} = \boldsymbol{x} \pm \|\boldsymbol{x}\|_2\boldsymbol{e}_1$, since this yields

$$\boldsymbol{P}\boldsymbol{x} = \mp\|\boldsymbol{x}\|_2\boldsymbol{e}_1.$$

Let us illustrate the action of $\boldsymbol{P}$ to the first column of $\boldsymbol{A}$:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & \cdots & a_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \rightarrow \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \mp\|\boldsymbol{x}\|_2 & a_{2,2} & \cdots & a_{2,n} \\ 0 & a_{3,2} & \cdots & a_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}.$$

Note that the first step is not complete yet: Remember that, for a similarity transformation, we need to apply the Householder matrix twice, i.e., $\boldsymbol{P}^*\boldsymbol{A}\boldsymbol{P}$. Note that the right multiplication with $\boldsymbol{P}$ does not destroy the zeros:

$$\underset{\boldsymbol{A}}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} \xrightarrow{\boldsymbol{P}^*\cdot} \underset{\boldsymbol{P}^*\boldsymbol{A}}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix}} \xrightarrow{\cdot\boldsymbol{P}} \underset{\boldsymbol{P}^*\boldsymbol{A}\boldsymbol{P}}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix}}.$$

Let us denote the Householder matrix in the first step by $\boldsymbol{P}_1$. The above procedure is repeated with the Householder matrix $\boldsymbol{P}_2$ to the second column of $\boldsymbol{P}_1^*\boldsymbol{A}\boldsymbol{P}_1$, then to the third column of $\boldsymbol{P}_2^*\boldsymbol{P}_1^*\boldsymbol{A}\boldsymbol{P}_1\boldsymbol{P}_2$ with the Householder matrix $\boldsymbol{P}_3$, and so on, until we end up with a matrix in upper Hessenberg form as given in Definition 2.19. Let us denote the Householder matrix in step i by $\boldsymbol{P}_i$. After $n-2$ steps, we obtain the upper Hessenberg form:

$$\underbrace{\boldsymbol{P}_{n-2}^* \cdots \boldsymbol{P}_1^*}_{\boldsymbol{P}^*}\, \boldsymbol{A}\, \underbrace{\boldsymbol{P}_1 \cdots \boldsymbol{P}_{n-2}}_{\boldsymbol{P}} = \boldsymbol{H} = \begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{bmatrix}.$$

Remark 2.22. Here are a few additional comments about the process:

- In practice, one choses $\boldsymbol{v} = \boldsymbol{x} + \operatorname{sign}(x_1)\|\boldsymbol{x}\|_2\boldsymbol{e}_1$, where x_1 is the first entry of the vector $\boldsymbol{x}$ under consideration.
- Note that in each step i, the corresponding vector $\boldsymbol{x}_i$, and hence $\boldsymbol{v}_i$ and $\boldsymbol{P}_i$, shrink by one in size.
- The reduction of an $n \times n$ matrix to upper Hessenberg form via Householder reflections requires $\mathcal{O}(n^3)$ operations.

One may ask why do we first bring $\boldsymbol{A}$ to upper Hessenberg form and not immediately to triangular form using Householder reflections? In that case, the right multiplication with $\boldsymbol{P}$ would destroy the zeros previously introduced:

$$\underset{\boldsymbol{A}}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}} \xrightarrow{\boldsymbol{P}^* \cdot} \underset{\boldsymbol{P}^*\boldsymbol{A}}{\begin{bmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix}} \xrightarrow{\cdot \boldsymbol{P}} \underset{\boldsymbol{P}^*\boldsymbol{A}\boldsymbol{P}}{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}}.$$

This should not come as a surprise: we already knew from Abel (1824) that it is impossible to obtain a Schur form of $\boldsymbol{A}$ in a finite number of steps; see the beginning of this Section.

Remark 2.23. If $\boldsymbol{A}$ is symmetric, the reduction to upper Hessenberg form turns into a tridiagonal matrix. That is because the right multiplication with $\boldsymbol{P}_i$ also introduces zeros above the diagonal:

$$\underset{\boldsymbol{A}}{\begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \cdots & * \end{bmatrix}} \xrightarrow{\boldsymbol{P}_1^* \cdot} \underset{\boldsymbol{P}_1^*\boldsymbol{A}}{\begin{bmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}} \xrightarrow{\cdot \boldsymbol{P}_1} \underset{\boldsymbol{P}_1^*\boldsymbol{A}\boldsymbol{P}_1}{\begin{bmatrix} * & * & 0 & \cdots & 0 \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}}.$$

Later on, we will discuss fast algorithms for eigenvalue problems with symmetric tridiagonal matrices.

Remark 2.24. The Hessenberg reduction via Householder reflections is backward stable, i.e., there exists a small perturbation $\delta\boldsymbol{A}$ of $\boldsymbol{A}$ such that

$$\hat{\boldsymbol{H}} = \hat{\boldsymbol{P}}^*(\boldsymbol{A} + \delta\boldsymbol{A})\hat{\boldsymbol{P}}, \quad \|\delta\boldsymbol{A}\|_F \le cn^2\epsilon_{\text{machine}}\|\boldsymbol{A}\|_F.$$

Here, $\hat{\boldsymbol{H}}$ is the computed upper Hessenberg matrix, $\hat{\boldsymbol{P}} = \hat{\boldsymbol{P}}_1 \cdots \hat{\boldsymbol{P}}_{n-2}$ is a product of exactly unitary Householder matrices based on computed vectors $\hat{\boldsymbol{v}}_i$, and $c > 0$ a constant. For more details, we refer to [**66**, p. 351] and [**27**, Sec. 19.3].

The command in Matlab for obtaining an upper Hessenberg matrix is `[P,H] = hess(A)`. During the next sections, we will see how the upper Hessenberg form (or the tridiagonal form in case of symmetric matrices) is used within eigenvalue solvers.

Before we talk about algorithms, we need to understand when it is difficult to compute eigenvalues accurately. The following example shows that eigenvalues of a matrix are continuous (but not necessarily differentiable) functions of it.

Example 2.25. Consider the perturbed Jordan block

$$\boldsymbol{A}(\varepsilon) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \varepsilon & & & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The characteristic polynomial is given as $p_{\boldsymbol{A}(\varepsilon)}(x) = (-1)^n(x^n - \varepsilon)$. Hence, the eigenvalues are $\lambda_j(\varepsilon) = \varepsilon^{\frac{1}{n}} \exp(\frac{2\imath j\pi}{n})$ for $j = 1, \dots, n$. None of the eigenvalues is differentiable at $\varepsilon = 0$. Their rate of change at the origin is infinite. Consider for instance the case $n = 20$ and $\varepsilon = 10^{-16}$ (machine precision), then $\lambda_1(\varepsilon) = 0.1507 + 0.0490\imath$ whereas $\lambda(0) = 0$.

Let us quickly address the issue of estimating the quality of computed eigenvalues. The question here is: How do eigenvalues and eigenvectors vary when the original matrix undergoes small perturbations? We start with considering the sensitivity of simple eigenvalues.

Theorem 2.26 (See, e.g. [**22**, Chap. 7.2.2].). *Let $\boldsymbol{A} \in \mathbb{C}^{n \times n}$ with a simple eigenvalue λ, a right (unit norm) eigenvector $\boldsymbol{x}$, and a left (unit norm) eigenvector $\boldsymbol{y}$. Let $\boldsymbol{A} + \delta\boldsymbol{A}$ be a perturbation of $\boldsymbol{A}$ and $\lambda + \delta\lambda$ the corresponding perturbed eigenvalue. Then*

$$\delta\lambda = \frac{\boldsymbol{y}^* \delta\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{y}^*\boldsymbol{x}} + \mathcal{O}\left(\|\delta\boldsymbol{A}\|_2^2\right).$$

The condition number of λ is defined as $s(\lambda) = \frac{1}{|\boldsymbol{y}^*\boldsymbol{x}|}$. It can be shown that

$$s(\lambda) = \frac{1}{\cos\left(\theta(\boldsymbol{x}, \boldsymbol{y})\right)},$$

where $\theta(\boldsymbol{x}, \boldsymbol{y})$ is the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

In general, $\mathcal{O}(\varepsilon)$ perturbations in $\boldsymbol{A}$ can induce $\frac{\varepsilon}{s(\lambda)}$ changes in an eigenvalue. Thus, if $s(\lambda)$ is small, then λ is ill-conditioned, and $\boldsymbol{A}$ is "close to" a matrix with multiple eigenvalues. If $\boldsymbol{A}$ is normal, then every simple eigenvalue satisfies $s(\lambda) = 1$, which means that these eigenvalues are well-conditioned. In the case of a multiple eigenvalue λ, $s(\lambda)$ is not unique anymore. For a defective eigenvalue λ, it holds in general that $\mathcal{O}(\varepsilon)$ perturbations in $\boldsymbol{A}$ can result in $\mathcal{O}\left(\varepsilon^{\frac{1}{p}}\right)$ changes in λ, where p denotes the size of the largest Jordan block associated with λ. This is the effect we have observed in Example 2.25: $\boldsymbol{A}(0)$ has the effective eigenvalue zero with algebraic multiplicity n and geometric multiplicity one. Hence, $\mathcal{O}(10^{-16})$ perturbations in $\boldsymbol{A}$ can result in $\mathcal{O}\left(10^{-\frac{16}{20}}\right) = \mathcal{O}(0.1585)$ changes in the eigenvalue. In other words, small perturbations in the input data caused a large perturbation in the output. This can lead to numerical instabilities of eigenvalue solvers.

In the following, we start with algorithms for computing a few up to all eigenvalues for small to moderate-sized matrices. Then, we continue with large and sparse matrices.

3. Small to moderate-sized matrices

In general, as previously stated, we may separate methods into ones that are based on matrix decompositions vs. ones that are based on matrix-vector products. The power method, which we start with in the sequel, is an important building block for both classes of methods. It is based on matrix-vector products, but it is invaluable for eigensolvers based on decompositions. We choose to include it in this section, noting that it is relevant also for eigensolvers for large and sparse matrices.

3.1. Power method.

The power method is one of the oldest techniques for solving eigenvalue problems. It is used for computing a *dominant eigenpair*, i.e., the eigenvalue of maximum modulus of a matrix $\boldsymbol{A}$ and a corresponding eigenvector. The algorithm consists of generating a sequence of matrix-vector multiplications $\{\boldsymbol{A}^k\boldsymbol{v}_0\}_{k=0,1,\dots}$, where $\boldsymbol{v}_0$ is some nonzero initial vector.

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{A}\boldsymbol{x}_j = \lambda_j\boldsymbol{x}_j$ for $j = 1,\dots,n$. Assume that the eigenvectors $\boldsymbol{x}_j, j = 1,\dots,n$, are linearly independent, i.e., $\boldsymbol{A}$ is nondefective. Given $\boldsymbol{0} \neq \boldsymbol{v}_0 \in \mathbb{C}^n$, we can expand it using the eigenvectors of $\boldsymbol{A}$ to

$$\boldsymbol{v}_0 = \sum_{j=1}^{n} \beta_j \boldsymbol{x}_j,$$

where $\beta_j \in \mathbb{C}$ for $j = 1,\dots,n$. Applying $\boldsymbol{A}$ to $\boldsymbol{v}_0$ yields

$$\boldsymbol{A}\boldsymbol{v}_0 = \sum_{j=1}^{n} \beta_j \boldsymbol{A}\boldsymbol{x}_j = \sum_{j=1}^{n} \beta_j \lambda_j \boldsymbol{x}_j.$$

Hence, the eigenvectors corresponding to eigenvalues of larger modulus are favored. The above procedure can be repeated. In fact, for any $k \in \mathbb{N}$, we have

$$\boldsymbol{A}^k\boldsymbol{v}_0 = \sum_{j=1}^{n} \beta_j \boldsymbol{A}^k\boldsymbol{x}_j = \sum_{j=1}^{n} \beta_j \lambda_j^k \boldsymbol{x}_j.$$

In order for the following algorithm to converge, we need the following assumptions: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$. Since we are interested in the eigenvalue of maximum modulus, we need some distance to the remaining eigenvalues. λ_1 is called the *dominant eigenvalue.* We further need $\boldsymbol{v}_0$ to have a component in the direction of the eigenvector corresponding to λ_1, i.e., $\beta_1 \neq 0$. Note that this assumption is less concerning in practice since rounding errors during the iteration typically introduce components in the direction of $\boldsymbol{x}_1$. However, we need it for the following theoretical study. Due to $\beta_1 \neq 0$, we can write

$$\boldsymbol{A}^k\boldsymbol{v}_0 = \beta_1\lambda_1^k\boldsymbol{x}_1 + \sum_{j=2}^{n} \beta_j\lambda_j^k\boldsymbol{x}_j = \beta_1\lambda_1^k\left(\boldsymbol{x}_1 + \sum_{j=2}^{n}\frac{\beta_j}{\beta_1}\left(\frac{\lambda_j}{\lambda_1}\right)^k\boldsymbol{x}_j\right).$$

Since λ_1 is a dominant eigenvalue, we get $\left(\frac{\lambda_j}{\lambda_1}\right)^k \overset{k\to\infty}{\longrightarrow} 0$ for all $j = 2,\dots,n$. Hence, it can be shown (cf. [**47**, Theor. 4.1]) that $\boldsymbol{A}^k\boldsymbol{v}_0$, as well as the scaled version $\boldsymbol{v}_k = \frac{\boldsymbol{A}^k\boldsymbol{v}_0}{\|\boldsymbol{A}^k\boldsymbol{v}_0\|_2}$ which is used in practice to avoid overflow/underflow, converges linearly to a multiple of $\boldsymbol{x}_1$ with a convergence rate proportional to $\frac{|\lambda_2|}{|\lambda_1|}$. A value of $\frac{|\lambda_2|}{|\lambda_1|} \approx 1$ indicates a slow convergence behavior. Algorithm 3.1 shows the power method. The approximated eigenvalue in step k

$$\lambda_1^{(k)} = \boldsymbol{v}_k^T \boldsymbol{A} \boldsymbol{v}_k$$

is computed using the *Rayleigh quotient*; see Definition 2.6. This is based on the following: Given a vector $\hat{\boldsymbol{x}}_1$ that approximates the eigenvector $\boldsymbol{x}_1$. Then, $\hat{\lambda}_1 = \hat{\boldsymbol{x}}_1^T \boldsymbol{A} \hat{\boldsymbol{x}}_1$ is the best eigenvalue approximation in the least-squares sense, i.e.,

$$\hat{\lambda}_1 = \arg\min_{\mu} \|\boldsymbol{A}\hat{\boldsymbol{x}}_1 - \mu \hat{\boldsymbol{x}}_1\|_2^2. \tag{3.1}$$

We can solve this minimization problem by solving the *normal equation*

$$\begin{aligned} \hat{\boldsymbol{x}}_1^T \hat{\boldsymbol{x}}_1 \, \mu &= \hat{\boldsymbol{x}}_1^T \boldsymbol{A} \hat{\boldsymbol{x}}_1 \\ \Leftrightarrow \mu &= \frac{\hat{\boldsymbol{x}}_1^T \boldsymbol{A} \hat{\boldsymbol{x}}_1}{\hat{\boldsymbol{x}}_1^T \hat{\boldsymbol{x}}_1}; \end{aligned}$$

see, e.g., [**46**, Chap. 5.3.3]. Since we normalize the computed eigenvectors in the power method, i.e., $\|\hat{\boldsymbol{x}}_1\|_2 = 1$, we get the desired result. The cost for k iterations

Algorithm 3.1: Power method

```
1 Choose $\boldsymbol{v}_0 = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$
2 for $k = 1, 2, \ldots,$ until termination do
3     $\tilde{\boldsymbol{v}} = \boldsymbol{A}\boldsymbol{v}_{k-1}$
4     $\boldsymbol{v}_k = \frac{\tilde{\boldsymbol{v}}}{\|\tilde{\boldsymbol{v}}\|_2}$
5     $\lambda_1^{(k)} = \boldsymbol{v}_k^T \boldsymbol{A} \boldsymbol{v}_k$
6 end
```

is $\mathcal{O}(2kn^2)$ floating point operations (flops).

The power method can be applied to large, sparse, or implicit matrices. It is simple and basic but can be slow. We assumed for the convergence that $\boldsymbol{A}$ is nondefective. For the case of a defective $\boldsymbol{A}$, the power method can still be applied but converges even more slowly; see, e.g., [**28**]. Moreover, we want to emphasize again that the power method only works if the matrix under consideration has *one* dominant eigenvalue. This excludes the case of, e.g., a dominant complex eigenvalue[3] or of dominant eigenvalues of opposite signs. The power method is rather used as a building block for other, more robust and general algorithms. We refer to [**66**, Chap. 10] for a detailed discussion of the power method. Next, we discuss a method that overcomes the mentioned difficulties.

3.2. Inverse power method. We have seen that the power method is in general slow. Moreover, it is good only for one well-separated dominant eigenvalue. How can we accelerate it, and what about the more general case of looking for a non-dominant eigenpair? The inverse power method uses *shift and invert* techniques to overcome these limitations of the power method. It aims to compute the eigenvalue of $\boldsymbol{A}$ that is closest to a certain scalar (shift) and a corresponding eigenvector. It also enhances the convergence behavior. The price for these improvements is the

[3]As noted in Section 2.1, eigenvalues of real matrices always occur in complex conjugate pairs.

solution of a linear system in each iteration.

The idea is the following: Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ has eigenpairs $(\lambda_j, \boldsymbol{x}_j)_{j=1,\dots,n}$ with $|\lambda_1| \geq \ldots \geq |\lambda_n|$. Let $\alpha \in \mathbb{R}$ with $\alpha \neq \lambda_j$ for $j = 1,\dots,n$. This will be the *shift* in the inverse power method. In practice, we choose $\alpha \approx \lambda_i$ for some i depending on which (real) eigenvalue λ_i we want to find. Hence, in order for the method to work, we need to know approximately the value of the eigenvalue we are interested in. Then, $\boldsymbol{A}-\alpha\boldsymbol{I}$ has eigenpairs $(\lambda_j-\alpha, \boldsymbol{x}_j)_{j=1,\dots,n}$, and $(\boldsymbol{A}-\alpha\boldsymbol{I})^{-1}$ has eigenpairs $(\mu_j, \boldsymbol{x}_j)_{j=1,\dots,n}$ with $\mu_j = (\lambda_j-\alpha)^{-1}$. Let λ_i and λ_j be the two eigenvalues that are closest to α with $|\lambda_i-\alpha| < |\lambda_j-\alpha|$. Then, the two largest eigenvalues μ_1 and μ_2 of $(\boldsymbol{A}-\alpha\boldsymbol{I})^{-1}$ are

$$\mu_1 = \frac{1}{\lambda_i-\alpha}, \quad \mu_2 = \frac{1}{\lambda_j-\alpha}.$$

Hence, the power method applied to $(\boldsymbol{A}-\alpha\boldsymbol{I})^{-1}$ converges to μ_1 and an eigenvector of μ_1 with convergence rate

$$\frac{|\mu_2|}{|\mu_1|} = \frac{\frac{1}{|\lambda_j-\alpha|}}{\frac{1}{|\lambda_i-\alpha|}} = \frac{|\lambda_i-\alpha|}{|\lambda_j-\alpha|}.$$

We know from the previous section that we need a small value of $\frac{|\mu_2|}{|\mu_1|}$ in order to converge fast. Hence, we desire $|\lambda_i-\alpha| \ll |\lambda_j-\alpha|$, which requires a "good" choice of the shift α. If we are interested for instance in the dominant eigenvalue, estimations based on norms of $\boldsymbol{A}$ can be used; see, e.g., [**22**, Chap. 2.3.2].

Algorithm 3.2: Inverse power method

```
1 Choose $\boldsymbol{v}_0 = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$
2 for $k = 1,2,\dots,$ until termination do
3     Solve $(\boldsymbol{A}-\alpha\boldsymbol{I})\tilde{\boldsymbol{v}} = \boldsymbol{v}_{k-1}$
4     $\boldsymbol{v}_k = \frac{\tilde{\boldsymbol{v}}}{\|\tilde{\boldsymbol{v}}\|_2}$
5     $\lambda^{(k)} = \boldsymbol{v}_k^T\boldsymbol{A}\boldsymbol{v}_k$
6 end
```

As already mentioned at the beginning of this section, the price for overcoming difficulties of the power method by using a shift and invert approach is the solution of a linear system in every iteration. If α is fixed, then we have to solve linear systems with one matrix and many right-hand sides: If a direct method can be applied, then we form an LU decomposition of $\boldsymbol{A}-\alpha\boldsymbol{I}$ once. The cost for solving the two triangular systems arising from the LU decomposition is $\mathcal{O}(n^2)$. For huge problems, iterative methods have to be employed to solve the linear systems. This pays off only if the inverse iteration converges very fast.

In summary, we have seen that we can apply the inverse power method to find different eigenvalues using different shifts. During the whole iteration, the inverse power method uses a fixed shift α. The next method involves a dynamic shift α_k.

3.3. Rayleigh quotient iteration. The idea of the Rayleigh quotient iteration is to learn the shift as the iteration proceeds using the calculated eigenvalue

$\lambda^{(k-1)}$ from the previous step $k-1$. Now, each iteration is potentially more expensive since the linear systems involve different matrices in each step. However, the new algorithm may converge in many fewer iterations. In the Hermitian case, we potentially obtain a cubic convergence rate; see, e.g., [**39**].

Note that the matrix $(\boldsymbol{A}-\lambda^{(k-1)}\boldsymbol{I})$ may be singular. This is the case when the shift hits an eigenvalue of $\boldsymbol{A}$. The cost for solving the linear system with $(\boldsymbol{A}-\lambda^{(k-1)}\boldsymbol{I})$ is $\mathcal{O}(n^3)$ if $\boldsymbol{A}$ is full. For an upper Hessenberg matrix, it reduces to $\mathcal{O}(n^2)$, and for a tridiagonal matrix even to $\mathcal{O}(n)$.

Next, we discuss a technique that uses information of a computed dominant eigenpair for the approximation of a second-dominant eigenpair.

3.4. Deflation. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ have eigenvalues $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, right eigenvectors $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$, and left eigenvectors $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n$. Note that $(\lambda_1,\boldsymbol{x}_1)$ is a dominant eigenpair. Suppose we have approximated the dominant eigenvector $\boldsymbol{x}_1$ of $\boldsymbol{A}$ by $\hat{\boldsymbol{x}}_1$ with $\|\hat{\boldsymbol{x}}_1\|_2 = 1$. Now, we are interested in approximating the next eigenvalue λ_2.

Deflation is based on a simple rank-one modification of $\boldsymbol{A}$, as follows: Compute $\boldsymbol{A}_1 = \boldsymbol{A} - \alpha\hat{\boldsymbol{x}}_1\boldsymbol{w}^T$, where $\alpha \in \mathbb{R}$ is an appropriate shift and $\boldsymbol{w} \in \mathbb{R}^n$ an arbitrary vector such that $\boldsymbol{w}^T\hat{\boldsymbol{x}}_1 = 1$.

THEOREM 3.1 (Wielandt; see [**65**].). *In the case $\hat{\boldsymbol{x}}_1 = \boldsymbol{x}_1$, the eigenvalues of $\boldsymbol{A}_1$ are $\lambda_1-\alpha,\lambda_2,\ldots,\lambda_n$. Moreover, the right eigenvector $\boldsymbol{x}_1$ and the left eigenvectors $\boldsymbol{y}_2,\ldots,\boldsymbol{y}_n$ are preserved.*

PROOF.

$$\left(\boldsymbol{A} - \alpha\boldsymbol{x}_1\boldsymbol{w}^T\right)\boldsymbol{x}_1 = \boldsymbol{A}\boldsymbol{x}_1 - \alpha\boldsymbol{x}_1\boldsymbol{w}^T\boldsymbol{x}_1 = \lambda_1\boldsymbol{x}_1 - \alpha\boldsymbol{x}_1$$

since $\boldsymbol{w}^T\boldsymbol{x}_1 = 1$. For $i = 2,\ldots,n$, we have

$$\boldsymbol{y}_i^*\left(\boldsymbol{A} - \alpha\boldsymbol{x}_1\boldsymbol{w}^T\right) = \boldsymbol{y}_i^*\boldsymbol{A} - \alpha\boldsymbol{y}_i^*\boldsymbol{x}_1\boldsymbol{w}^T = \boldsymbol{y}_i^*\boldsymbol{A} = \lambda_i\boldsymbol{y}_i^*$$

since $\boldsymbol{y}_i^*\boldsymbol{x}_1 = 0$ for $i = 2,\ldots,n$. □

Hence, a modification of $\boldsymbol{A}$ to $\boldsymbol{A}_1 = \boldsymbol{A} - \alpha\hat{\boldsymbol{x}}_1\boldsymbol{w}^T$ displaces the dominant eigenvalue of $\boldsymbol{A}$. The rank-one modification should be chosen such that λ_2 becomes the dominant eigenvalue of $\boldsymbol{A}_1$. We can then proceed for instance with the power method applied to $\boldsymbol{A}_1$ in order to obtain an approximation of λ_2. This technique is called *Wielandt deflation.*

There are many ways to choose $\boldsymbol{w}$. A simple choice (due to Hotelling [**29**]) is to choose $\boldsymbol{w} = \boldsymbol{y}_1$ the first left eigenvector (or an approximation of it) or $\boldsymbol{w} = \boldsymbol{x}_1$ or rather its approximation $\boldsymbol{w} = \hat{\boldsymbol{x}}_1$. It can be shown (cf. [**47**, Chap. 4.2.2]) that if $\boldsymbol{x}_1$ is nearly orthogonal to $\boldsymbol{x}_2$ or if $\frac{\lambda_1-\lambda_2}{\alpha} \ll 1$, the choice $\boldsymbol{w} = \boldsymbol{x}_1$ is nearly optimal in terms of eigenvalue conditioning.

Note that we never need to form the matrix $\boldsymbol{A}_1$ explicitly. This is important since $\boldsymbol{A}_1$ is a dense matrix. For calculating the matrix-vector product $\boldsymbol{y} = \boldsymbol{A}_1\boldsymbol{x}$, we just need to perform $\boldsymbol{y} \leftarrow \boldsymbol{A}\boldsymbol{x}$, $\beta = \alpha\boldsymbol{w}^T\boldsymbol{x}$, and $\boldsymbol{y} \leftarrow \boldsymbol{y} - \beta\hat{\boldsymbol{x}}_1$. We can apply this procedure recursively without difficulty. However, keep in mind that, for a long deflation process, errors accumulate.

So far, the discussed algorithms compute only one eigenpair at once, i.e., a one-dimensional invariant subspace. Next, we consider another generalization of the power and inverse power method that can be used to compute higher-dimensional invariant subspaces.

3.5. Orthogonal iteration. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ have eigenpairs $(\lambda_j, \boldsymbol{x}_j)_{j=1,\dots,n}$ with $|\lambda_1| \geq \ldots \geq |\lambda_n|$. From the real Schur decomposition in Theorem 2.18, we know there exists an orthogonal $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T},$$

where the diagonal blocks of $\boldsymbol{T}$ correspond to the eigenvalues of $\boldsymbol{A}$ in real form. Assume that the eigenvalues λ_i, represented in $\boldsymbol{T}$ in real form, are ordered from λ_1 to λ_n. Let $1 \leq r < n$. Then, we can do the following partitioning:

$$\boldsymbol{Q} = [\boldsymbol{Q}^{(r)}, \boldsymbol{Q}^{(n-r)}], \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}^{(r,r)} & \boldsymbol{T}^{(r,n-r)} \\ \boldsymbol{0} & \boldsymbol{T}^{(n-r,n-r)} \end{bmatrix},$$

where $\boldsymbol{Q}^{(r)} \in \mathbb{R}^{r\times r}$ and $\boldsymbol{T}^{(r,r)} \in \mathbb{R}^{r\times r}$. Note that r should be chosen such that the $(r+1, r)$ entry in $\boldsymbol{T}$ is zero, i.e., we do not split a complex conjugate eigenpair to $\boldsymbol{T}^{(r,r)}$ and $\boldsymbol{T}^{(n-r,n-r)}$. Then,

$$\boldsymbol{A}\boldsymbol{Q}^{(r)} = \boldsymbol{Q}^{(r)}\boldsymbol{T}^{(r,r)},$$

i.e., $\operatorname{ran}(\boldsymbol{Q}^{(r)})$ is an $\boldsymbol{A}$-invariant subspace corresponding to the r largest (in modulus) eigenvalues. Due to this property, this subspace is also called *dominant.*

Now, we are interested in computing such a dominant r-dimensional invariant subspace. Hence, instead of dealing with a matrix-vector product as in the algorithms before, we go over to a matrix-matrix product, i.e., we apply $\boldsymbol{A}$ to a few vectors simultaneously. This can be achieved by the orthogonal iteration presented in Algorithm 3.3.

Algorithm 3.3: Orthogonal iteration

```
1 Choose $\boldsymbol{Q}_0 \in \mathbb{R}^{n\times r}$ with orthonormal columns
2 for $k = 1, 2, \ldots,$ until termination do
3     $\boldsymbol{Z}_k = \boldsymbol{A}\boldsymbol{Q}_{k-1}$
4     $\boldsymbol{Q}_k\boldsymbol{R}_k = \boldsymbol{Z}_k$      (QR factorization)
5 end
```

The QR factorization in Line 4 refers to the QR decomposition in Definition 2.21. It can be computed by, e.g., the modified Gram–Schmidt algorithm in $\mathcal{O}\left(2nr^2\right)$ flops [**22**, Chap. 5.2.8], Householder reflections in $\mathcal{O}\left(2r^2\left(n-\frac{r}{3}\right)\right)$ flops [**22**, Chap. 5.2.2], or Givens transformations in $\mathcal{O}\left(3r^2\left(n-\frac{r}{3}\right)\right)$ flops [**22**, Chap. 5.2.5]. The MATLAB function for computing a (complex) QR decomposition is `[Q,R] = qr(A)` and is based on Householder reflections. Note that the complexity can be reduced if the corresponding matrix is of upper Hessenberg form. We will discuss this further below. Note that Line 3 and 4 yield

$$\boldsymbol{A}\boldsymbol{Q}_{k-1} = \boldsymbol{Q}_k\boldsymbol{R}_k,$$

where $\boldsymbol{R}_k$ is upper triangular. Now, if $|\lambda_r| > |\lambda_{r+1}|$ and $\boldsymbol{Q}_0$ has components in the desired eigendirections, then we have

$$\operatorname{ran}(\boldsymbol{Q}_k) \overset{k\to\infty}{\longrightarrow} \operatorname{ran}(\boldsymbol{Q}^{(r)})$$

with a convergence rate proportional to $\frac{|\lambda_{r+1}|}{|\lambda_r|}$. For more details, we refer to [**22**, Chap. 7.3.2].

Remark 3.2. By replacing the QR factorization in Line 4 of Algorithm 3.3 with $\boldsymbol{Q}_k = \boldsymbol{Z}_k$, we obtain the *subspace iteration*, also called *simultaneous iteration*. Under the same conditions as before, it holds

$$\operatorname{ran}(\boldsymbol{Z}_k) \overset{k\to\infty}{\longrightarrow} \operatorname{ran}(\boldsymbol{Q}^{(r)}).$$

However, the columns of $\boldsymbol{Z}_k$ form an increasingly ill-conditioned basis for $\boldsymbol{A}^k\operatorname{ran}(\boldsymbol{Q}_0)$ since each column of $\boldsymbol{Z}_k$ converges to a multiple of the dominant eigenvector. The orthogonal iteration overcomes this difficulty by orthonormalizing the columns of $\boldsymbol{Z}_k$ at each step.

From the orthogonal iteration, we can derive the QR iteration, a method for finding all eigenvalues of a matrix $\boldsymbol{A}$.

3.6. QR iteration.

We obtain the QR iteration from the orthogonal iteration if we set $r = n$, i.e., we want to compute all eigenvalues, and $\boldsymbol{Q}_0 = \boldsymbol{I}$.

Algorithm 3.4: Prelude to QR iteration

```
1 Choose $Q_0 = I$ (orthogonal)
2 for $k = 1, 2, \dots,$ until termination do
3     $Z_k = AQ_{k-1}$
4     $Q_k R_k = Z_k$      (QR factorization)
5     $A_k = Q_k^T A Q_k$
6 end
```

We can rewrite Algorithm 3.4 by using the following equivalence:

$$\boldsymbol{A}_{k-1} \overset{\text{Line 5}}{=} \boldsymbol{Q}_{k-1}^T \boldsymbol{A} \boldsymbol{Q}_{k-1} \overset{\text{Line 3}}{=} \boldsymbol{Q}_{k-1}^T \boldsymbol{Z}_k \overset{\text{Line 4}}{=} \boldsymbol{Q}_{k-1}^T \boldsymbol{Q}_k \boldsymbol{R}_k =: \tilde{\boldsymbol{Q}}_k \boldsymbol{R}_k, \tag{3.2}$$

$$\boldsymbol{A}_k \overset{\text{Line 5}}{=} \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_{k-1} \boldsymbol{Q}_{k-1}^T \boldsymbol{Q}_k \overset{\text{Line 3}}{=} \boldsymbol{Q}_k^T \boldsymbol{Z}_k \boldsymbol{Q}_{k-1}^T \boldsymbol{Q}_k$$
$$\overset{\text{Line 4}}{=} \boldsymbol{R}_k \boldsymbol{Q}_{k-1}^T \boldsymbol{Q}_k \overset{(3.2)}{=} \boldsymbol{R}_k \tilde{\boldsymbol{Q}}_k.$$

Note that the product of two orthogonal matrices is orthogonal. Hence, $\tilde{\boldsymbol{Q}}_k$ is orthogonal. Therefore, $\boldsymbol{A}_k$ is determined by a QR decomposition of $\boldsymbol{A}_{k-1}$. This form of the QR iteration is presented in Algorithm 3.5. Note that $\boldsymbol{Q}_0$ does not have to be the identity matrix.
From Line 3 and 4 of Algorithm 3.5 we get

$$\boldsymbol{A}_k = (\boldsymbol{Q}_0 \cdots \boldsymbol{Q}_k)^T \boldsymbol{A} (\boldsymbol{Q}_0 \cdots \boldsymbol{Q}_k) =: \hat{\boldsymbol{Q}}_k^T \boldsymbol{A} \hat{\boldsymbol{Q}}_k,$$

where $\hat{\boldsymbol{Q}}_k$ is orthogonal. Hence, $\operatorname{ran}(\hat{\boldsymbol{Q}}_k)$ is an $\boldsymbol{A}$-invariant subspace and $\lambda(\boldsymbol{A}) = \lambda(\boldsymbol{A}_k)$. If $|\lambda_1| > \ldots > |\lambda_n|$ and $\boldsymbol{Q}_0$ has components in the desired eigendirections, then we have

$$\operatorname{ran}(\hat{\boldsymbol{Q}}_k(:, 1:l)) \overset{k\to\infty}{\longrightarrow} \operatorname{ran}(\boldsymbol{Q}(:, 1:l)) \quad \forall 1 \le l \le n$$

Algorithm 3.5: QR iteration

```
1 $\boldsymbol{A}_0 = \boldsymbol{Q}_0^T \boldsymbol{A} \boldsymbol{Q}_0$ (real Schur form, $\boldsymbol{Q}_0 \in \mathbb{R}^{n \times n}$ orthogonal)
2 for $k = 1, 2, \ldots,$ until termination do
3     $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{A}_{k-1}$     (QR factorization)
4     $\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k$
5 end
```

with a convergence rate proportional to $\frac{|\lambda_{l+1}|}{|\lambda_l|}$. Hence, $\boldsymbol{A}_k \overset{k\to\infty}{\longrightarrow} \boldsymbol{T}$, where $\boldsymbol{Q}^T \boldsymbol{A} \boldsymbol{Q} = \boldsymbol{T}$ is a real Schur decomposition of $\boldsymbol{A}$. For more details, we refer to [**22**, Chap. 7.3.3].

Overall, the QR iteration computes the Schur form of a matrix. As in the previous section, we considered the real Schur form here. But note that if we allow complex arithmetic, we get the same results with a (complex) Schur form. For further readings on the QR iteration we refer to, e.g., [**62, 55, 32**].

As already mentioned in the previous section, in this form the cost of each step of the QR iteration is $\mathcal{O}(n^3)$. But we can reduce the complexity if we start with $\boldsymbol{A}_0$ in upper Hessenberg form. Moreover, we can speed up the convergence using shifts.

3.7. QR iteration with shifts. If we choose $\boldsymbol{Q}_0$ such that $\boldsymbol{A}_0$ is in upper Hessenberg form, the cost of each step of the QR iteration reduces to $\mathcal{O}(n^2)$. If $\boldsymbol{A}$ is symmetric, then the cost per step is $\mathcal{O}(n)$. It can be shown that each $\boldsymbol{A}_k$ is upper Hessenberg. This is the first modification. Second, shifts $\zeta_k \in \mathbb{R}$ are introduced in order to accelerate the deflation process (see Theorem 2.16). Deflation occurs every time $\boldsymbol{A}_k$ is reduced, i.e., at least one of its subdiagonal entries is zero. In such a case, we continue with two smaller subproblems. The matrices $\boldsymbol{A}_{k-1}$ and $\boldsymbol{A}_k$ in

Algorithm 3.6: QR iteration with shifts

```
1 $\boldsymbol{A}_0 = \boldsymbol{Q}_0^T \boldsymbol{A} \boldsymbol{Q}_0$ upper Hessenberg form (Tridiagonal if $\boldsymbol{A}$ is symmetric)
2 for $k = 1, 2, \ldots,$ until termination do
3     $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{A}_{k-1} - \zeta_k \boldsymbol{I}$     (QR factorization)
4     $\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k + \zeta_k \boldsymbol{I}$
5 end
```

Algorithm 3.6 are orthogonally similar since

$$\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k + \zeta_k \boldsymbol{I} = \boldsymbol{Q}_k^T \left(\boldsymbol{Q}_k \boldsymbol{R}_k + \zeta_k \boldsymbol{I} \right) \boldsymbol{Q}_k = \boldsymbol{Q}_k^T \boldsymbol{A}_{k-1} \boldsymbol{Q}_k,$$

and $\boldsymbol{Q}_k$ from the QR decomposition is orthogonal.

Why does the shift strategy work? If ζ_k is an eigenvalue of the unreduced Hessenberg matrix $\boldsymbol{A}_{k-1}$, then $\boldsymbol{A}_{k-1} - \zeta_k \boldsymbol{I}$ is singular. This implies $\boldsymbol{R}_k$ is singular, where the (n, n) entry of $\boldsymbol{R}_k$ is zero. Then, the last row of the upper Hessenberg matrix $\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k + \zeta_k \boldsymbol{I}$ consists of zeros except for the (n, n) entry which is ζ_k. So we have converged to the form

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{A}' & \boldsymbol{a} \\ \boldsymbol{0}^T & \zeta_k \end{bmatrix},$$

and can now work on a smaller matrix (deflate) and continue the QR iteration. We can accept the (n, n) entry as ζ_k as it is presumably a good approximation to the eigenvalue. In summary, we obtain deflation after one step in exact arithmetic if we shift by an exact eigenvalue.

If $\zeta = \zeta_k$ for all $k = 1, 2, \ldots$, and we order the eigenvalues λ_i of $\boldsymbol{A}$ such that

$$|\lambda_1 - \zeta| \geq \ldots \geq |\lambda_n - \zeta|,$$

then the pth subdiagonal entry in $\boldsymbol{A}_k$ converges to zero with rate $\frac{|\lambda_{p+1}-\zeta|}{|\lambda_p-\zeta|}$. Of course, we need $|\lambda_{p+1} - \zeta| < |\lambda_p - \zeta|$ in order to get any convergence result.

In practice, deflation occurs whenever a subdiagonal entry $a^{(k)}_{p+1,p}$ of $\boldsymbol{A}_k$ is small enough, e.g., if

$$|a^{(k)}_{p+1,p}| \leq c\epsilon_{\text{machine}}(|a^{(k)}_{p,p}| + |a^{(k)}_{p+1,p+1}|)$$

for a small constant $c > 0$.

Let us quickly summarize some shift strategies: The single-shift strategy uses $\zeta_k = a^{(k-1)}_{n,n}$. It can be shown (cf. [**22**, Chap. 7.5.3]) that the convergence $a^{(k)}_{n,n-1} \overset{k\to\infty}{\longrightarrow} 0$ is even quadratic. When we deal with complex eigenvalues, then $\zeta_k = a^{(k-1)}_{n,n}$ tends to be a poor approximation. Then, the double-shift strategy is preferred which performs two single-shift steps in succession, i.e., Lines 3–4 in Algorithm 3.6 are repeated a second time with a second shift. Using implicit QR factorizations, one double-shift step can be implemented with $\mathcal{O}(n^2)$ flops ($\mathcal{O}(n)$ flops in the symmetric case); see e.g., [**22**, Chap. 7.5.5]. This technique was first described by Francis [**18, 19**] and refers to a *Francis QR step.*

The overall QR algorithm requires $\mathcal{O}(n^3)$ flops. For more details about the QR iteration, we refer to, e.g., [**42, 33, 35, 60, 63, 64**]. Further readings concerning shift strategies include [**17, 15, 61**].

We know that the Hessenberg reduction of a symmetric matrix leads to a tridiagonal matrix. In the following, we review methods for this special case.

3.8. Algorithms for symmetric (tridiagonal) matrices.

Before we consider eigenvalue problems for the special case of symmetric tridiagonal matrices, we review one of the oldest methods for symmetric matrices $\boldsymbol{A}$ — *Jacobi's method.* For general symmetric eigenvalue problems, we refer the reader to [**11**, Chap. 5] — it contains important theoretic concepts, e.g., gaps of eigenvalues and the related perturbation theory, and gives a nice overview of direct eigenvalue solvers.

3.8.1. *Jacobi's method.*

Jacobi's method is one of the oldest algorithms [**30**] for eigenvalue problems with a cost of $\mathcal{O}(cn^3)$ flops with a large constant c. However, it is still of current interest due to its parallelizability and accuracy [**12**].

The method is based on a sequence of orthogonal similarity transformations

$$\ldots \boldsymbol{Q}_3^T\boldsymbol{Q}_2^T\boldsymbol{Q}_1^T\boldsymbol{A}\boldsymbol{Q}_1\boldsymbol{Q}_2\boldsymbol{Q}_3\ldots, \tag{3.3}$$

such that each transformation becomes closer to diagonal form. We can write (3.3) as

$$\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1}^T \boldsymbol{A}_k \boldsymbol{Q}_{k+1}, \ k = 0, 1, 2, \ldots$$

with $\boldsymbol{A}_0 = \boldsymbol{A}$. In particular, the orthogonal matrices $\boldsymbol{Q}_i$ are chosen such that the Frobenius norm of the off-diagonal elements

$$\text{off}(\boldsymbol{A}_k) = \sqrt{\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(a_{i,j}^{(k)}\right)^2}$$

is reduced with each transformation. This is done using *Jacobi rotations* (*Givens rotations*)

$$\boldsymbol{J}(p,q,\theta) = \begin{array}{c} \left[\begin{array}{cccccccccc} 1 &&&&&&&&& \\ & \ddots &&&&&&&& \\ && 1 &&&&&&& \\ &&& c & 0 & \cdots & 0 & s &&& \\ &&& 0 & 1 && & 0 &&& \\ &&& \vdots && \ddots && \vdots &&& \\ &&& 0 &&& 1 & 0 &&& \\ &&& -s & 0 & \cdots & 0 & c &&& \\ &&&&&&&& 1 && \\ &&&&&&&&& \ddots & \\ &&&&&&&&&& 1 \end{array}\right] \begin{array}{c} \\ \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \\ \end{array} \\ \begin{array}{cccccccccc} &&& p &&&& q &&& \end{array} \end{array}$$

where $1 \leq p < q \leq n$, $c = \cos(\theta)$, and $s = \sin(\theta)$. Givens rotations are orthogonal. The application of $\boldsymbol{J}(p,q,\theta)^T$ to a vector rotates the vector counterclockwise in the (p,q) coordinate plane by θ radians. In order to make $\boldsymbol{A}_{k+1}$ iteratively more diagonal, $\boldsymbol{Q}_{k+1} = \boldsymbol{J}(p,q,\theta)$ is chosen to make one pair of off-diagonal entries of $\boldsymbol{A}_{k+1} = \boldsymbol{Q}_{k+1}^T \boldsymbol{A}_k \boldsymbol{Q}_{k+1}$ zero at a time. Thereby, θ is chosen such that the (p,q) and (q,p) entry of $\boldsymbol{A}_{k+1}$ become zero. To determine θ, we can consider the corresponding 2×2 system

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a_{p,p}^{(k)} & a_{p,q}^{(k)} \\ a_{q,p}^{(k)} & a_{q,q}^{(k)} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} a_{p,p}^{(k+1)} & 0 \\ 0 & a_{q,q}^{(k+1)} \end{bmatrix},$$

where $a_{p,p}^{(k+1)}$ and $a_{q,q}^{(k+1)}$ are the eigenvalues of

$$\begin{bmatrix} a_{p,p}^{(k)} & a_{p,q}^{(k)} \\ a_{q,p}^{(k)} & a_{q,q}^{(k)} \end{bmatrix}.$$

One can show (cf. [**11**, Chap. 5.3.5]) that $\tan(2\theta) = \frac{2a_{p,q}^{(k)}}{a_{q,q}^{(k)} - a_{p,p}^{(k)}}$ and using this, we can compute c and s. Using the fact that the Frobenius norm is preserved by orthogonal transformations, we obtain (cf. [**11**, Lemma 5.4])

$$\text{off}(\boldsymbol{A}_{k+1})^2 = \text{off}(\boldsymbol{A}_k)^2 - 2\left(a_{p,q}^{(k)}\right)^2, \tag{3.4}$$

i.e., $\boldsymbol{A}_{k+1}$ moves closer to diagonal form with each Jacobi step. In order to maximize the reduction in (3.4), p and q should be chosen such that $|a_{p,q}^{(k)}|$ is maximal. With this choice, we get after k Jacobi steps (cf. [**11**, Theor. 5.11])

$$\operatorname{off}(\boldsymbol{A}_k)^2 \leq \left(1 - \frac{2}{n(n-1)}\right)^k \operatorname{off}(\boldsymbol{A}_0)^2,$$

i.e., convergence at a linear rate. This scheme is the original version from Jacobi in 1846 and is referred to as *classical Jacobi algorithm.* It even can be shown that the asymptotic convergence rate is quadratic (cf. [**11**, Theor. 5.12]); see [**49, 58**]. While the cost for an update is $\mathcal{O}(n)$ flops, the search for the optimal (p, q) costs $\mathcal{O}(n^2)$ flops. For a simpler method, we refer the reader to the *cyclic Jacobi method*; see e.g. [**66**, p. 270] or [**22**, Chap. 8.5.3]. In general, the cost of the cyclic Jacobi method is considerably higher than the cost of the symmetric QR iteration. However, it is easily parallelizable.

Again we want to emphasize that we do not need to tridiagonalize in Jacobi's method. In the following, we discuss two methods that need to start by reducing a symmetric matrix $\boldsymbol{A}$ to tridiagonal form: *bisection* and *divide-and-conquer.* Another method is *MR*^{3} or *MRRR* (Algorithm of Multiple Relatively Robust Representations) [**13**] — a sophisticated variant of the inverse iteration, which is efficient when eigenvalues are close to each other.

3.8.2. *Bisection.* For the rest of Section 3, we consider eigenvalue problems for symmetric tridiagonal matrices of the form

$$\boldsymbol{A} = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}. \tag{3.5}$$

We begin with bisection, a method that can be used to find a subset of eigenvalues, e.g., the largest/smallest eigenvalue or eigenvalues within an interval. Let $\boldsymbol{A}^{(k)} = \boldsymbol{A}(1:k, 1:k)$ be the leading $k \times k$ principal submatrix of $\boldsymbol{A}$ with characteristic polynomial

$$p^{(k)}(x) = \det(\boldsymbol{A}^{(k)} - x\boldsymbol{I})$$

for $k = 1, \ldots, n$. If $b_i \neq 0$ for $i = 1, \ldots, n-1$, then

$$\det(\boldsymbol{A}^{(k)}) = a_k \det(\boldsymbol{A}^{(k-1)}) - b_{k-1}^2 \det(\boldsymbol{A}^{(k-2)}),$$

which yields

$$p^{(k)}(x) = (a_k - x)p^{(k-1)}(x) - b_{k-1}^2 p^{(k-2)}(x)$$

with $p^{(-1)}(x) = 0$ and $p^{(0)}(x) = 1$. Hence, $p^{(n)}(x)$ can be evaluated in $\mathcal{O}(n)$ flops. Given $y < z \in \mathbb{R}$ with $p^{(n)}(y)p^{(n)}(z) < 0$ (Hence, there exists an $w \in (y, z)$ with $p^{(n)}(w) = 0$.), we can use the *method of bisection* (see, e.g., [**22**, Chap. 8.4.1]) to find an approximate root of $p^{(n)}(x)$ and hence an approximate eigenvalue of $\boldsymbol{A}$. The method of bisection converges linearly in the sense that the error is approximately halved at each step.

Assume that the eigenvalues $\lambda_j(\boldsymbol{A}^{(k)})$ of $\boldsymbol{A}^{(k)}$ are ordered as

$$\lambda_1(\boldsymbol{A}^{(k)}) \geq \ldots \geq \lambda_k(\boldsymbol{A}^{(k)}).$$

For computing, e.g., $\lambda_k(\boldsymbol{A})$ for a given k or the largest eigenvalue that is smaller than a given $\mu \in \mathbb{R}$, then we need the following theorem:

THEOREM 3.3 (Sturm Sequence Property; see [**22**, Theor. 8.4.1].). *If $\boldsymbol{A}$ is unreduced, i.e., $b_i \neq 0$ for $i = 1, \ldots, n-1$, then the eigenvalues of $\boldsymbol{A}^{(k-1)}$ strictly separate the eigenvalues of $\boldsymbol{A}^{(k)}$:*

$$\begin{aligned}\lambda_k(\boldsymbol{A}^{(k)}) < \lambda_{k-1}(\boldsymbol{A}^{(k-1)}) < \lambda_{k-1}(\boldsymbol{A}^{(k)})& \\ < \lambda_{k-2}(\boldsymbol{A}^{(k-1)}) < \lambda_{k-2}(\boldsymbol{A}^{(k)}) < \ldots& \\ < \lambda_2(\boldsymbol{A}^{(k)}) < \lambda_1(\boldsymbol{A}^{(k-1)}) < \lambda_1(\boldsymbol{A}^{(k)}).&\end{aligned}$$

Moreover, if $a(\mu)$ denotes the number of sign changes in the sequence

$$\{p^{(0)}(\mu), p^{(1)}(\mu), \ldots, p^{(n)}(\mu)\},$$

where $p^{(k)}(\mu)$ has the opposite sign from $p^{(k-1)}(\mu)$ if $p^{(k)}(\mu) = 0$, then $a(\mu)$ equals the number of $\boldsymbol{A}$'s eigenvalues that are less than μ.

In order to find an initial interval for the method of bisection, we make use of the following simplified version of the *Gershgorin theorem*:

THEOREM 3.4 (Gershgorin). *If $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric, then*

$$\lambda(\boldsymbol{A}) \subseteq \cup_{i=1}^{n} [a_{i,i} - r_i, a_{i,i} + r_i],$$

where $r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{i,j}|$.

The more general version of the Gershgorin theorem can be found, e.g., in [**22**, Theor. 8.1.3]. Suppose we want to compute $\lambda_k(\boldsymbol{A})$, where $\boldsymbol{A}$ is symmetric tridiagonal as in (3.5). Then, from Theorem 3.4, we get $\lambda_k(\boldsymbol{A}) \in [y, z]$ with

$$y = \min_{1 \leq i \leq n} a_i - |b_i| - |b_{i-1}|, \quad z = \max_{1 \leq i \leq n} a_i + |b_i| + |b_{i-1}|$$

and $b_0 = b_n = 0$. Hence, with this choice of y and z, we can reformulate the method of bisection to converge to $\lambda_k(\boldsymbol{A})$; see, e.g., [**22**, Chap. 8.4.2]. Another version of this scheme can be used to compute subsets of eigenvalues of $\boldsymbol{A}$; see [**3**]. For a variant that computes specific eigenvalues, we refer to [**39**, p. 46].

The cost of bisection is $\mathcal{O}(nk)$ flops, where k is the number of desired eigenvalues. Hence, it can be much faster than the QR iteration if $k \ll n$. Once the desired eigenvalues are found, we can use the *inverse power method* (Section 3.2) to find the corresponding eigenvectors. The inverse power method costs in the best case (well-separated eigenvalues) $\mathcal{O}(nk)$ flops. In the worst case (many clustered eigenvalues), the cost is $\mathcal{O}(nk^2)$ flops and the accuracy of the computed eigenvectors is not guaranteed. Next, we review a method that is better suited for finding all (or most) eigenvalues and eigenvectors, especially when the eigenvalues may be clustered.

3.8.3. *Divide-and-conquer.* The idea of divide-and-conquer is to recursively divide the eigenvalue problem into smaller subproblems until we reach matrices of dimension one, for which the eigenvalue problem is trivial. The method was first introduced in 1981 [**9**] while its parallel version was developed in 1987 [**14**].

The starting point is to write the symmetric tridiagonal matrix $\boldsymbol{A}$ in (3.5) as a sum of a block diagonal matrix of two tridiagonal matrices $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$, plus a rank-1 correction:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{T}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}_2 \end{bmatrix} + b_m \boldsymbol{v}\boldsymbol{v}^T,$$

where $\boldsymbol{v} \in \mathbb{R}^n$ is a column vector whose mth and $(m+1)$st entry is equal to one $(1 \leq m \leq n-1)$ and all remaining entries are zero. Suppose we have the real Schur decompositions of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, i.e., $\boldsymbol{A}_i = \boldsymbol{Q}_i \boldsymbol{D}_i \boldsymbol{Q}_i^T$ for $i = 1, 2$ with $\boldsymbol{Q}_1 \in \mathbb{R}^{m \times m}$ and $\boldsymbol{Q}_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ orthogonal. Then,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2 \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_2 \end{bmatrix} + b_m \boldsymbol{u}\boldsymbol{u}^T \right) \begin{bmatrix} \boldsymbol{Q}_1^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2^T \end{bmatrix},$$

where

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{Q}_1^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2^T \end{bmatrix} \boldsymbol{v} = \begin{bmatrix} \text{last column of } \boldsymbol{Q}_1^T \\ \text{first column of } \boldsymbol{Q}_2^T \end{bmatrix}.$$

Hence, $\lambda(\boldsymbol{A}) = \lambda(\boldsymbol{D} + b_m \boldsymbol{u}\boldsymbol{u}^T)$ where $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_2 \end{bmatrix} = \operatorname{diag}(d_i)_{i=1,\ldots,n}$ is a diagonal matrix. Hence, the problem now reduces to finding the eigenvalues of a diagonal plus a rank-1 matrix. This can be further simplified to finding the eigenvalues of the identity matrix plus a rank-1 matrix, using simply the characteristic polynomial: In particular, under the assumption that $\boldsymbol{D} - \lambda \boldsymbol{I}$ is nonsingular, and using

$$\det(\boldsymbol{D} + b_m \boldsymbol{u}\boldsymbol{u}^T - \lambda \boldsymbol{I}) = \det(\boldsymbol{D} - \lambda \boldsymbol{I}) \det\left(\boldsymbol{I} + b_m (\boldsymbol{D} - \lambda \boldsymbol{I})^{-1} \boldsymbol{u}\boldsymbol{u}^T \right),$$

we obtain

$$\lambda \in \lambda(\boldsymbol{A}) \Leftrightarrow \det\left(\boldsymbol{I} + b_m (\boldsymbol{D} - \lambda \boldsymbol{I})^{-1} \boldsymbol{u}\boldsymbol{u}^T \right) = 0.$$

The matrix $\boldsymbol{I} + b_m (\boldsymbol{D} - \lambda \boldsymbol{I})^{-1} \boldsymbol{u}\boldsymbol{u}^T$ is of special structure, and its determinant can be computed using the following lemma:

LEMMA 3.5 (See [**11**, Lemma 5.1].). *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Then*

$$\det\left(\boldsymbol{I} + \boldsymbol{x}\boldsymbol{y}^T \right) = 1 + \boldsymbol{y}^T \boldsymbol{x}.$$

In our case, we get

$$\det\left(\boldsymbol{I} + b_m (\boldsymbol{D} - \lambda \boldsymbol{I})^{-1} \boldsymbol{u}\boldsymbol{u}^T \right) = 1 + b_m \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} \equiv f(\lambda),$$

and the eigenvalues of $\boldsymbol{A}$ are the roots of the *secular equation* $f(\lambda) = 0$. This can be solved, e.g., by Newton's method, which converges in practice in a bounded number of steps per eigenvalue. Note that solving the secular equation needs caution due to possible deflations ($d_i = d_{i+1}$ or $u_i = 0$) or small values of u_i. For more details on the function f and on solving the secular equation, we refer to [**11**, Chap. 5.3.3]. The cost for computing all eigenvalues is $\mathcal{O}(n^2 \log(n))$ flops by using the above strategy of recursively divide the eigenvalue problem into smaller subproblems. Hence, the pure eigenvalue computation (without eigenvectors) is more expensive than in the QR iteration. However, the eigenvectors can be computed more cheaply:

LEMMA 3.6 (See [**11**, Lemma 5.2].). *If* $\lambda \in \lambda(\boldsymbol{D} + b_m \boldsymbol{u}\boldsymbol{u}^T)$, *then* $(\boldsymbol{D} - \lambda \boldsymbol{I})^{-1}\boldsymbol{u}$ *is a corresponding eigenvector.*

The cost for computing all eigenvectors is $\mathcal{O}(n^2)$ flops. However, the eigenvector computation from Lemma 3.6 is not numerically stable. If λ is too close to a diagonal entry d_i, we obtain large roundoff errors since we divide by $d_i - \lambda$. If two eigenvalues λ_i and λ_j are very close, the orthogonality of the computed eigenvectors can get lost. For a numerical stable computation, we refer to [**24**], which requires in practice $\mathcal{O}(cn^3)$ flops where $c \ll 1$.

Here, we finish the discussion of eigenvalue problems for small to moderate-sized matrices. We now move to discuss problems where the matrix is large and sparse.

4. Large and sparse matrices

In this section, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is considered to be large and sparse. Sparse matrices are matrices with very few nonzero entries. Sparse often means that there are $\mathcal{O}(1)$ nonzero entries per row. We note that matrices that are not necessarily sparse but give rise to very fast matrix-vector products (for example, via the Fast Fourier Transform) often also allow for applying the methods discussed in this section.

The meaning of "large matrices" is relative. Let us say we consider matrices of size millions. In order to take advantage of the large number of zero entries, special storage schemes are required; see, e.g., [**47**, Chap. 2]. We will assume that it is not easy to form a matrix decomposition such as the QR factorization. In particular, similarity transformations would destroy the sparsity. Hence, we will mainly rely on matrix-vector products, which are often computable in $\mathcal{O}(n)$ flops instead of $\mathcal{O}(n^2)$.

In this chapter, we review methods for computing a few eigenpairs of $\boldsymbol{A}$. In fact, in practice, one often needs the k smallest/largest eigenvalues or the k eigenvalues closest to $\mu \in \mathbb{C}$ for a small k and their corresponding eigenvectors. In the following, we introduce *orthogonal projection methods*, from which we can derive the state-of-the-art *Krylov methods*, which make use of cheap matrix-vector products. Note that projection methods even play a role for the methods discussed in Section 3. For instance, when we use the power method but the dominant eigenvalue is complex, then we can obtain the eigenvalue approximation using projection techniques with information obtained from the power method.

4.1. Orthogonal projection methods.

Suppose we want to find an approximation $(\hat{\lambda}, \hat{\boldsymbol{x}})$ of an eigenpair $(\lambda, \boldsymbol{x})$ of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. The idea of *projection techniques* is to extract $\hat{\boldsymbol{x}}$ from some subspace $\mathcal{K}$. This is called the *subspace of approximants* or the *right subspace*. The uniqueness of $\hat{\boldsymbol{x}}$ is typically realized via the imposition of orthogonality conditions. We denote by

$$\boldsymbol{r} = \boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}$$

the residual vector. It is a measure for the quality of the approximate eigenpair $(\hat{\lambda}, \hat{\boldsymbol{x}})$. The orthogonality conditions consist of constraining the residual $\boldsymbol{r}$ to be orthogonal to some subspace $\mathcal{L}$, i.e.,

$$\boldsymbol{r} \perp \mathcal{L}.$$

$\mathcal{L}$ is called the *left subspace*. This framework is commonly known as the *Petrov-Galerkin conditions* in diverse areas of mathematics, e.g., the finite element method. The case $\mathcal{L} = \mathcal{K}$ leads to the *Galerkin conditions* and gives an *orthogonal projection*, which we discuss next. The case where $\mathcal{L}$ is different from $\mathcal{K}$ is called *oblique projection*, and we quickly have a look into this framework at the end of this section.

Let us assume that $\boldsymbol{A}$ is symmetric. Let $\mathcal{K}$ be a k-dimensional subspace of $\mathbb{R}^n$. An *orthogonal projection* technique onto $\mathcal{K}$ seeks an approximate eigenpair $(\hat{\lambda}, \hat{\boldsymbol{x}})$ such that $\hat{\boldsymbol{x}} \in \mathcal{K}$ and

$$\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}} \perp \mathcal{K},$$

or equivalently

$$\left(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}, \boldsymbol{v}\right) = 0 \quad \forall \boldsymbol{v} \in \mathcal{K}. \tag{4.1}$$

Let $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k\}$ be an orthonormal basis of $\mathcal{K}$ and $\boldsymbol{Q}_k = [\boldsymbol{q}_1|\ldots|\boldsymbol{q}_k] \in \mathbb{R}^{n\times k}$. Then, (4.1) becomes

$$\left(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}, \boldsymbol{q}_i\right) = 0 \quad \forall i = 1, \ldots, k.$$

If we express $\hat{\boldsymbol{x}}$ in terms of the basis of $\mathcal{K}$, i.e., $\hat{\boldsymbol{x}} = \boldsymbol{Q}_k\boldsymbol{y}$, we get

$$\left(\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \hat{\lambda}\boldsymbol{Q}_k\boldsymbol{y}, \boldsymbol{q}_i\right) = 0 \quad \forall i = 1, \ldots, k,$$

and due to $\boldsymbol{Q}_k^T\boldsymbol{Q}_k = \boldsymbol{I}$, we obtain

$$\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} = \hat{\lambda}\boldsymbol{y}.$$

This is the basis for *Krylov subspace methods*, which we discuss in the next section. The matrix $\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k \in \mathbb{R}^{k\times k}$ will often be smaller than $\boldsymbol{A}$ and is either upper Hessenberg (nonsymmetric case) or tridiagonal (symmetric case). The *Rayleigh–Ritz procedure* presented in Algorithm 4.1 computes such a Galerkin approximation. The θ_i are called *Ritz values* and $\hat{\boldsymbol{x}}_i$ are the *Ritz vectors*. We will see that the Ritz

Algorithm 4.1: Rayleigh–Ritz procedure

```
1 Compute an orthonormal basis $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k\}$ of the subspace $\mathcal{K}$. Set
    $\boldsymbol{Q}_k = [\boldsymbol{q}_1|\ldots|\boldsymbol{q}_k]$.
2 Compute $\boldsymbol{T}_k = \boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k$.
3 Compute $j$ eigenvalues of $\boldsymbol{T}_k$, say $\theta_1, \ldots, \theta_j$.
4 Compute the corresponding eigenvectors $\boldsymbol{v}_j$ of $\boldsymbol{T}_k$. Then, the corresponding
    approximate eigenvectors of $\boldsymbol{A}$ are $\hat{\boldsymbol{x}}_j = \boldsymbol{Q}_k\boldsymbol{v}_j$.
```

values and Ritz vectors are the best approximate eigenpairs in the least-squares sense. But first, let us put the presented framework into a similarity transformation of $\boldsymbol{A}$: Suppose $\boldsymbol{Q} = [\boldsymbol{Q}_k, \boldsymbol{Q}_u] \in \mathbb{R}^{n\times n}$ is an orthogonal matrix with $\boldsymbol{Q}_k \in \mathbb{R}^{n\times k}$ being the matrix above that spans the subspace $\mathcal{K}$. We introduce

$$\boldsymbol{T} := \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k & \boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_u \\ \boldsymbol{Q}_u^T\boldsymbol{A}\boldsymbol{Q}_k & \boldsymbol{Q}_u^T\boldsymbol{A}\boldsymbol{Q}_u \end{bmatrix} =: \begin{bmatrix} \boldsymbol{T}_k & \boldsymbol{T}_{uk} \\ \boldsymbol{T}_{ku} & \boldsymbol{T}_u \end{bmatrix}.$$

Let $\boldsymbol{T}_k = \boldsymbol{V}\boldsymbol{\Theta}\boldsymbol{V}^T$ be the eigendecomposition of $\boldsymbol{T}_k$. Note that for $k = 1$, $\boldsymbol{T}_1$ is just the Rayleigh quotient (see Definition 2.6).

Now, we can answer the question on the "best" approximation to an eigenvector in $\mathcal{K}$. Similar to the observation in Section 3.1 that the Rayleigh quotient is the best eigenvalue approximation in the least-squares sense, we have the following useful result.

THEOREM 4.1 (See [**11**, Theor. 7.1].). *The minimum of $\|\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_k\boldsymbol{R}\|_2$ over all $k \times k$ symmetric matrices $\boldsymbol{R}$ is attained by $\boldsymbol{R} = \boldsymbol{T}_k$, in which case $\|\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_k\boldsymbol{R}\|_2 = \|\boldsymbol{T}_{ku}\|_2$. Let $\boldsymbol{T}_k = \boldsymbol{V}\boldsymbol{\Theta}\boldsymbol{V}^T$ be the eigendecomposition of $\boldsymbol{T}_k$. The minimum of $\|\boldsymbol{A}\boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{D}\|_2$ over all $n \times k$ orthogonal matrices $\boldsymbol{P}_k$ ($\boldsymbol{P}_k^T\boldsymbol{P}_k = \boldsymbol{I}$) where* $\text{span}(\boldsymbol{P}_k) = \text{span}(\boldsymbol{Q}_k)$ *and over diagonal matrices $\boldsymbol{D}$ is also $\|\boldsymbol{T}_{ku}\|_2$ and is attained by $\boldsymbol{P}_k = \boldsymbol{Q}_k\boldsymbol{V}$ and $\boldsymbol{D} = \boldsymbol{\Lambda}$.*

In practice, the columns of $\boldsymbol{Q}_k$ will be computed by, e.g., the *Lanczos algorithm* or *Arnoldi algorithm*, which we discuss in Section 4.3 and 4.4.

Now, let us have a quick look at the *oblique projection* technique in which $\mathcal{L}$ is different from $\mathcal{K}$. Let $\mathcal{K}$ and $\mathcal{L}$ be k-dimensional subspaces of $\mathbb{R}^n$. An *oblique projection* technique onto $\mathcal{K}$ seeks an approximate eigenpair $(\hat{\lambda}, \hat{\boldsymbol{x}})$ such that $\hat{\boldsymbol{x}} \in \mathcal{K}$ and

$$\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}} \perp \mathcal{L},$$

or equivalently

$$\left(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}, \boldsymbol{v}\right) = 0 \quad \forall \boldsymbol{v} \in \mathcal{L}. \tag{4.2}$$

Let $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k\}$ be an orthonormal basis of $\mathcal{K}$, $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k\}$ an orthonormal basis of $\mathcal{L}$, $\boldsymbol{Q}_k = [\boldsymbol{q}_1|\ldots|\boldsymbol{q}_k] \in \mathbb{R}^{n\times k}$, and $\boldsymbol{P}_k = [\boldsymbol{p}_1|\ldots|\boldsymbol{p}_k] \in \mathbb{R}^{n\times k}$. Further, we assume biorthogonality, i.e., $\boldsymbol{P}_k^T\boldsymbol{Q}_k = \boldsymbol{I}$. Then, (4.2) becomes

$$\left(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}, \boldsymbol{p}_i\right) = 0 \quad \forall i = 1, \ldots, k.$$

If we express $\hat{\boldsymbol{x}}$ in terms of the basis of $\mathcal{K}$, i.e., $\hat{\boldsymbol{x}} = \boldsymbol{Q}_k\boldsymbol{y}$, we get

$$\left(\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \hat{\lambda}\boldsymbol{Q}_k\boldsymbol{y}, \boldsymbol{p}_i\right) = 0 \quad \forall i = 1, \ldots, k,$$

and due to $\boldsymbol{P}_k^T\boldsymbol{Q}_k = \boldsymbol{I}$, we obtain

$$\boldsymbol{P}_k^T\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} = \hat{\lambda}\boldsymbol{y}.$$

Oblique projection techniques form the basis for the *non-Hermitian Lanczos process* [**25, 26, 41**], which belongs to the class of *Krylov subspace solvers.* Krylov subspace solvers form the topic of the next section. For a further discussion on the oblique projection technique, we refer to, e.g., [**47**, Chap. 4.3.3].

4.2. Krylov subspace methods.

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Krylov subspace methods are used to solve linear systems or eigenvalue problems of sparse matrices. They only require that $\boldsymbol{A}$ be accessible via a "black-box" subroutine which describes the application of $\boldsymbol{A}$ to a vector. A k-dimensional *Krylov subspace* associated with a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{v}$ is the subspace given by

$$\mathcal{K}_k(\boldsymbol{A}; \boldsymbol{v}) = \text{span}\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{v}\}.$$

The corresponding *Krylov matrix* is denoted by

$$\boldsymbol{K}_k(\boldsymbol{A};\boldsymbol{v}) = [\boldsymbol{v}|\boldsymbol{A}\boldsymbol{v}|\boldsymbol{A}^2\boldsymbol{v}|\dots|\boldsymbol{A}^{k-1}\boldsymbol{v}].$$

Using a Krylov subspace as *right subspace* $\mathcal{K}$ in projection methods has proven to be efficient. Various Krylov subspace methods arose from different choices of the *left subspaces* $\mathcal{L}$. The Krylov subspace $\mathcal{K}_k(\boldsymbol{A};\boldsymbol{v})$ arises naturally if we refer to it as the subspace generated by $k-1$ steps of the power iteration (see Section 3.1) with initial guess $\boldsymbol{v}$. Similarly, for the inverse power iteration (see Section 3.2), we obtain the subspace

$$\mathcal{K}_k\left((\boldsymbol{A}-\alpha\boldsymbol{I})^{-1};\boldsymbol{v}\right).$$

Both iterations produce a sequence of vectors $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k$ that span a Krylov subspace and take $\boldsymbol{v}_k$ as the approximate eigenvector. Now, rather than taking $\boldsymbol{v}_k$, it is natural to use the whole sequence $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k$ in searching for the eigenvector. In fact, we saw in the previous section (Theorem 4.1 for the symmetric case) that we can even use $\mathcal{K}_k$ to compute the k best approximate eigenvalues and eigenvectors. There are three basic algorithms for generating a basis for the Krylov subspace: the *Lanczos process* for symmetric matrices, which we discuss next, the *Arnoldi process* for nonsymmetric matrices (Section 4.4), and the *nonsymmetric Lanczos process.* The latter computes matrices $\boldsymbol{Q}$ and $\boldsymbol{P}$ with $\boldsymbol{P}^T\boldsymbol{Q}=\boldsymbol{I}$ such that $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{Q}$ is tridiagonal; see, e.g., [**25, 26, 41**]. Moreover, there exist *block versions* of the Arnoldi and Lanczos process; see, e.g., [**8, 48**], which may exploit the block structure of a matrix in some situations. They are basically an acceleration technique of the *subspace iteration*, similar to the way the subspace iteration generalizes the power methods.

4.3. The Lanczos process.

Let $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ be symmetric. The *Lanczos process* computes an orthogonal basis for the Krylov subspace $\mathcal{K}_k(\boldsymbol{A};\boldsymbol{v})$ for some initial vector $\boldsymbol{v}$, and approximates the eigenvalues of $\boldsymbol{A}$ by the Ritz values.

Recall that the Hessenberg reduction of a symmetric matrix $\boldsymbol{A}$ reduces to a tridiagonal matrix, i.e., there exists an orthogonal $\boldsymbol{Q}\in\mathbb{R}^{n\times n}$ such that

$$\boldsymbol{T}=\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q}=\begin{bmatrix}\alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n\end{bmatrix}. \tag{4.3}$$

The connection between the tridiagonalization of $\boldsymbol{A}$ and the QR factorization of $\boldsymbol{K}_k(\boldsymbol{A};\boldsymbol{q}_1)$, where $\boldsymbol{q}_1=\boldsymbol{Q}\boldsymbol{e}_1$ is given as follows:

THEOREM 4.2 (See [**22**, Theor. 8.3.1].). *Let (4.3) be the tridiagonal decomposition of a symmetric matrix $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ with $\boldsymbol{q}_1=\boldsymbol{Q}\boldsymbol{e}_1$. Then:*

(1) $\boldsymbol{Q}^T\boldsymbol{K}_n(\boldsymbol{A};\boldsymbol{q}_1)=\boldsymbol{R}$ *is upper triangular.*
(2) *If $\boldsymbol{R}$ is nonsingular, then $\boldsymbol{T}$ is unreduced.*
(3) *If $k=\arg\min_{j=1,\dots,n}\{r_{j,j}=0\}$, then $k-1=\arg\min_{j=1,\dots,n-1}\{\beta_j=0\}$.*

It follows from (1) in Theorem 4.2 that $\boldsymbol{Q}\boldsymbol{R}$ is the QR factorization of $\boldsymbol{K}_n(\boldsymbol{A};\boldsymbol{q}_1)$. In order to preserve the sparsity, we need an alternative to similarity transformations in order to compute the tridiagonalization. Let us write $\boldsymbol{Q} = [\boldsymbol{q}_1|\dots|\boldsymbol{q}_n]$. Considering the kth column of $\boldsymbol{AQ} = \boldsymbol{QT}$, we obtain the following three-term recurrence:

$$\boldsymbol{A}\boldsymbol{q}_k = \beta_{k-1}\boldsymbol{q}_{k-1} + \alpha_k \boldsymbol{q}_k + \beta_k \boldsymbol{q}_{k+1}. \tag{4.4}$$

Since the columns of $\boldsymbol{Q}$ are orthonormal, multiplying (4.4) from the left by $\boldsymbol{q}_k$ yields

$$\alpha_k = \boldsymbol{q}_k^T \boldsymbol{A}\boldsymbol{q}_k.$$

This leads to the method in Algorithm 4.2 developed by Lanczos in 1950 [**34**].

Algorithm 4.2: Lanczos process

```
1  Given $q_0 = 0$, $q_1 = \frac{v}{\|v\|_2}$, $\beta_0 = 0$
2  for $k = 1, 2, \dots$ do
3      $z_k = Aq_k$
4      $\alpha_k = q_k^T z_k$
5      $z_k = z_k - \beta_{k-1}q_{k-1} - \alpha_k q_k$
6      $\beta_k = \|z_k\|_2$
7      if $\beta_k = 0$ then
8          quit
9      end
10     $q_{k+1} = \frac{z_k}{\beta_k}$
11 end
```

The vectors $\boldsymbol{q}_k$ computed by the Lanczos algorithm are called *Lanczos vectors.* The Lanczos process stops before the complete tridiagonalization if $\boldsymbol{q}_1$ is contained in an exact $\boldsymbol{A}$-invariant subspace:

THEOREM 4.3 (See [**22**, Theor. 10.1.1].). *The Lanczos Algorithm 4.2 runs until $k = m$, where*

$$m = \operatorname{rank}(\boldsymbol{K}_n(\boldsymbol{A}, \boldsymbol{q}_1)).$$

Moreover, for $k = 1, \dots, m$, we have

$$\boldsymbol{AQ}_k = \boldsymbol{Q}_k\boldsymbol{T}_k + \beta_k \boldsymbol{q}_{k+1}\boldsymbol{e}_k^T, \tag{4.5}$$

where $\boldsymbol{T}_k = \boldsymbol{T}(1:k, 1:k)$, $\boldsymbol{Q}_k = [\boldsymbol{q}_1|\dots|\boldsymbol{q}_k]$ has orthonormal columns with

$$\operatorname{span}\{\boldsymbol{q}_1, \dots, \boldsymbol{q}_k\} = \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{q}_1).$$

In particular, $\beta_m = 0$, and hence

$$\boldsymbol{AQ}_m = \boldsymbol{Q}_m\boldsymbol{T}_m.$$

The eigenvalues of the tridiagonal $\boldsymbol{T}_m$ can then be computed via, e.g., the QR iteration. A corresponding eigenvector can be obtained by using the inverse power iteration with the approximated eigenvalue as shift.

We can show that the quality of the approximation after k Lanczos steps depends on β_k and on parts of the eigenvectors of $\boldsymbol{T}_k$ (cf. [**22**, Chap. 10.1.4]): Therefore, let $(\theta, \boldsymbol{y})$ be an eigenpair of $\boldsymbol{T}_k$. Applying (4.5) to $\boldsymbol{y}$ yields

$$\begin{aligned}\boldsymbol{AQ}_k\boldsymbol{y} &= \boldsymbol{Q}_k\boldsymbol{T}_k\boldsymbol{y} + \beta_k\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T\boldsymbol{y}\\ &= \theta\boldsymbol{Q}_k\boldsymbol{y} + \beta_k\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T\boldsymbol{y}\end{aligned}$$

and hence the following error estimation (cf. Theorem 4.1)

$$\|\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \theta\boldsymbol{Q}_k\boldsymbol{y}\|_2 = |\beta_k|\,|\boldsymbol{e}_k^T\boldsymbol{y}|.$$

Hence, we want to accomplish $\beta_k = 0$ fast. Regarding the convergence theory, we refer to [**31, 38, 43**] and [**22**, Chap. 10.1.5]. In summary, the Ritz values converge fast to the extreme eigenvalues. Using shift and invert strategies (as in the inverse power method in Section 3.2), we can obtain convergence to interior eigenvalues. In practice, rounding errors have a significant effect on the behavior of the Lanczos iteration. If the computed β_k are close to zero, then the Lanczos vectors lose their orthogonality. Reorthogonalization strategies provide a remedy; see, e.g., [**38, 21, 40, 50, 6, 67**]. Nevertheless, we know from the last section that the Ritz values and vectors are good approximations.

So far, we have assumed that $\boldsymbol{A}$ is symmetric. Next, we consider the nonsymmetric case.

4.4. The Arnoldi process.

For a nonsymmetric $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we know that there exists a Hessenberg decomposition, i.e., there exists an orthogonal $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ such that

$$\boldsymbol{H} = \boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & h_{n-1,n} \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix}. \tag{4.6}$$

The connection between the Hessenberg reduction of $\boldsymbol{A}$ and the QR factorization of $\boldsymbol{K}_k(\boldsymbol{A};\boldsymbol{q}_1)$, where $\boldsymbol{q}_1 = \boldsymbol{Q}\boldsymbol{e}_1$ is given as follows (cf. the symmetric case in Theorem 4.2)

THEOREM 4.4 (See [**22**, Theor. 7.4.3].). *Suppose $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is orthogonal and let $\boldsymbol{q}_1 = \boldsymbol{Q}\boldsymbol{e}_1$ and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Then, $\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{H}$ is an unreduced upper Hessenberg matrix if and only if $\boldsymbol{Q}^T\boldsymbol{K}_n(\boldsymbol{A};\boldsymbol{q}_1) = \boldsymbol{R}$ is nonsingular and upper triangular.*

It follows from Theorem 4.4 that $\boldsymbol{Q}\boldsymbol{R}$ is the QR factorization of $\boldsymbol{K}_n(\boldsymbol{A};\boldsymbol{q}_1)$. As before, in order to preserve the sparsity, we need an alternative to similarity transformations in order to compute the Hessenberg reduction. Let us write $\boldsymbol{Q} = [\boldsymbol{q}_1|\ldots|\boldsymbol{q}_n]$. Considering the kth column of $\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{H}$, we obtain the following recurrence:

$$\boldsymbol{A}\boldsymbol{q}_k = \sum_{i=1}^{k+1} h_{i,k}\boldsymbol{q}_i. \tag{4.7}$$

Since the columns of $\boldsymbol{Q}$ are orthonormal, multiplying (4.7) from the left by $\boldsymbol{q}_i$ yields

$$h_{i,k} = \boldsymbol{q}_i^T\boldsymbol{A}\boldsymbol{q}_k$$

for $i = 1,\ldots,k$. This leads to the method in Algorithm 4.3 developed by Arnoldi in 1951 [**1**]. It can be viewed as an extension of the Lanczos process to nonsymmetric matrices. Note that, in contrast to the symmetric case, we have no three-term recurrence anymore. Hence, we have to store all computed vectors $\boldsymbol{q}_k$. These vectors are called *Arnoldi vectors.* The Arnoldi process can be seen as a modified Gram-Schmidt orthogonalization process (cf. [**22**, Chap. 5.2.8]) since in each step

Algorithm 4.3: Arnoldi process

```
1  Given $\boldsymbol{q}_1 = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$
2  for $k = 1, 2, \dots$ do
3      $\boldsymbol{z}_k = \boldsymbol{A}\boldsymbol{q}_k$
4      for $i = 1, \dots, k$ do
5          $h_{i,k} = \boldsymbol{q}_i^T \boldsymbol{z}_k$
6          $\boldsymbol{z}_k = \boldsymbol{z}_k - h_{i,k}\boldsymbol{q}_i$
7      end
8      $h_{k+1,k} = \|\boldsymbol{z}_k\|_2$
9      if $h_{k+1,k} = 0$ then
10         quit
11     end
12     $\boldsymbol{q}_{k+1} = \frac{\boldsymbol{z}_k}{h_{k+1,k}}$
13 end
```

k we orthogonalize $\boldsymbol{A}\boldsymbol{q}_k$ against all previous $\boldsymbol{q}_i$ — ths requires $\mathcal{O}(kn)$ flops. Hence, the computational cost grows rapidly with the number of steps. After k steps of the Arnoldi Algorithm 4.3, we have

$$\boldsymbol{A}\boldsymbol{Q}_k = \boldsymbol{Q}_k\boldsymbol{H}_k + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T, \tag{4.8}$$

where $\boldsymbol{H}_k = \boldsymbol{H}(1:k, 1:k)$ and

$$\operatorname{span}\{\boldsymbol{q}_1, \dots, \boldsymbol{q}_k\} = \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{q}_1). \tag{4.9}$$

We can show that the quality of the approximation depends on the magnitude of $h_{k+1,k}$ and on parts of the eigenvectors of $\boldsymbol{H}_k$ (cf. [**22**, Chap. 10.5.1]): Therefore, let $(\theta, \boldsymbol{y})$ be an eigenpair of $\boldsymbol{H}_k$. Applying (4.8) to $\boldsymbol{y}$ yields

$$\begin{aligned}\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} &= \boldsymbol{Q}_k\boldsymbol{H}_k\boldsymbol{y} + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T\boldsymbol{y} \\ &= \theta\boldsymbol{Q}_k\boldsymbol{y} + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T\boldsymbol{y}\end{aligned}$$

and hence the following error estimation (cf. Theorem 4.1)

$$\|\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \theta\boldsymbol{Q}_k\boldsymbol{y}\|_2 = |h_{k+1,k}|\,|\boldsymbol{e}_k^T\boldsymbol{y}|.$$

Hence, we want $h_{k+1,k} = 0$ fast. As the Lanczos process, the Arnoldi process has a (lucky) breakdown at step $k = m$ if $h_{m+1,m} = 0$ since $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{q}_1)$ is an $\boldsymbol{A}$-invariant subspace in this case. In the following, we discuss accelerating techniques for the Arnoldi and Lanczos process.

4.5. Restarted Arnoldi and Lanczos.

Note that with each Arnoldi step we have to store one additional Arnoldi vector. A remedy is restarting the Arnoldi process with carefully chosen restarts after a certain maximum of steps is reached. Acceleration techniques (mainly of a polynomial nature) generate an initial guess with small components in the unwanted parts of the spectrum. The strategies we present are called *polynomial acceleration* or *filtering techniques.* They exploit the powers of a matrix similar as the power method in the sense that they generate iterations of the form

$$\boldsymbol{z}_r = p_r(\boldsymbol{A})\boldsymbol{z}_0,$$

where p_r is a polynomial of degree r. In the case of the power method, we have $p_r(t) = t^r$. Filtering methods have been successfully combined with subspace iteration. When combined with the Arnoldi process, they are often called *implicitly restarted methods*, which we discuss next. Selecting a good polynomial often relies on some knowledge of the eigenvalues or related quantities (e.g., Ritz values).

Suppose $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is diagonalizable and has eigenpairs $\{(\lambda_i, \boldsymbol{x}_i)\}_{i=1,\ldots,n}$ with $\lambda_1 \geq \ldots \geq \lambda_n$. Let

$$\boldsymbol{q}_1 = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$$

be an initial guess for the Arnoldi process. After running r steps of the Arnoldi process, we do a restart. We may seek a new initial vector from the span of the Arnoldi vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r$, which has, due to (4.9), the form

$$\begin{aligned}\boldsymbol{q}_+ &= \sum_{j=1}^{r} \beta_j \boldsymbol{A}^{j-1}\boldsymbol{q}_1 = \sum_{j=1}^{r} \beta_j \sum_{i=1}^{n} \alpha_i \boldsymbol{A}^{j-1}\boldsymbol{x}_i = \sum_{j=1}^{r} \beta_j \sum_{i=1}^{n} \alpha_i \lambda_i^{j-1}\boldsymbol{x}_i \\ &= \sum_{i=1}^{n} \alpha_i p_{r-1}(\lambda_i)\boldsymbol{x}_i.\end{aligned}$$

Suppose we are interested in the eigenvalue λ_j. If $|\alpha_j p_{r-1}(\lambda_j)| \gg |\alpha_l p_{r-1}(\lambda_l)|$ for all $l \neq j$, then $\boldsymbol{q}_+$ has large components in the eigendirection $\boldsymbol{x}_j$. Note that the α_i are unknown. Hence, with an appropriate constructed polynomial, we can amplify the components in the desired parts of the spectrum. For instance, we are seeking for a polynomial that satisfies $p_{r-1}(\lambda_j) = 1$ and $|p_{r-1}(\lambda_l)| \ll 1$ for all $l \neq j$. However, the eigenvalues λ_i are unknown as well. Hence we need some approximation. Let Ω be a domain (e.g., an ellipse) that contains $\lambda(\boldsymbol{A}) \setminus \{\lambda_j\}$, and suppose we have an estimate of λ_j. Then, we can aim to solve

$$\min_{\substack{p_{r-1}\in P_{r-1},\\ p_{r-1}(\lambda_j)=1}} \max_{t\in\Omega} |p_{r-1}(t)|.$$

Suitable polynomials include the shifted and scaled Chebyshev polynomials, and in the symmetric case, we can exploit the three-term recurrence for fast computation; see, e.g., [**45**].

An alternative to Chebyshev polynomials is the following: Given $\{\theta_i\}_{i=1,\ldots,r-1}$, then one natural idea is to set

$$p_{r-1}(t) = (t-\theta_1)(t-\theta_2)\cdots(t-\theta_{r-1}); \tag{4.10}$$

see [**22**, Chap. 10.5.2]. If $\lambda_i \approx \theta_l$ for some l, then $\boldsymbol{q}_+$ has small components in the eigendirection $\boldsymbol{x}_i$. Hence, the θ_i are all unwanted values. For θ_i we can use the Ritz values, which presumably approximate the eigenvalues of $\boldsymbol{A}$. For further heuristics, we refer to [**44**].

The above strategies are *explicit restarting* techniques, which use only one vector for the restart. The following *implicit restarting* strategy uses k vectors from the previous Arnoldi process for the new restarted Arnoldi process and throws away the remaining $r-k =: p$ vectors. The procedure was developed in 1992 [**53**]. It implicitly determines a polynomial of the form (4.10) using the QR iteration with shifts. Suppose we have performed r steps of the Arnoldi iteration with starting vector $\boldsymbol{q}_1$. Due to (4.8), we have

$$\boldsymbol{A}\boldsymbol{Q}_r = \boldsymbol{Q}_r\boldsymbol{H}_r + h_{r+1,r}\boldsymbol{q}_{r+1}\boldsymbol{e}_r^T, \tag{4.11}$$

where $\boldsymbol{H}_r \in \mathbb{R}^{r\times r}$ is upper Hessenberg, $\boldsymbol{Q}_r \in \mathbb{R}^{n\times r}$ has orthonormal columns, and $\boldsymbol{Q}_r\boldsymbol{e}_1 = \boldsymbol{q}_1$. Next, we apply p steps of the QR iteration with shifts $\theta_1, \ldots, \theta_p$ (Algorithm 3.6), i.e., in step i we compute

$$\boldsymbol{V}_i\boldsymbol{R}_i = \boldsymbol{H}^{(i-1)} - \theta_i\boldsymbol{I}, \tag{4.12}$$

$$\boldsymbol{H}^{(i)} = \boldsymbol{R}_i\boldsymbol{V}_i + \theta_i\boldsymbol{I}, \tag{4.13}$$

where $\boldsymbol{H}^{(0)} = \boldsymbol{H}_r$. After p steps, we have

$$\boldsymbol{H}^{(p)} = \boldsymbol{R}_p\boldsymbol{V}_p + \theta_p\boldsymbol{I} = \boldsymbol{V}_p^T\left(\boldsymbol{V}_p\boldsymbol{R}_p + \theta_p\boldsymbol{I}\right)\boldsymbol{V}_p = \boldsymbol{V}_p^T\boldsymbol{H}^{(p-1)}\boldsymbol{V}_p = \ldots = \boldsymbol{V}^T\boldsymbol{H}^{(0)}\boldsymbol{V}$$

with $\boldsymbol{V} = \boldsymbol{V}_1\cdots\boldsymbol{V}_p$. We use the notation

$$\boldsymbol{H}_+ := \boldsymbol{H}^{(p)} = \boldsymbol{V}^T\boldsymbol{H}^{(0)}\boldsymbol{V} = \boldsymbol{V}^T\boldsymbol{H}_r\boldsymbol{V}. \tag{4.14}$$

The relationship to a polynomial of the form (4.10) is the following:

Theorem 4.5 (See [**22**, Theor. 10.5.1].). *If $\boldsymbol{V} = \boldsymbol{V}_1\cdots\boldsymbol{V}_p$ and $\boldsymbol{R} = \boldsymbol{R}_p\cdots\boldsymbol{R}_1$ are defined by (4.12)–(4.13), then*

$$\boldsymbol{V}\boldsymbol{R} = (\boldsymbol{H}_r - \theta_1\boldsymbol{I})\cdots(\boldsymbol{H}_r - \theta_p\boldsymbol{I}).$$

Using (4.14), we get in (4.11)

$$\boldsymbol{A}\boldsymbol{Q}_r = \boldsymbol{Q}_r\boldsymbol{V}\boldsymbol{H}_+\boldsymbol{V}^T + h_{r+1,r}\boldsymbol{q}_{r+1}\boldsymbol{e}_r^T. \tag{4.15}$$

Multiplying (4.15) from the right by $\boldsymbol{V}$ yields

$$\boldsymbol{A}\boldsymbol{Q}_+ = \boldsymbol{Q}_+\boldsymbol{H}_+ + h_{r+1,r}\boldsymbol{q}_{r+1}\boldsymbol{e}_r^T\boldsymbol{V} \tag{4.16}$$

with $\boldsymbol{Q}_+ = \boldsymbol{Q}_r\boldsymbol{V}$. It can be shown that $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_p$ from the shifted QR iteration are upper Hessenberg. Hence, $\boldsymbol{V}(r, 1 : r-p-1) = \boldsymbol{0}^T$ and therefore $\boldsymbol{e}_r^T\boldsymbol{V} = [0\cdots0\,\alpha\ *\cdots*]$ is a row vector of length r whose first $r-p-1$ entries are zero. Now, using the notation $\boldsymbol{Q}_+ = [\hat{\boldsymbol{Q}}_{r-p}, \hat{\boldsymbol{Q}}_p]$ with $\hat{\boldsymbol{Q}}_{r-p} \in \mathbb{R}^{n\times(r-p)}$ we can write (4.16) as

$$\boldsymbol{A}[\hat{\boldsymbol{Q}}_{r-p}, \hat{\boldsymbol{Q}}_p] = [\hat{\boldsymbol{Q}}_{r-p}, \hat{\boldsymbol{Q}}_p]\begin{bmatrix} \hat{\boldsymbol{H}}_{r-p} & * \\ \beta\boldsymbol{e}_1\boldsymbol{e}_{r-p}^T & * \end{bmatrix} + h_{r+1,r}\boldsymbol{q}_{r+1}[\underbrace{0\cdots0}_{r-p-1}\,\alpha * \cdots *]. \tag{4.17}$$

Now, we throw away the last p columns in (4.17) and obtain an $(r-p)$-step Arnoldi decomposition

$$\begin{aligned}\boldsymbol{A}\hat{\boldsymbol{Q}}_{r-p} &= \hat{\boldsymbol{Q}}_{r-p}\hat{\boldsymbol{H}}_{r-p} + \beta\hat{\boldsymbol{Q}}_p\boldsymbol{e}_1\boldsymbol{e}_{r-p}^T + h_{r+1,r}\boldsymbol{q}_{r+1}[\underbrace{0\cdots0}_{r-p-1}\,\alpha] \\ &= \hat{\boldsymbol{Q}}_{r-p}\hat{\boldsymbol{H}}_{r-p} + \left(\beta\hat{\boldsymbol{Q}}_p\boldsymbol{e}_1 + \alpha h_{r+1,r}\boldsymbol{q}_{r+1}\right)\boldsymbol{e}_{r-p}^T \\ &=: \hat{\boldsymbol{Q}}_{r-p}\hat{\boldsymbol{H}}_{r-p} + \hat{\boldsymbol{v}}_{r+1}\boldsymbol{e}_{r-p}^T.\end{aligned}$$

This is the Arnoldi recursion we would have obtained by restarting the Arnoldi process with the starting vector $\boldsymbol{q}_+ = \boldsymbol{Q}_+\boldsymbol{e}_1$. Hence, we do not need to restart the Arnoldi process from step one but rather from step $r-p+1$. For further details, we refer to [**22**, Chap. 10.5.3] and the references therein.

REMARK 4.6. It can be shown (cf. [**22**, Chap. 10.5.3]) that

$$\boldsymbol{q}_+ = c\,(\boldsymbol{A} - \theta_1 \boldsymbol{I}) \cdots (\boldsymbol{A} - \theta_p \boldsymbol{I})\, \boldsymbol{Q}_r \boldsymbol{e}_1$$

for some scalar c and is hence of the form (4.10).

For further reading on the Arnoldi process we refer to, e.g., [**54, 47, 62, 55**].

Next we present another acceleration technique which is very popular for solving linear systems.

4.6. Preconditioning. In the following, we quickly review the preconditioning concept for solving large and sparse systems of linear equations of the general form

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}. \tag{4.18}$$

Here, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the given coefficient matrix, $\boldsymbol{z} \in \mathbb{R}^n$ is the unknown solution vector, and $\boldsymbol{b} \in \mathbb{R}^n$ is the given right-hand side vector. In order for the Equation (4.18) to have a unique solution, we assume that $\boldsymbol{A}$ is nonsingular. Systems of the form (4.18) arise after the discretization of a continuous problem like partial differential equations such as the time-harmonic Maxwell equations. Other applications arise in incompressible magnetohydrodynamics as well as constrained optimization. As already mentioned in Section 4.2, Krylov subspace solvers are also used for solving linear systems. In fact, they are state-of-the-art iterative solvers. However, they are usually only efficient in combination with an accelerator, which is called a *preconditioner*. The aim of a preconditioner is to enhance the convergence of the iterative solver. In our case, we want to accelerate the speed of convergence of Krylov subspace solvers. The basic idea is to construct a nonsingular matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ and solve

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{z} = \boldsymbol{P}^{-1}\boldsymbol{b} \tag{4.19}$$

instead of $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$. In order for $\boldsymbol{P}$ to be efficient, it should approximate $\boldsymbol{A}$, and at the same time, the action of $\boldsymbol{P}^{-1}$ should require little work. The construction process of $\boldsymbol{P}$ should incorporate the goal of eigenvalue clustering. That means, $\boldsymbol{P}^{-1}\boldsymbol{A}$ is aimed to have a few number of eigenvalues or eigenvalue clusters. This is bases on the following: In a nutshell, for linear systems the residual of a Krylov subspace solver $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}_k$ satisfies $\boldsymbol{r}_k = p_k(\boldsymbol{A})\boldsymbol{r}_0$, and one approach would be to minimize the norm of the residual, which amounts to requiring that $\|p_k(\lambda_i)\boldsymbol{v}_i\|_2$ be as small as possible for all $i = 1, \dots, n$. Here, $\{(\lambda_i, \boldsymbol{v}_i)\}_{i=1,\dots,n}$ are the eigenpairs of $\boldsymbol{A}$. Therefore, replacing $\boldsymbol{A}$ by $\boldsymbol{P}^{-1}\boldsymbol{A}$ such that $\boldsymbol{P}^{-1}\boldsymbol{A}$ has more clustered eigenvalues is one way to go. This typically results in outstanding performances of Krylov subspace solvers. For an overview of iterative solvers and preconditioning techniques, we refer to [**20, 23, 46, 16, 5, 2, 4, 59**].

Preconditioning plays an important role in eigenvalue problems as well. Taken in the same spirit as seeking an operator that improves the spectrum, we can think of the inverse power iteration (see Section 3.2) as a preconditioning approach: The operator $(\boldsymbol{A} - \theta\boldsymbol{I})^{-1}$ has a much better spectrum than $\boldsymbol{A}$ for a suitable chosen shift θ. So, we can run Arnoldi on $(\boldsymbol{A} - \theta\boldsymbol{I})^{-1}$ rather than $\boldsymbol{A}$ since the eigenvectors of $\boldsymbol{A}$ and $(\boldsymbol{A} - \theta\boldsymbol{I})^{-1}$ are identical. Another idea is to incorporate polynomial preconditioning, i.e., replace $\boldsymbol{A}$ by $p_k(\boldsymbol{A})$. As a guideline, want to transform the k

wanted eigenvalues of $\boldsymbol{A}$ to k eigenvalues of $p_k(\boldsymbol{A})$ that are much larger than the other eigenvalues, so as to accelerate convergence. Preconditioning also plays a role in solving *generalized eigenvalue problems*

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}.$$

They can be solved, e.g., by the *Jacobi–Davidson method*, whose idea we briefly discuss in Section 4.8 for solving the standard algebraic eigenvalue problem $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$. The discussion of generalized eigenproblems is out of the scope of this survey. We refer the reader to [**56, 57, 62**] for a background to these problems.

4.7. Davidson method. Davidson's method is basically a preconditioned version of the Lanczos process, but the amount of work increases similarly to Arnoldi, due to increased orthogonalization requirements. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\mathcal{K}_k = \mathcal{K}_k(\boldsymbol{A}; \boldsymbol{v})$ be a Krylov subspace with respect to some vector $\boldsymbol{v}$. Let $\{\boldsymbol{q}_1, \dots, \boldsymbol{q}_k\}$ be an orthonormal basis of $\mathcal{K}_k$. In the orthogonal projection technique, we are seeking for an $\hat{x} \in \mathcal{K}_k$ such that

$$\left(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}, \boldsymbol{q}_i\right) = 0 \quad \forall i = 1, \dots, k,$$

Suppose, we have a Ritz pair $(\theta_i, \boldsymbol{u}_i)$. Then, the residual is given by

$$\boldsymbol{r}_i = \boldsymbol{A}\boldsymbol{u}_i - \theta_i \boldsymbol{u}_i = (\boldsymbol{A} - \theta_i \boldsymbol{I})\boldsymbol{u}_i.$$

Now, we can improve the eigenpair approximation by precondition the residual, i.e., by solving

$$(\boldsymbol{P} - \theta_i \boldsymbol{I})\, \boldsymbol{t} = \boldsymbol{r}_i,$$

and define $\boldsymbol{t}$ as a new search direction, enriching the subspace. That is, $\boldsymbol{t}$ is orthogonalized against all basis vectors $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$, and the resulting vector $\boldsymbol{q}_{k+1}$ enriches $\mathcal{K}_k$ to $\mathcal{K}_{k+1}$.

Davidson [**10**] originally proposed to precondition with the diagonal matrix of $\boldsymbol{A}$, i.e., $\boldsymbol{P} = \mathrm{diag}(\boldsymbol{A})$, since he dealt with a diagonal dominant matrix $\boldsymbol{A}$. Additionally, diagonal preconditioning offers a computationally cheap iteration. For the use of other preconditioners, we refer to [**7**]. Further references on Davidson's method include [**54, 55, 36**].

Next, we consider an extension of Davdison's method, which has the potential of working better for matrices that are not diagonally dominant.

4.8. Jacobi–Davidson method. The idea is to extend the strategy of preconditioning the residual. If $(\hat{\lambda}, \hat{\boldsymbol{x}})$ with $\|\hat{\boldsymbol{x}}\|_2 = 1$ is an approximate eigenpair of $\boldsymbol{A}$, then the residual is $\boldsymbol{r} = \boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\lambda}\hat{\boldsymbol{x}}$. Now, we look for $(\hat{\lambda} + \delta\hat{\lambda}, \hat{\boldsymbol{x}} + \delta\hat{\boldsymbol{x}})$ to improve the eigenpair. We write

$$\boldsymbol{A}(\hat{\boldsymbol{x}} + \delta\hat{\boldsymbol{x}}) = (\hat{\lambda} + \delta\hat{\lambda})(\hat{\boldsymbol{x}} + \delta\hat{\boldsymbol{x}}),$$

which is equivalent to

$$(\boldsymbol{A} - \hat{\lambda}\boldsymbol{I})\delta\hat{\boldsymbol{x}} - \delta\hat{\lambda}\hat{\boldsymbol{x}} = -\boldsymbol{r} + \delta\hat{\lambda}\delta\hat{\boldsymbol{x}}.$$

By neglecting the second-order term, we obtain

$$(\boldsymbol{A} - \hat{\lambda}\boldsymbol{I})\delta\hat{\boldsymbol{x}} - \delta\hat{\lambda}\hat{\boldsymbol{x}} = -\boldsymbol{r}.$$

This is an underdetermined system and a constraint must be added, e.g., $\|\hat{\boldsymbol{x}} + \delta\hat{\boldsymbol{x}}\|_2 = 1$. With $\|\hat{\boldsymbol{x}}\|_2 = 1$ and neglecting the second-order term, this condition becomes

$$\hat{\boldsymbol{x}}^T \delta\hat{\boldsymbol{x}} = 0.$$

If $\hat{\lambda} = \hat{\boldsymbol{x}}^T \boldsymbol{A}\hat{\boldsymbol{x}}$, then we obtain $\delta\hat{\boldsymbol{x}}$ by solving the projected system

$$\begin{aligned}\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)(\boldsymbol{A} - \hat{\boldsymbol{\lambda}}\boldsymbol{I})\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)\delta\hat{\boldsymbol{x}} &= -\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)(\boldsymbol{r} - \delta\hat{\boldsymbol{\lambda}}\hat{\boldsymbol{x}}) \\ &= -\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)\boldsymbol{r} \\ &= -\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\boldsymbol{\lambda}}\hat{\boldsymbol{x}}) \\ &= -\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)\boldsymbol{A}\hat{\boldsymbol{x}} \\ &= -(\boldsymbol{A}\hat{\boldsymbol{x}} - \hat{\boldsymbol{\lambda}}\hat{\boldsymbol{x}}) = -\boldsymbol{r}\end{aligned}$$

subject to the constraint $\hat{\boldsymbol{x}}^T \delta\hat{\boldsymbol{x}} = 0$. As in the previous section, we replace $\boldsymbol{A}$ by a preconditioner $\boldsymbol{P}$, such that we have to solve an approximate projected system

$$\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)(\boldsymbol{P} - \hat{\boldsymbol{\lambda}}\boldsymbol{I})\left(\boldsymbol{I} - \hat{\boldsymbol{x}}\hat{\boldsymbol{x}}^T\right)\delta\hat{\boldsymbol{x}} = -\boldsymbol{r}$$

subject to the constraint $\hat{\boldsymbol{x}}^T \delta\hat{\boldsymbol{x}} = 0$.

The connection of the described method to Jacobi is given in Remark 4.7.

Remark 4.7. Given an approximate eigenpair $(\hat{\lambda}, \hat{\boldsymbol{x}})$ of $\boldsymbol{A}$, Jacobi [**30**] proposed to solve an eigenvalue problem $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ by finding a correction $\boldsymbol{t}$ such that

$$\boldsymbol{A}(\hat{\boldsymbol{x}} + \boldsymbol{t}) = \lambda(\hat{\boldsymbol{x}} + \boldsymbol{t}), \quad \hat{\boldsymbol{x}} \perp \boldsymbol{t}.$$

This is called the *Jacobi Orthogonal Component Correction (JOCC).*

The Jacobi–Davidson framework can also be connected with Newton's method; see, e.g., [**47**, Chap. 8.4].

The debate over the advantaged and disadvantages of Jacobi–Davidson versus other approaches such as the Arnoldi process (with shift and invert) is delicate. Sleijpen and van der Vorst [**51**] relate it to whether the new direction has a strong component in previous directions. It is a fairly technical argument, and not much theory is available. For more details about the Jacobi–Davidson method, we refer to [**51, 52, 54**].

5. Conclusions

The numerical solution of eigenvalue problems is an extremely active area of research. Eigenvalues are very important in many areas of applications, and challenges keep arising. The survey covers only some basic principles, which have established themselves as the fundamental building blocks of eigenvalue solvers. We have left out some important recent advances, which are extremely important but also rather technical. Generalized eigenvalue problems are also very important, but there is not enough room to cover them in this survey.

One of the main messages of this survey is the distinction between important *mathematical* observations about eigenvalues, and practical *computational* considerations. Objects such as the Jordan Canonical Form or determinants are classical mathematical tools, but they cannot be easily utilized in practical computations. On the other hand, sparsity of the matrix and the availability of matrix decompositions are often overlooked when a pure mathematical discussion of the problem ensues, but they are absolutely essential in the design of numerical methods.

Altogether, this topic is satisfyingly rich and challenging. It continues to be one of the most important problems in mathematical sciences.

References

1. W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math. **9** (1951), 17–29.
2. O. Axelsson, *A survey of preconditioned iterative methods for linear systems of algebraic equations*, BIT **25** (1985), no. 1, 165–187.
3. W. Barth, R. S. Martin, and J. H. Wilkinson, *Handbook Series Linear Algebra: Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection*, Numer. Math. **9** (1967), no. 5, 386–393.
4. M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys. **182** (2002), no. 2, 418–477.
5. M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer. **14** (2005), 1–137.
6. D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal. **2** (1994), no. 1, 1–21.
7. M. Crouzeix, B. Philippe, and M. Sadkane, *The Davidson method*, SIAM J. Sci. Comput. **15** (1994), no. 1, 62–76.
8. J. Cullum and W. E. Donath, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices*, 1974 IEEE Conference on Decision and Control, 1974, pp. 505–509.
9. J. J. M. Cuppen, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math. **36** (1981), no. 2, 177–195.
10. E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys. **17** (1975), no. 1, 87–94.
11. J. Demmel, *Applied numerical linear algebra*, SIAM, 1997.
12. J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl. **13** (1992), no. 4, 1204–1245.
13. I. S. Dhillon and B. N. Parlett, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra Appl. **387** (2004), 1–28.
14. J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comp. **8** (1987), no. 2, s139–s154.
15. A. A. Dubrulle and G. H. Golub, *A multishift QR iteration without computation of the shifts*, Numer. Algorithms **7** (1994), no. 2, 173–181.
16. H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: With applications in incompressible fluid dynamics*, Numer. Math. Sci. Comput., Oxford Univ. Press, Oxford, 2005.
17. J. Erxiong, *A note on the double-shift QL algorithm*, Linear Algebra Appl. **171** (1992), 121–132.
18. J. G. F. Francis, *The QR transformation: A unitary analogue to the LR transformation—Part 1*, Comput. J. **4** (1961), no. 3, 265–271.
19. ———, *The QR transformation—Part 2*, The Comput. J. **4** (1962), no. 4, 332–345.
20. R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numer. **1** (1992), 57–100.
21. G. H. Golub, R. R. Underwood, and J. H. Wilkinson, *The Lanczos algorithm for the symmetric $Ax = \lambda Bx$ problem*, Tech. report, Dep. Comput. Sci., Stanford Univ., Stanford, CA, 1972.
22. G. H. Golub and C. F. van Loan, *Matrix computations*, 4th ed., Johns Hopkins Stud. Math. Sci., Johns Hopkins Univ. Press, Baltimore, MD, 2013.
23. A. Greenbaum, *Iterative methods for solving linear systems*, Frontiers Appl. Math., vol. 17, SIAM, Philadelphia, PA, 1997.

24. M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl. **16** (1995), no. 1, 172–191.
25. M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms. I*, SIAM J. Matrix Anal. Appl. **13** (1992), no. 2, 594–639.
26. ———, *A completed theory of the unsymmetric Lanczos process and related algorithms. II*, SIAM J. Matrix Anal. Appl. **15** (1994), no. 1, 15–58.
27. N. Higham, *Accuracy and stability of numerical algorithms*, second ed., SIAM, 2002.
28. L. Hogben, *Elementary linear algebra*, West, St. Paul, MN, 1987.
29. H. Hotelling, *Analysis of a complex of statistical variables into principal components*, J. Educ. Psychol. **24** (1933), 417–441.
30. C. G. J. Jacobi, *Über ein leichtes Verfahren die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen*, J. Reine Angew. Math. **30** (1846), 51–94.
31. S. Kaniel, *Estimates for some computational techniques in linear algebra*, Math. Comp. **20** (1966), 369–378.
32. D. Kressner, *Numerical methods for general and structured eigenvalue problems*, Lect. Notes Comput. Sci. Eng., vol. 46, Springer, Berlin, 2005.
33. V. N. Kublanovskaja, *On some algorithms for the solution of the complete eigenvalue problem*, Ž. Vyčisl. Mat. i Mat. Fiz. **1** (1961), 555–570.
34. C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand. **45** (1950), 255–282.
35. R. S. Martin, G. Peters, and J. H. Wilkinson, *Handbook Series Linear Algebra: The QR algorithm for real Hessenberg matrices*, Numer. Math. **14** (1970), no. 3, 219–231.
36. R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comp. **7** (1986), no. 3, 817–825.
37. M. Overton, *Numerical computing with IEEE floating point arithmetic*, SIAM, 2001.
38. C. C. Paige, *The computation of eigenvalues and eigenvectors of very large sparse matrices*, PhD thesis, Univ. London, 1971.
39. B. Parlett, *The symmetric eigenvalue problem*, SIAM, 1998.
40. B. N. Parlett and D. S. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp. **33** (1979), no. 145, 217–238.
41. B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lánczos algorithm for unsymmetric matrices*, Math. Comp. **44** (1985), no. 169, 105–124.
42. H. Rutishauser, *Solution of eigenvalue problems with the LR-transformation*, Nat. Bur. Standards Appl. Math. Ser. (1958), no. 49, 47–81.
43. Y. Saad, *On the rates of convergence of the Lanczos and the block-Lanczos methods*, SIAM J. Numer. Anal. **17** (1980), no. 5, 687–706.
44. ———, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl. **34** (1980), 269–295.
45. ———, *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems*, Math. Comp. **42** (1984), no. 166, 567–588.
46. ———, *Iterative methods for sparse linear systems*, 2nd ed., SIAM, Philadelphia, PA, 2003.
47. ———, *Numerical methods for large eigenvalue problems*, 2nd ed., SIAM, Philadelphia, PA, 2011.
48. M. Sadkane, *A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices*, Numer. Math. **64** (1993), no. 1, 181–193.
49. A. Schönhage, *Zur quadratischen Konvergenz des Jacobi-Verfahrens*, Numer. Math. **6** (1964), 410–412.
50. H. D. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl. **61** (1984), 101–131.
51. G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl. **17** (1996), no. 2, 401–425.
52. ———, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev. **42** (2000), no. 2, 267–293.
53. D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl. **13** (1992), no. 1, 357–385.
54. ———, *Numerical methods for large eigenvalue problems*, Acta Numer. **11** (2002), 519–584.
55. G. Stewart, *Matrix algorithms: Volume II: Eigensystems*, SIAM, 2001.

56. G. W. Stewart, *Introduction to matrix computations*, Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, 1973, Computer Science and Applied Mathematics.
57. G. W. Stewart and J. G. Sun, *Matrix perturbation theory*, Computer Science and Scientific Computing, Academic Press, Inc., Boston, MA, 1990.
58. H. P. M. van Kempen, *On the quadratic convergence of the special cyclic Jacobi method.*, Numer. Math. **9** (1966), 19–22.
59. A. J. Wathen, *Preconditioning*, Acta Numer. **24** (2015), 329–376.
60. D. S. Watkins, *Understanding the* QR *algorithm*, SIAM Rev. **24** (1982), no. 4, 427–440.
61. ______, *The transmission of shifts and shift blurring in the QR algorithm*, Linear Algebra Appl. **241** (1996), 877–896.
62. ______, *The matrix eigenvalue problem: GR and Krylov subspace methods*, SIAM, 2007.
63. ______, *The* QR *algorithm revisited*, SIAM Rev. **50** (2008), no. 1, 133–145.
64. ______, *Francis's algorithm*, Amer. Math. Monthly **118** (2011), no. 5, 387–403.
65. H. Wielandt, *Das Iterationsverfahren bei nicht selbstadjungierten linearen Eigenwertaufgaben*, Math. Z. **50** (1944), 93–143.
66. J. H. Wilkinson, *The algebraic eigenvalue problem*, Monogr. Numer. Anal., Clarendon Press, Oxford, 1965.
67. K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl. **22** (2000), no. 2, 602–616.

Department of Computer Science, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada
E-mail address: `jbosch@cs.ubc.ca, greif@cs.ubc.ca`